LA CONCEPTION DES SYSTÈMES D'INFORMATION : ÉTAT DE L'ART ET NOUVELLES PERSPECTIVES

NDLR.– *Cet article est extrait de André Flory et Colette Rolland (textes présentés par) (1990),* Nouvelles perspectives des systèmes d'information*, sélection d'articles du congrès 90 de l'Association informatique des organisations et systèmes d'information et de décision, Paris, Éditions Eyrolles, p. 3-40.*

C. ROLLAND
Université Paris 1
UFR 06
17, place de la Sorbonne
PARIS cédex 05

A. FLORY
Laboratoire d'informatique appliquée
I.N.S.A.
Bâtiment 502
VILLEURBANNE 69621

RÉSUMÉ
Ce papier présente un état de l'art des techniques existantes de modélisation des systèmes d'information. Les deux grands types de méthodes : méthodes cartésiennes et méthodes systémiques sont analysées.

Dans une seconde partie, nous introduirons les deux tendances importantes des techniques de modélisation d'aujourd'hui. Tout d'abord, les techniques orientées objets, qui proposent une approche unifiée des données et des traitements, ainsi que les techniques fondées sur les outils, en particulier, les ateliers de génie logiciel.

Mots clés : Techniques de modélisation. Système d'information. Atelier de génie logiciel. Représentation des connaissances.

ABSTRACT
This paper first deals with a survey of existing modeling techniques. This state of art of the modeling techniques (hierarchical decomposition techniques and data semantic oriented techniques) is presented. In the second part, we introduce two major trends in to-day modeling techniques : first the object oriented techniques which propose an unified approach of data and process and also the tool based techniques and particulary CASE tools.

Key words : Modeling techniques. Information system. CASE tools. Knowledge representation.

1. INTRODUCTION

L'utilisation des systèmes d'information (S.I.) dans l'entreprise est devenue une réalité quotidienne. La gestion des S.I. et leur conception représentent donc aujourd'hui un problème majeur des organisations.

Le but de ce papier introductif est donc, d'une part, de faire le point sur les méthodes et modèles existants et d'autre part, de tenter de dégager les tendances et perspectives des outils, méthodes et modèles qui seront utilisés dans la décennie qui commence.

Il apparaît, en effet, que les méthodes ou outils que l'on a développés ne permettent pas de prendre en compte les nouvelles applications demandées par les utilisateurs. Nous citerons, en particulier, les bases de données actives (CLA89) qui nécessitent une étude sérieuse de la dynamique, les approches objets (KIN89) qui, avec l'apparition de SGBD commerciaux orientés objets, vont profondément transformer les outils informatiques et, enfin, l'utilisation des outils intelligents ou CASE : *Computer Assisted Software Engineering* (souvent baptisé *Atelier de génie logiciel*) qui introduisent une façon de procéder, tout à fait nouvelle, dans le processus de conception en introduisant des possibilités automatiques de documentation, de spécification et d'implantation des systèmes informatiques.

2. ÉTAT DE L'ART DES MÉTHODES

2.1 Les méthodes cartésiennes

Les méthodes de conception de systèmes d'information (S.I.) de la première génération sont basées sur les concepts et techniques de décomposition hiérarchique (D.H.) des processus et de flux de données (ou flux d'information). Elles sont apparues dans les années 60 et sont, probablement, les plus utilisées dans le monde.

Ces méthodes associent au *paradigme cartésien* une *approche fonctionnelle de conception*.

L'approche fonctionnelle

Les méthodes cartésiennes préconisent d'analyser et de concevoir le système d'information en se centrant sur ses fonctions. Elles le perçoivent comme un système de traitement de l'information qui répond aux règles de procédures de gestion pour produire des sorties.

L'analyse et la conception du système débutent par l'identification du S.I. à une fonction globale de gestion.

La conception du système d'information est alors assimilée à l'analyse de la fonction.

Le paradigme cartésien

Celui-ci est effectué selon une démarche descendante « top-down », de haut en bas, qui part du général, va vers le particulier et met en œuvre le principe de Descartes. Cette démarche conduit l'analyste à décomposer la boîte initiale (la fonction de gestion) en autant de boîtes qu'il le faut pour parvenir à des boîtes dont le contenu soit intelligible. La fonction de gestion est éclatée en un arbre de processus comme l'illustre la figure 1.

Cet exemple est extrait du cas d'application de la méthode Information Engineering de MARTIN (MAR86) à la conception d'un système d'information de gestion de conférences internationales.

L'analyse et la conception sont basées sur une technique de raffinements successifs utilisant le principe d'abstraction : au plus haut niveau d'abstraction, le processus « organisation de la conférence » est vu comme un tout (représenté par la boîte de même nom); pour être à un niveau d'abstraction plus fin, il est perçu comme trois processus complémentaires.

MÉTHODES CARTÉSIENNES
DÉCOMPOSITION TOP-DOWN

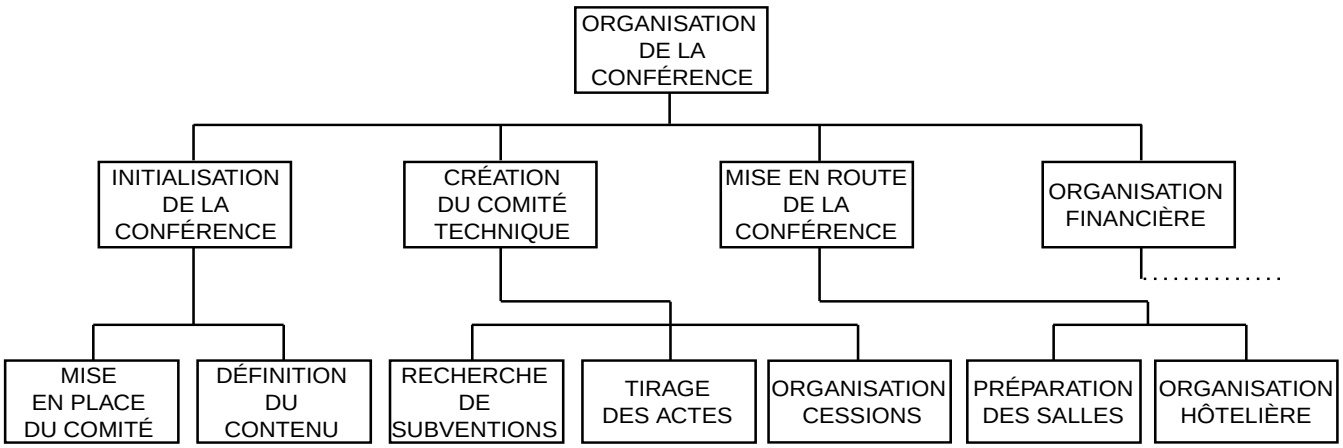


FIGURE 1

Les méthodes cartésiennes mettent l'accent sur la *modélisation des processus,* fondée sur une technique d'analyse qui consiste à éclater une fonction globalement perçue en processus spécifiques.

La décomposition cherche également à mettre en évidence les interrelations entre processus au moyen de flux de données, de produits, de signaux et autres moyens. La figure 2 illustre par exemple, le diagramme de dépendance des processus qui doit être tracé dans la méthode Information Engineering lorsque l'ensemble des processus élémentaires a été défini (mais le niveau de détail n'est pas prescrit par la méthode). Le diagramme montre les dépendances entre processus, les noms des flux d'information qui servent à les exprimer ainsi que des caractéristiques des dépendances telles que les cardinalités d'exécution, les conditions et événements (par exemple « réunion du comité » déclenche « Conférence »).

MÉTHODES CARTÉSIENNES
DIAGRAMME DE FLUX

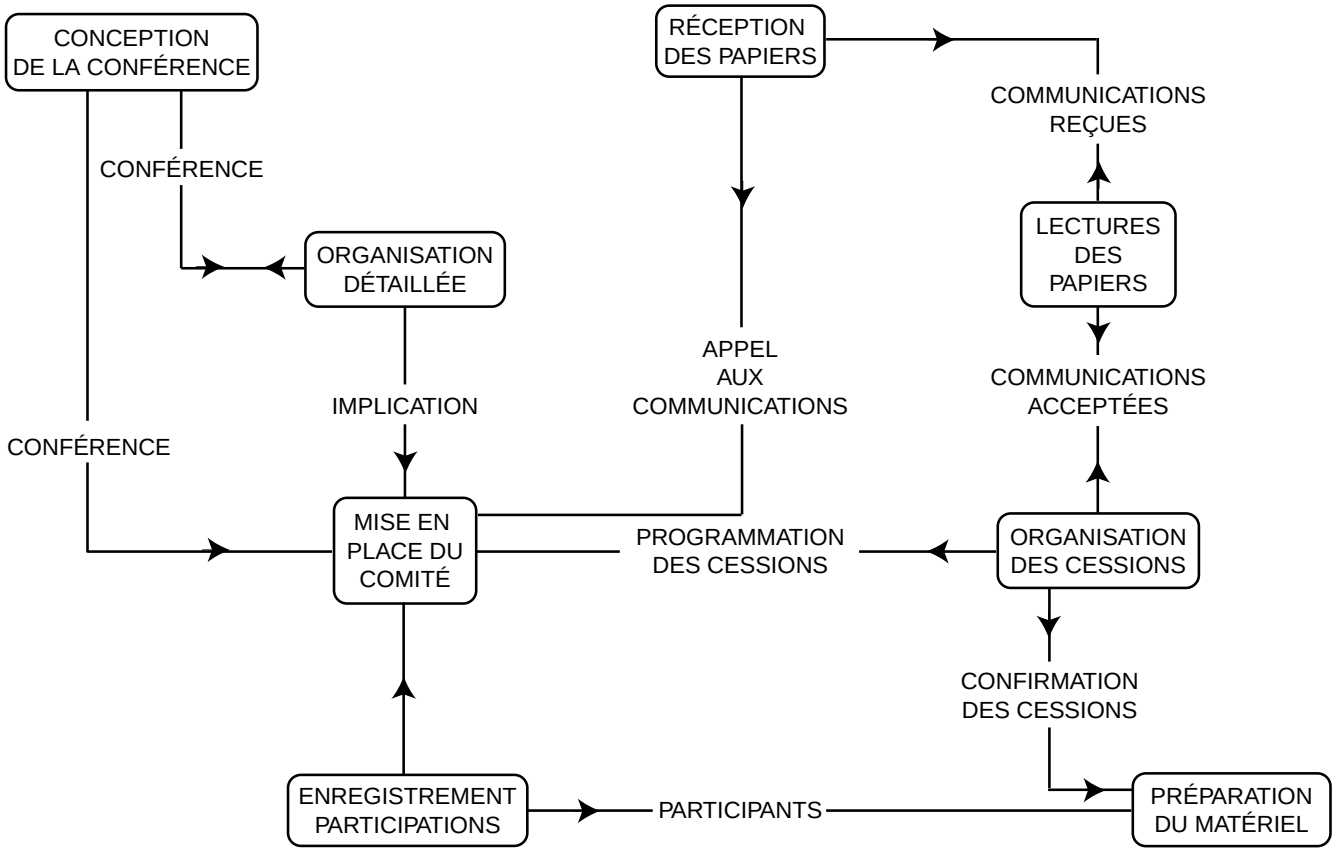


FIGURE 2

Les deux aspects, c'est-à-dire la décomposition des processus et leurs interactions, sont souvent considérés simultanément. La méthode « Structured Analysis and Design Technique » (S.A.D.T.) en est une bonne illustration. Le concept de base de S.A.D.T. est celui d'actigramme qui met l'accent sur la description des actions et leurs connexions au moyen de données. Un actigramme (figure 3) comporte les éléments suivants :

- les **actions** sont représentées par des boîtes nommées;
- les **entrées** sont les données exécutées par l'action;
- les **sorties** sont produites par l'action;
- les données de **contrôle** influencent l'action;
- les **mécanismes** représentent les ressources et outils qui aident à l'exécution de l'action.

ACTIGRAMME

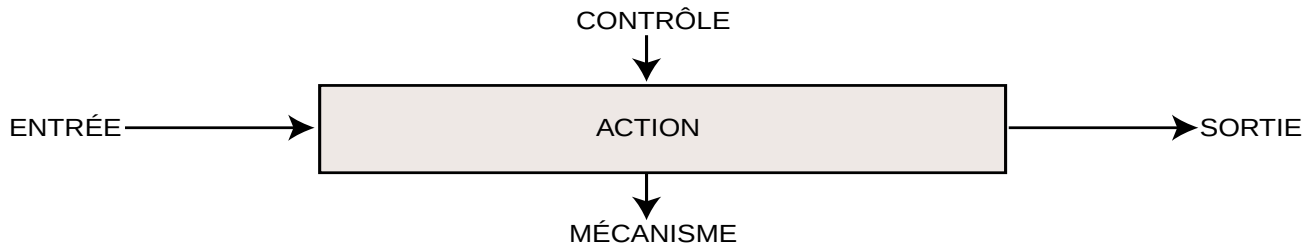


FIGURE 3

La conception d'un S.I. suivant la démarche S.A D.T. consiste à décomposer l'actigramme de plus haut niveau en passant par un nombre arbitraire de niveaux jusqu'à l'obtention d'actigrammes qui correspondent au niveau de détail souhaité. S.A.D.T. fournit des règles de conduite du raffinement. Ainsi, par exemple, S.A.D.T. préconise de décomposer un actigramme en au moins trois, et au plus six, actigrammes du niveau suivant.

Les méthodes cartésiennes ont été influencées, d'une part, par la programmation modulaire et les approches de décomposition fonctionnelle de PARNAS (PAR72) et WIRTH (WIR71) et, d'autre part, par les flux de données et les méthodes de conception structurées introduites par YOU (1979), puis développées par MEYERS (MEY78). Les méthodes I.E. (MART86), S.A.D.T. (ROSS77A, ROSSB, ROSS85), SA (ROSS77 B) ISAC (LUN82), PSL/PSA (TEI77), HIPO (STA76), SAISD (MAR78), JSD (JAC83), sont des exemples de méthodes cartésiennes parmi d'autres.

On trouvera dans (OL82) et (OL86) un bon survol des méthodes existantes parmi lesquelles un grand nombre appartient à la génération des méthodes cartésiennes.

Les méthodes fondées sur la décomposition hiérarchique des processus sont utilisées de manière intensive dans le monde entier, même si l'expérience montre que ces approches sont surtout adaptées à la description d'un système existant ou d'un système déjà conçu. Ce sont des instruments d'analyse plus que de conception.

Les méthodes cartésiennes ont des lacunes, parmi lesquelles les auteurs de cet article relèvent :

- L'absence de travaux théoriques susceptibles de fournir des fondements solides aux concepts et techniques de décomposition descendante.
- L'impression des définitions qui rendent difficile l'utilisation de concepts tels que flux de données, archive, action…
- L'impossibilité de prendre en compte le temps, la synchronisation et le parallélisme des processus. Il semble même que la synchronisation de processus soit impossible à modéliser dans une démarche descendante.
- L'incapacité à traiter les cas particuliers et d'exception, toute l'attention étant focalisée sur l'analyse des flux et activités typiques.
- L'insuffisance de la modélisation des données : en outre, la solution de juxtaposition de techniques de modélisation de données à celles de la décomposition des traitements largement

préconisée aujourd'hui, notamment dans les outils CASE (*Computer Assisted Software Engineering*), est loin d'être fondamentalement satisfaisante.

- L'absence de guide méthodologique précis rend la pratique de la méthode difficile à maîtriser. La compétence méthodologique ne peut s'acquérir que par l'expérimentation intensive.
- L'évaluation de la cohérence, de la complétude et de la qualité d'une solution est difficile.

Comme l'affirme C. FLOYD (FLO 86), il semble que les méthodes cartésiennes soient applicables à des systèmes de taille moyenne, ayant peu d'interactions homme-machine et lorsque les fonctionnalités du système sont relativement claires à l'avance. Cette critique n'implique pas que les méthodes fondées sur la décomposition descendante soient inutiles mais que, soit leur application doit être limitée aux problèmes que l'on vient de caractériser, soit qu'elles doivent être adaptées et transformées.

2.2 Les méthodes systématiques

Les méthodes de conception de S.I. de la seconde génération sont entièrement centrées sur la modélisation des données. Elles combinent une *approche conceptuelle* au *paradigme systémique*.

Le paradigme systémique

Contrairement aux méthodes cartésiennes, les approches systémiques ont leurs racines dans la théorie des systèmes. Le schéma de la figure 4 inspiré de la théorie de BOULDING (BOU56), introduit en France par J.L. LEMOIGNE (LEM77) et rendu populaire par MERISE, montre le système d'information en relation avec, d'une part, le système opérationnel de l'organisation et, d'autre part, son système de pilotage.

Le système d'information est perçu ici comme un artéfact qui fournit une représentation des faits présents et passés de la vie de l'organisation (c'est-à-dire des faits survenus dans son système opérant). Il est une mémoire collective des acteurs de l'organisation qui se souvient de l'embauche des employés, des commandes reçues, des livraisons effectuées, etc. Le S.I. est un « modèle » (dans le sens d'image abstraite) de la réalité organisationnelle qui apporte aux acteurs et décideurs la connaissance dont ils ont besoin pour agir et décider. Il mémorise sous forme de données, l'image des faits pertinents et amplifie ainsi les capacités individuelles de mémorisation des acteurs de l'organisation.

L'approche conceptuelle

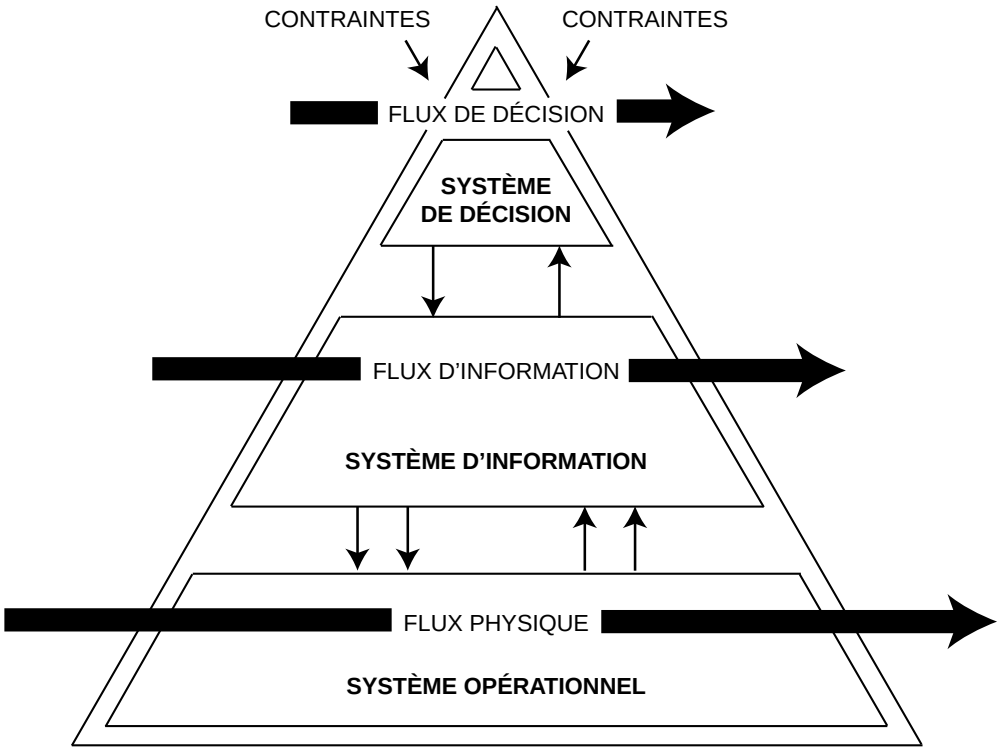
Le processus de conception du S.I. est alors assimilé à un processus de modélisation qui, naturellement, s'est centré sur la modélisation des données.

Une *donnée* est une valeur qui décrit, d'une certaine façon, un phénomène de la réalité et à partir de laquelle on peut obtenir de l'information.

L'information est l'incrément de connaissance que l'on peut inférer d'une donnée. L'inférence est basée sur une interprétation des données et de leurs relations. Un *modèle de données* est un outil intellectuel qui permet une telle interprétation.

Les premiers modèles de données fournirent des règles d'interprétation qui étaient dépendantes de la manière de stockage et d'accès aux données sur leurs supports physiques.

Au cours des quinze dernières années, l'objectif commun à toute la communauté scientifique du domaine a été de définir des modèles de données qui facilitent l'interprétation de leur sémantique et permettent la spécification du résultat de la modélisation à un haut niveau d'abstraction dans les termes de ce qui est appelé *schéma conceptuel*.



Les travaux préliminaires ont été développés autour du modèle relationnel (COD70), (COD71), (BER76), (FAG77), (BEE78). Le modèle relationnel est à proprement parler, un modèle d'implantation de données et non un modèle de représentation sémantique des connaissances.

Cependant, au travers des techniques de normalisation, le relationnel permet de prendre en compte une partie de la sémantique du mondé réel. La normalisation des relations est en fait, un processus de réduction qui permet d'obtenir des relations indécomposables. Une relation est décomposable, si elle ne possède pas, en son sein, des dépendances autres que celles dont l'origine est la clé.

Lorsqu'il existe une dépendance fonctionnelle (D.F.) issue d'un sous-ensemble de la clé, la relation est en première forme normale, lorsqu'il existe une dépendance fonctionnelle non issue de la clé, la relation est en deuxième forme normale (à condition qu'il n'existe pas de dépendances fonctionnelles issues d'un sous-ensemble de la clé). Dans les autres cas, elle est en troisième forme normale ou en forme normale de Boyce-Codd (dans ce cas, toutes les D.F. de la relation sont issues de la clé). Lorsqu'une relation n'a pas de dépendance fonctionnelle (relation réduite à la clé), elle est en quatrième forme normale, s'il n'existe pas une dépendance multivaluée au sens de la relation et en cinquième forme normale, s'il n'existe pas de dépendance de jointure.

Une relation est « parfaitement normalisée » et donc implémentable, si elle ne peut pas être décomposée.

Toutes les dépendances induisent des contraintes et la normalisation représente une forme de matérialisation de ces contraintes.

Il n'existe pas vraiment de méthode à proprement parler qui utilise strictement le modèle relationnel.

Le processus de décomposition (on part d'un ensemble de relations et d'un ensemble de dépendances) conduit à casser les relations en des relations plus petites qui sont normalisées. Le processus de composition, au contraire, part d'un ensemble d'attributs et de dépendances et se propose de regrouper les attributs en relations normalisées en fonction des dépendances qui lient les attributs.

Les relations normalisées représentent donc des groupes homogènes, au sens de la sémantique.

Cependant le modèle relationnel reste très limité, dans ses possibilités de représentation des informations. Il ne permet pas, en particulier, de visualiser les données et les liens qui existent entre elles.

C'est au contraire, le principal objectif des modèles sémantiques tels que le modèle *Entité-association* introduit par P. CHEN (CHEN76). P. Chen suggère de voir toute réalité comme composée d'entités ayant entre elles des associations.

Une *entité* est une chose qui peut être distinctement identifiée : une personne, une facture ou une compagnie…

Une *association* est une combinaison d'entités dans laquelle chacune d'elles joue un rôle spécifique. La relation « père-fils » est une association entre deux personnes qui sont des entités. Les entités et les associations sont caractérisées par des *propriétés* : le nom de la personne, le total de la facture, la raison sociale de la compagnie et la durée de la relation père-fils entre Jean et Paul.

Les entités, associations et propriétés sont classées et définies par des types : l'entité type « Employé », l'association type « Père-fils » ou la propriété type « Nom ». Un type définit en extension et en intention une population d'objets de la même nature.

Le modèle *Entité-association* fournit des techniques de représentation sous forme de diagrammes d'un schéma conceptuel qui sont illustrées à la figure 5.

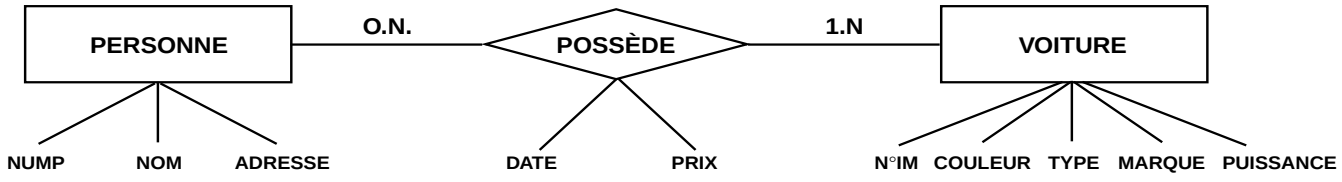


FIGURE 5

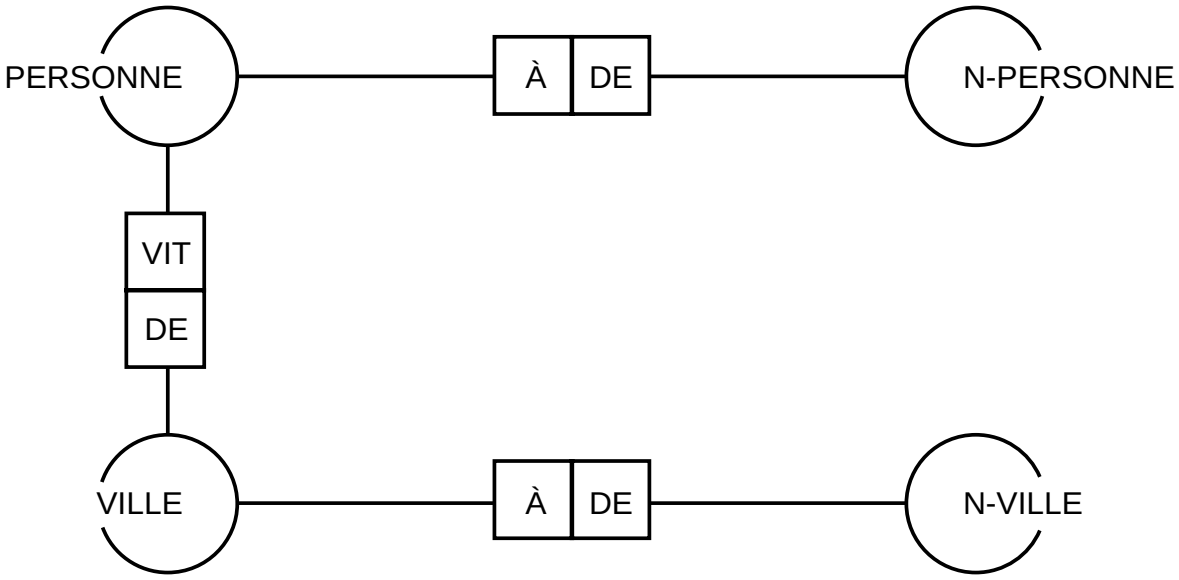
Le pouvoir d'expression et la simplicité du modèle *Entité-association* sont, sans nul doute, les raisons de son succès, tant auprès de la communauté des chercheurs que dans le milieu professionnel. Il est le modèle de données le plus utilisé dans le monde pour la construction de schémas conceptuels. Il a

cependant des insuffisances qui ont justifié de nombreuses propositions d'extensions dont on trouvera les développements dans les actes des Conférences « Entity Relationship approaches » qui se succèdent depuis 1979.

On peut enfin noter que de nombreux auteurs se sont attachés à définir des règles de passage d'un schéma Entité-association en une collection de schémas relationnels.

Les *modèles binaires* adoptent la même vision du monde réel, mais restreignent les associations à des valeurs binaires entre entités. Par exemple, le modèle de la méthode NIAM (*Nijssen Information Analysis Methodology*) développé par Nijssen (NIJ88), propose le fait qu'une PERSONNE ait un NOM ou qu'une PERSONNE habite une certaine VILLE se représente de la même manière (figure 6) par un lien binaire. PERSONNE et VILLE sont des NOLOT (*Non Lexical Object Types*). La démarche de modélisation NIAM est de type linguistique : elle conduit à représenter les faits (par exemple Pierre vit à Paris) qui s'expriment au moyen d'objets lexicaux (Pierre et Paris) par une structure qui attache les objets lexicaux types (NOM-PERSONNE et NOM-VILLE) aux objets non lexicaux types correspondants (PERSONNE et VILLE). NIAM a cette particularité de permettre de représenter dans un même schéma, les classes de phénomènes réels qui ont de l'intérêt pour le modélisateur (les NOLOT) et de montrer comment il a décidé de les décrire par des données (les LOT).

« PIERRE » VIT À « PARIS »



LA *PERSONNE* DE **NOM** PIERRE
VIT DANS LA
VILLE DE **NOM** PARIS

FIGURE 6

Les *modèles sémantiques* qui utilisent les réseaux introduits dans le monde de la modélisation des données par SMITH (SMI77) sont les plus récents.

Dans le modèle de Smith, les entités types et les associations types sont des objets. Les objets peuvent être associés par agrégation et généralisation afin de définir des objets plus complexes, qui, à leur tour, peuvent participer à d'autres généralisations et agrégations. La collection des données est vue comme une hiérarchie d'objets (figure 7).

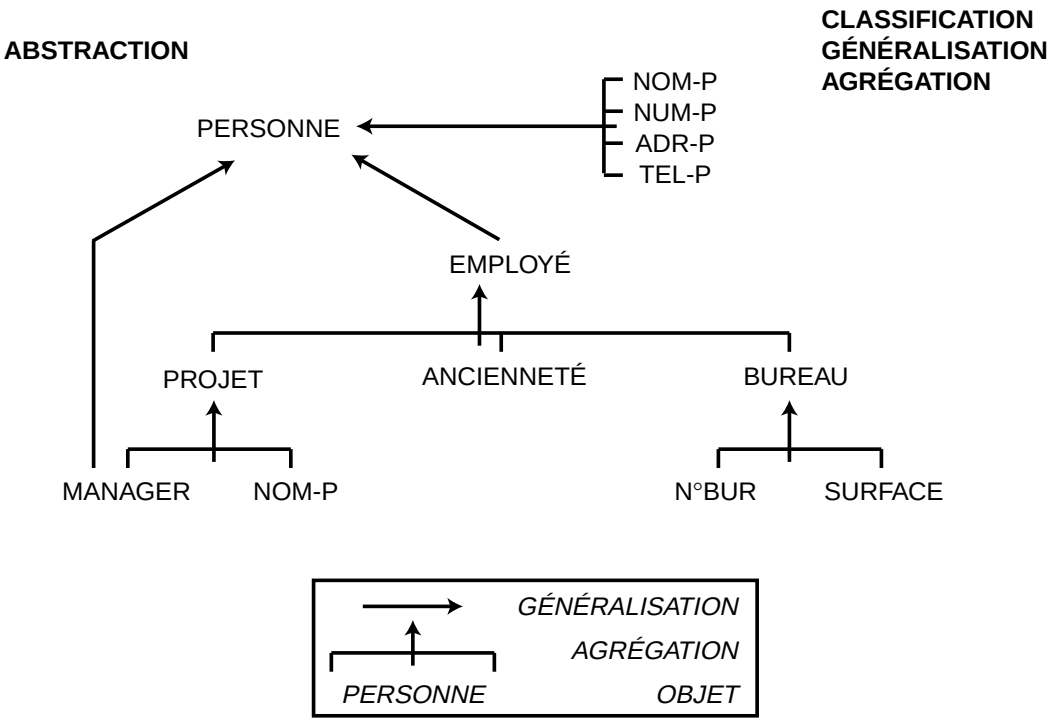


FIGURE 7

La structuration des objets dans ce modèle s'appuie sur le principe d'abstraction. Abstraire signifie cocher certains détails pour se concentrer sur des propriétés générales et communes à un ensemble d'objets. Il y a trois formes d'abstraction retenues : la classification, l'agrégation et la généralisation.

La *classification* permet de distinguer le niveau des instances de celui des *classes*. Une classe d'objets regroupe un ensemble d'instances de la même nature. Tous les modèles de données utilisent le principe de la classification. En outre, le lien de classification entre une instance et sa classe n'est pas illustré dans la représentation graphique du schéma. Les modèles sémantiques utilisent le terme d'objet pour faire référence à une classe et celui d'instance d'objet pour parler d'un élément de la classe.

Dans certains cas tels que TAXIS développés par MYLOPOULOS (MYL80) ou SDM de HAMMER et McLEOD (HAM81), la classification peut être appliquée aux classes elles-mêmes. Ceci permet d'introduire la notion de *méta-classe*.

L'*agrégation* est une abstraction grâce à laquelle la relation entre objets (appelés *composants*) est vue comme un unique objet (appelé *agrégat*).

AGRÉGATION

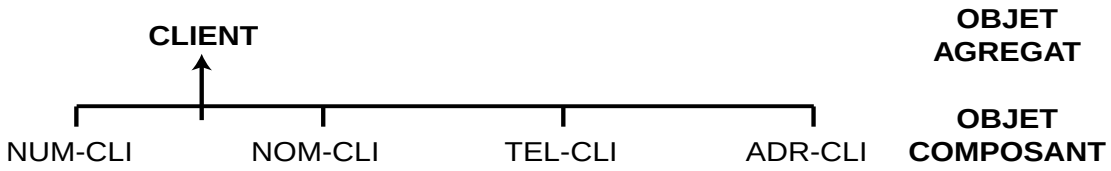


FIGURE 8

L'agrégation s'applique au niveau des classes comme à celui des instances. Elle peut par ailleurs être utilisée de manière répétitive ce qui conduit à une hiérarchie d'objets, telle qu'elle est illustrée sur la figure 9.

L'agrégation est une forme d'abstraction utile car elle permet de rendre peu à peu visible la structure d'un objet en montrant comment il est en relation avec ses objets composants. Comme le font remarquer Smith et Smith (SMI77), l'agrégation peut être utilisée ou bien dans une approche descendante pour structurer un objet complexe en le décomposant ou bien dans une approche ascendante pour regrouper des objets composants dans un objet agrégat.

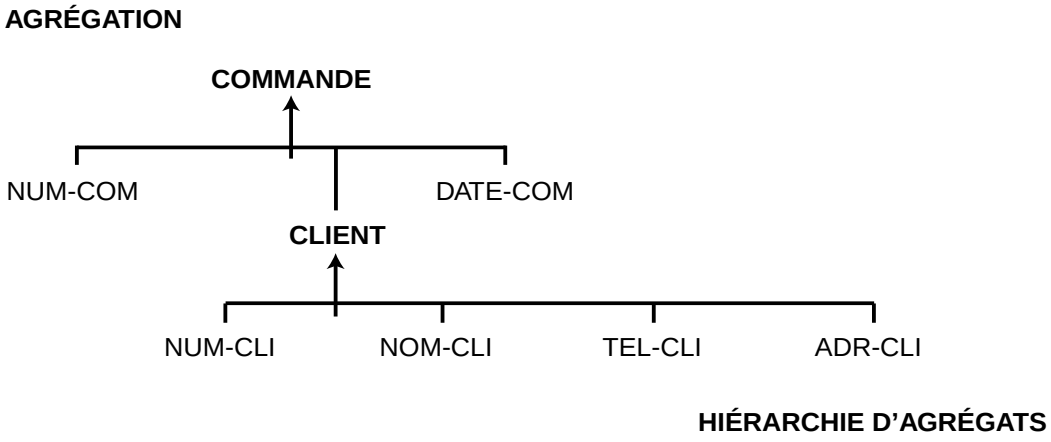


FIGURE 9

L'approche ascendante est de type synthétique et nous permet de comprendre un objet complexe. Au contraire, l'approche descendante est analytique et sert à concevoir un objet complexe.

Les deux approches peuvent être combinées au cours du processus de conception d'un système d'information. L'utilisation de l'agrégation n'est pas nouvelle. Dans le modèle relationnel, un schéma de relation est construit par agrégation d'attributs. Dans les modèles sémantiques, l'agrégation a été étendue de façon à permettre que les objets composants soient eux-mêmes complexes.

La *généralisation* introduite également par Smith et Smith est une caractéristique des modèles sémantiques qui est aujourd'hui très largement utilisée. Il s'agit d'une forme d'abstraction qui permet de voir un ensemble d'objets appelés *spécialisés* comme un objet complexe unique appelé *générique*.

Voir les objets EMPLOYÉ et ÉTUDIANT comme un seul objet PERSONNE correspond à une généralisation des types EMPLOYÉ et ÉTUDIANT au travers du type PERSONNE.

La généralisation exprime un lien appelé IS.A. entre un objet spécialisé et un objet générique. Dans son acception la plus courante, la relation IS.A traduit une inclusion d'objets.

« A IS.A B » signifie que toute instance de A est aussi une instance de B. (A est un B).

L'application répétée de la généralisation conduit à une hiérarchie IS.A dont les principales propriétés sont :

- *L'héritage de propriétés.* Les propriétés de l'objet générique sont *héritées* par les objets spécialisés. Ainsi, le fait qu'une personne a un nom, un numéro et une adresse est hérité par chacun des objets spécialisés de PERSONNE (c'est-à-dire ÉTUDIANT et EMPLOYÉ). Ceux-ci peuvent avoir des propriétés propres. En outre, certains modèles (TAXIS par exemple) acceptent une redéfinition des propriétés héritées.
- *L'héritage multiple.* Il est généralement admis qu'un objet puisse être généralisé de plusieurs façons. Par exemple, un ASSISTANT d'UNIVERSITÉ peut être un ENSEIGNANT mais aussi peut être un ÉTUDIANT. Ceci pose problème dès l'instant où l'objet spécialisé hérite de la même propriété de plusieurs objets génériques. Dans l'exemple de la figure 10, un assistant hérite de la propriété « nombre d'heures de cours » qui peut provenir soit de la classe enseignant, soit de la classe étudiant alors que, pour un même individu, ces propriétés ne prennent pas les mêmes valeurs.

HÉRITAGE MULTIPLE

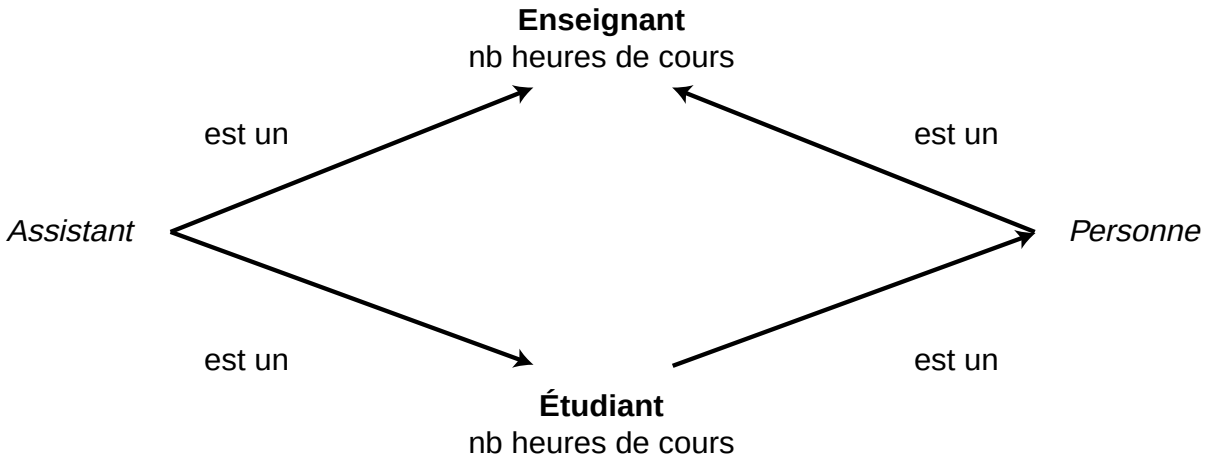


FIGURE 10

- *L'exclusion*. La *contrainte d'exclusion* entre deux objets spécialisés permet d'exprimer que les instances des classes correspondantes s'excluent. Des extensions ont été proposées par D. Smith (SMI77) avec la notion de *cluster* et par E. Schiel (SCH3) au moyen du concept de *rôle.*

La généralisation est un mécanisme de modélisation puissant qui permet de classer des objets distincts dans d'autres objets plus généraux. Elle permet :

- de simplifier le schéma conceptuel en utilisant le principe d'héritage;
- de raisonner à différents niveaux d'abstractions :
  - au niveau générique : le concepteur raisonne sur les propriétés communes à plusieurs objets,
  - au niveau spécialisé : il décrit les propriétés spécifiques des objets;
- de décrire plus précisément et plus finement la réalité organisation-réelle.

Utilisées de façon combinée, la généralisation et l'agrégation permettent d'exprimer à la fois la structure et la classification des objets. La structure d'une classe générique peut être définie comme un agrégat d'objets composants et les agrégats peuvent être classés par rapport à un objet générique. La classification d'un objet peut se faire dans une hiérarchie d'agrégats. Les deux hiérarchies doivent être comprises comme deux vues orthogonales de la même réalité.

2.3 Les méthodes assistées par des outils

Les dernières années de la décennie 80 ont vu naître des produits d'assistance au développement des systèmes d'information. La troisième génération de méthodes de conception de S.I. est basée sur des systèmes d'outils d'aide au développement souvent appelés outils CASE. Ces outils sont fréquemment implantés sur des postes de travail individuels. Ils comportent un dictionnaire de données, des interfaces graphiques, des modules de documentation, des moyens de contrôle de la qualité et de la correction des schémas et, de façon limitée, des générateurs de code.

Ces produits ont pour ambition d'accroître la productivité des équipes de développement et de permettre le passage de l'artisanat à une production industrielle des systèmes d'information.

Ils souhaitent parvenir à l'automatisation du processus de développement des S.I. dont le cadre est défini par les méthodes. Mais il s'agit d'un idéal que les outils actuels sont loin d'atteindre.

L'apport des outils est plus dans la gestion automatisée des spécifications (des schémas des différents niveaux : conceptuel, logique, physique, externe) que dans l'automatisation du processus par lequel les spécifications sont produites.

Cette situation s'explique par l'existence de nombreux modèles et langages permettant d'établir des spécifications et sur lesquels les outils ont pu s'appuyer. L'absence totale de formalisation des processus de conception rend impossible leur automatisation dans l'état actuel des connaissances.

L'architecture et les fonctions de l'archétype d'outil du marché sont illustrées à la figure 11.

- Elle montre que le dictionnaire de spécifications (également appelé encyclopédie, méta-base, dictionnaire de données…) est l'élément central de tout outil CASE. Le dictionnaire sert à mémoriser les éléments des différents schémas préconisés par la méthode assistée par l'outil. Le dictionnaire est une méta-base de données dans la mesure où les données de cette base sont les types décrits dans les schémas du S.I. en cours de construction.
  La structure du dictionnaire, le modèle qui sert à la définir, la notion de méta-modèle permettent de définir le modèle lui-même comme une instance du méta-modèle. Ils sont des éléments importants pour l'évaluation d'un outil CASE.

ARCHITECTURE ET FONCTIONS DE L'ARCHÉTYPE D'OUTIL DU MARCHÉ

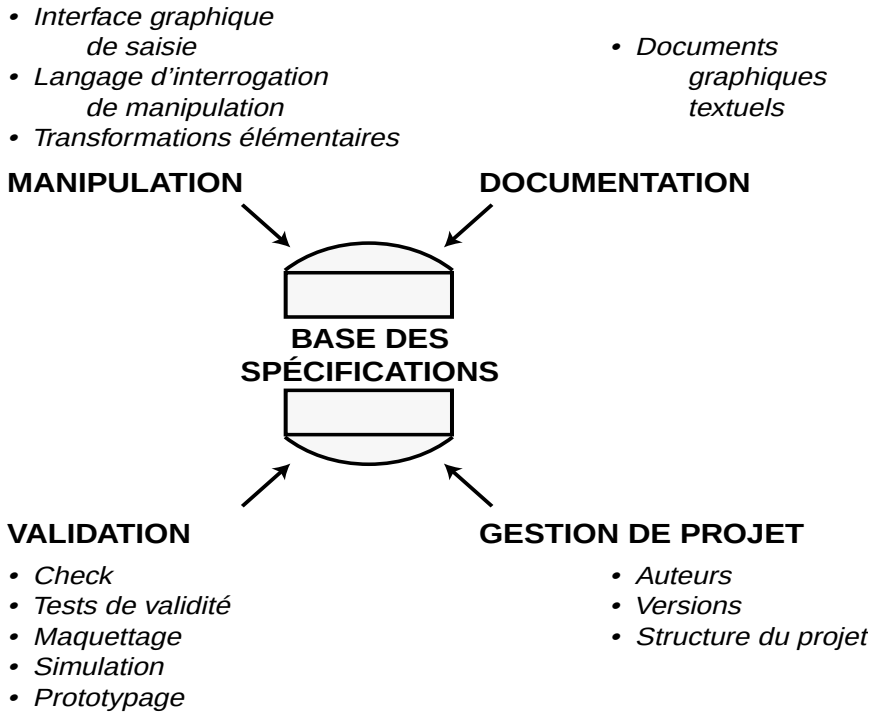


FIGURE 11

- L'accès au dictionnaire de spécifications est en général, assuré par un éditeur graphique. Les spécifications sont entrées sous forme de diagrammes. Leur contenu et leur forme sont stockés dans le dictionnaire.
  Dans la mesure où le dictionnaire est une base de données gérée par un SGBD, on conçoit que l'accès au dictionnaire soit aisé.
  Dans certains cas, l'utilisation de l'outil aura à sa disposition un langage de requête, tel que SQL, lui autorisant toute question au dictionnaire (voir par exemple, le logiciel GRAPHTALK de Rank Xérox). Dans d'autres cas, il sera limité à l'usage de requêtes prédéfinies accessibles par un menu.
- Certaines transformations, telles que, par exemple, le passage d'un schéma Entité/Association en une collection de schémas de relations ou la normalisation d'une telle collection sont assurées par les outils.
- La fonction de validation est implantée à différents degrés de sophistication, allant du simple contrôle de nature syntaxique (vérification que les définitions des concepts des modèles sont respectées) à des aides à la visualisation de schémas faisant appel aux techniques de maquettage, prototypage et simulation.
- La documentation automatique est un confort apporté par les outils que les équipes de développement ne peuvent qu'apprécier. L'intégration dans le dictionnaire, de toute l'information relative aux spécifications d'un projet de S.I. permet de disposer d'une documentation à jour et cohérente (dans la mesure où les spécifications le sont).

Les outils CASE n'assistent que partiellement les méthodes. Le concepteur (développeur) suit une démarche manuelle et se sert de l'outil pour l'aider à gérer les spécifications qu'il élabore au fur et à mesure où il progresse dans le processus de développement.

Dans une étape ultérieure, les outils devront assister le développeur dans sa démarche en le guidant vers la solution, en automatisant une partie de ses activités de conception, en l'aidant à raisonner sur des spécifications et en réutilisant des solutions déjà construites. Il faut pour cela que les outils évoluent, de simples gestionnaires de spécifications qu'ils sont aujourd'hui, vers des outils plus intelligents, capables de se comporter comme le fait (consciemment ou inconsciemment) l'expert concepteur. Ceci requiert sans doute une nouvelle formulation du processus de conception qui sera discuté dans le chapitre suivant.

Notons, pour conclure, que les outils ont, dans leur grande majorité, retenu une démarche de conception qui juxtapose la modélisation des données selon les termes des méthodes systémiques à la modélisation des traitements, en accord avec les techniques de décomposition descendantes des processus.

Ceci oblige les outils à intégrer des contrôles de vérification croisée entre schémas de données et schémas de traitement, de façon à éviter qu'il y ait une trop grande divergence des solutions conçues sur le plan des données indépendamment de celles qui le sont sur le plan des traitements.

3. NOUVELLES PERSPECTIVES

On se propose de présenter dans ce chapitre, trois des tendances émergentes en matière de méthodes et d'outils de conception de systèmes d'information. Ce sont :

- Les approches de la dynamique
- L'orientation objet des méthodes
- Les outils « intelligents »

3.1 Les approches de la dynamique

Ces approches suggèrent d'intégrer dans un même schéma conceptuel la modélisation de la structure et celle du comportement du futur système d'information. Elles réfutent la dichotomie données-traitements et peuvent être assimilées à une extension des approches orientées données intégrant la modélisation des lois de comportement des données. Les méthodes REMORA (ROL82), ACM-PCM (BRO82), TAXIS (MYL80), RML (BOR85), SDM (HAM81), RUBRIC (ASS89) sont des exemples de cette classe.

Ces méthodes ont un cadre commun de modélisation des objets de la réalité : les objets varient dans le temps, ils changent d'état, ont une vie faite de changements successifs. Les objets ont à la fois des propriétés de structure et des propriétés de comportement. La structure d'un objet décrit ses *intra* et *inter relations* avec d'autres objets. La dynamique exprime *quand* et *comment* les objets changent d'état.

Ainsi, dans la méthode REMORA (ROL82), les changements d'état d'objets résultent de l'exécution d'*opérations*. Une opération REMORA est élémentaire car elle n'altère l'état que d'un seul objet.

Ouvrir une chambre dans un hôtel, modifier l'adresse d'un client, annuler une réservation sont des exemples d'opérations.

L'exécution d'une opération résulte de la survenance d'un *événement*. Un événement est quelque chose qui survient instantanément, à un moment particulier du temps. Les événements n'ont pas d'histoire, pas de vie. Quand ils sont survenus, ils ne peuvent être ni supprimés ni modifiés. En REMORA, un événement survient lorsqu'un changement d'état particulier s'est passé sur un objet. Les événements correspondent à des changements d'état particulier d'objets qui déclenchent certaines opérations. Par exemple, la disponibilité d'un exemplaire d'un ouvrage dans une bibliothèque déclenche la réservation de cet exemplaire pour un abonné qui l'attend, la convocation de l'abonné et la mise à jour de son nombre potentiel d'emprunts. Comme l'illustre la figure 12 sur cet exemple, la modélisation conceptuelle REMORA se centre sur la spécification des transitions d'état à état que le système d'information va subir au cours du temps. Une transition résulte de la survenance d'un événement (par exemple, le fait qu'un exemplaire d'ouvrage qui était non disponible le devienne) et déclenche l'exécution d'un ensemble d'opérations éventuellement sous condition (les trois opérations ne sont déclenchées que si la condition C1, statuant qu'il existe un abonné en attente de cet ouvrage, est vraie).

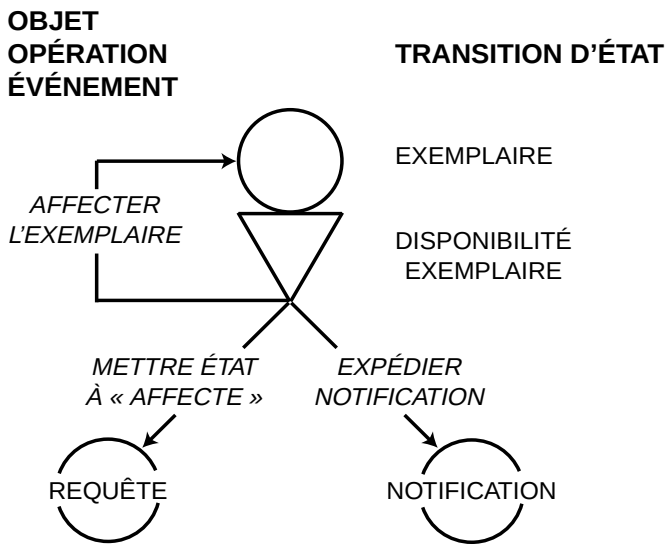


FIGURE 12

Une transition doit faire passer le système d'objets d'un état réputé cohérent à un autre état cohérent.

Les méthodes de la dynamique sont sensibilisées aux problèmes d'évolution cohérente des données d'un système d'information. Elles cherchent à spécifier de façon aussi rigoureuse que possible, ce que sont les objets informationnels d'un S.I., comment ils évoluent dans le temps et selon quelles lois et enfin quand ces changements se produisent.

Le temps est partie intégrante de tout modèle orienté « dynamique ». Par exemple, dans la modélisation préconisée par la méthode CIAM (GUS82), tout attribut caractéristique d'une entité type qui dépend du

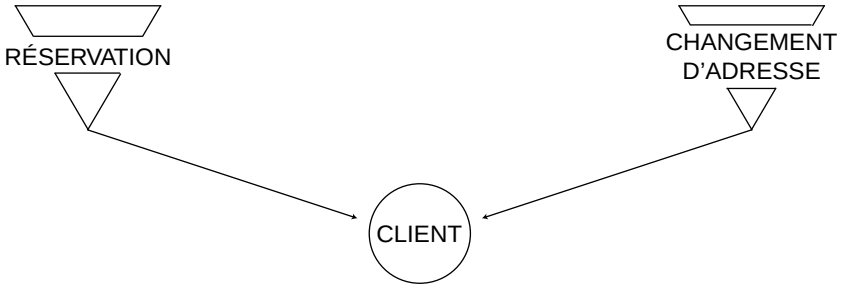
temps est défini comme une fonction du temps. Aussi, l'adresse d'un client à un instant t est une fonction du temps telle que :

Client (x), time (t) ⟶ adresse (x,t)

La conceptualisation dans une perspective de temps non restreint est déclarative dans la mesure où l'état des objets du S.I. se déduit des événements survenus sur ces objets au moyen de règles de déduction.

La figure 13 illustre de quelle façon CIAM permet de spécifier que l'adresse d'un client à n'importe quel instant t se déduit des propriétés des événements « changement d'adresse » survenus avant t pour ce client. En fait, la règle exprime que l'adresse du client à l'instant t est la valeur de l'adresse du dernier événement « changement d'adresse » pour ce client avant t.

CHANGEMENT D'ÉTAT ÉVÉNEMENT-ENTITÉ



CONDITION D'EXISTENCE

$\forall x \ \forall d \ \exists y$ (RÉSERVATION (y) $\wedge$ personne (y) = x $\wedge$ day (y) < d $\longrightarrow$ client (x,d)

RÈGLES DE DÉRIVATION

$\forall x \ \forall d \ \exists y$ (CHANGEMENT ADRESSE (y)

client (y) = x $\wedge$ day (y) < d

$\wedge \nexists z$ (CHANGEMENT-D-ADRESSE (z)

$\wedge$ client (z) = x $\wedge$ day (z) > $\longrightarrow$ adresse (x,d) = adresse (y)))

FIGURE 13

Les approches de la dynamique déductive ne font pas d'hypothèse sur la façon dont le système d'information stocke les états successifs des objets. Elles sont basées, au contraire, sur l'idée que ce sont les événements survenus sur les objets qui sont mémorisés et permettent de déduire l'état dans lequel se trouvent les objets lorsque c'est nécessaire.

La modélisation du temps est un sujet important, totalement sous-traité par les méthodes cartésiennes et très insuffisamment pris en compte par les méthodes systématiques orientées « données ». Elle est, au contraire, un aspect majeur de toute méthode relevant d'une approche de la dynamique. Schématiquement, les travaux relevant de ce problème satisfont deux types d'objectifs :

- définir des modèles du temps;
- choisir des procédés d'estampillage des phénomènes évoluant dans le temps.

Le « temps générique » introduit par K. Anderson (AND82), l'approche ensembliste des « domaines temporels » proposée par J. Clifford (CLI87) et les « abstractions temporelles » définies par Bolour (BOL83) sont des exemples significatifs de travaux du premier type. Comme exemples du second, on peut mentionner l'approche CIAM (GUS82), le modèle REMORA des c-objets (ROL82) et la solution de DADES (OLI82) qui toutes relèvent de la logique des prédicats du premier ordre.

La tendance actuelle vise à utiliser des logiques non standards telles que la logique temporelle. Les six opérateurs temporels suivants ont par exemple été retenus dans les méthodes ERAE (DUB86), TEMPORA (TEM89), OBLOG (SER89) et CPL (DIG89).

- $\bigcirc \phi$ signifie que ϕ est vrai dans l'état suivant
- $\Diamond \phi$ signifie que ϕ est vrai dans l'état futur
- $\nabla \phi$ signifie que ϕ est vrai dans tous les états futurs
- $\bullet \phi$ signifie que ϕ est vrai dans l'état précédent
- $\blacklozenge \phi$ signifie que ϕ est vrai dans un état passé
- $\blacktriangledown \phi$ signifie que ϕ est vrai dans tous les états passés

Des solutions plus sophistiquées sont proposées par d'autres auteurs pour distinguer le nécessaire du possible. Par exemple, Dignum et All définissent une logique déontique basée sur les deux opérateurs de *permission* et d'*obligation*.

Par exemple, $p, b\ (\text{Person}(p) \wedge \text{book}(b)$

$\Longrightarrow$ $(\text{borrow}(p,b))\ \text{OBL}\ (\text{return}(p,b) \leq 15\ \text{days}))$

signifie que « si une personne p emprunte un livre b, elle doit le rapporter dans un délai de 15 jours ». Cependant, il n'est pas garanti qu'elle le rapporte. La contrainte déontique exprime une obligation morale de l'emprunteur.

Le concepteur peut aussi utiliser l'opérateur de nécessité (NEC). Dans ce cas, la règle :

$p, b, (\text{Person}(p) \wedge \text{book}(b)) \Longrightarrow (\text{borrow}(p,b))\ \text{NEC}\ (\text{return}(p,b) \leq 15\ \text{days}))$

signifie que « si une personne p emprunte un livre b, elle doit nécessairement le rapporter ». L'absence de rapport du livre dans un délai de 15 jours conduit à un état incohérent. La spécification conceptuelle doit inclure une règle qui permet de dire ce qui doit être fait pour sortir de l'état incohérent.

La logique déontique permet de modéliser le comportement des acteurs du système d'information et de faire la différenciation entre cohérence de l'information et satisfaction des contraintes.

En conclusion, on peut dire que les méthodes dynamiques préconisent de compléter la description des propriétés statiques dans chaque état du S.I. par celles des propriétés dynamiques en caractérisant :

a) l'interaction entre états et événements;

b) les contraintes d'ordonnancement des états.

La logique des prédicats du premier ordre est appropriée à la description des états. La logique dynamique telle que celle de Harel (HAR 84) est adaptée pour a, et les logiques temporelles ou déontiques sont pertinentes pour b.

3.2 L'orientation objet

L'intérêt des approches objets n'est plus à démontrer aujourd'hui, et on peut même se demander si ce terme n'est pas finalement souvent utilisé à tort et à travers. Ces approches, développées à l'origine pour les langages de programmation, ont largement dépassé depuis le cadre strict de la programmation, pour aborder le domaine des bases de données. On compte, en effet, aujourd'hui une demi-douzaine de SGBD « orientés objets » commercialisés ou prêts à l'être. Nous citerons, en particulier, ORION (BAN87) qui représente actuellement le produit le plus complet, IRIS (FIS89) GEMSTONE (BRE89) ainsi que deux produits français G BASE (GRA88) et O2 (BAN88).

L'intérêt croissant, pour cette approche, se justifie :

– *Pour les utilisateurs de systèmes d'information*. Beaucoup d'entre eux sont amenés à travailler sur de nouvelles applications nécessitant la représentation de nouveaux types de données (en particulier, des graphiques, textes et images) que les SGBD relationnels classiques ne peuvent traiter. En outre, les nouvelles possibilités d'interfaces homme-machine, que l'on voit se développer par l'extension du graphisme et des grands écrans, nécessitent les logiciels spécifiques pour leur gestion.
– *Pour les informaticiens*. L'apport essentiel provient du fait que ces approches fournissent une vue unifiée des données et des traitements.

Contrairement aux approches antérieures, l'objet est défini, à la fois, par sa structure de données, mais tout autant par les manipulations dont il est susceptible d'être l'objet et par les événements qu'ils sont capables de déclencher.

Les concepts utilisés

Objet et classe

Après la période des premières recherches où la terminologie n'était pas encore figée, il semble que l'on s'achemine vers un consensus sur la définition des mots.

On définit un *objet* comme « une collection d'éléments de données structurées identifiée par une référence unique » (GAR90), alors qu'une *classe* représente « un groupe d'objets ayant les mêmes propriétés, caractérisé par une collection d'opérations qui s'appliquent aux objets de la classe en cachant la structure ».

Dans une approche du type de celle choisie par O2, on déclarerait une classe de la façon suivante :

```
CLASSE      PERSONNE
NOM :       STRING
PHOTO :     BITMAP
ÂGE :       INTEGER
ADRESSE :   STRING
ENFANTS :   SET OF (PERSONNE)
```

Agrégation et généralisation

Les approches objets utilisent les concepts d'agrégation et de généralisation que nous avons présentés lors de l'examen des réseaux sémantiques.

La *généralisation* se traduira par la définition :

CLASSE PERSONNE EST UN HUMAIN
(où HUMAIN est une classe déjà définie).

Il est également possible dans l'approche objet de modéliser les liens sémantiques entre les éléments. La plupart des systèmes retiennent trois types de lien :

- le lien « est un »,
- le lien « est composant de »,
- le lien « est instance de ».

Encapsulation

Toutes les approches objets utilisent un concept que l'on appelle « *l'encapsulation* ». Les techniques d'encapsulation sont utilisées depuis bien longtemps en programmation parce qu'il est nécessaire de diviser les programmes en des parties plus petites qui sont « encapsulées » dans les programmes : ceux-ci sont alors plus facilement développés et maintenus.

L'encapsulation permet donc de ne considérer que les entrées et les sorties d'un processus, sans avoir à considérer ce qu'il y a dedans.

Approche de la dynamique

Une *méthode* représente une opération associée à une classe qui manipule ou retourne l'état d'un objet ou d'une partie d'on objet de la classe. Par exemple :

Méthode : changement d'adresse (adresse : adresse) : personne

On appelle *signature* d'une méthode sa vue externe, c'est-à-dire son nom, la classe sur laquelle elle intervient, le type des arguments (ou de la classe) utilisé et, en général, le résultat. Pour l'utilisateur, seule la signature a un intérêt. Les méthodes peuvent être publiques (on peut les voir de toutes les classes et tout le monde peut les utiliser). Elles peuvent être privées, c'est-à-dire visibles seulement à l'intérieur de leur classe. Ces méthodes sont donc spécifiques à leur classe.

Objets actifs

Un *objet actif* est un objet qui possède un large degré de responsabilité et d'autonomie. Dans la plupart des systèmes développés à ce jour, les objets doivent recevoir un message avant d'entreprendre une quelconque action : ce sont en fait des objets passifs. Dans quelques cas, les objets peuvent eux-mêmes être à l'origine d'action sans pour autant avoir au préalable reçu un message : ils portent en eux la possibilité de déclencher l'action. On parle alors d'objets actifs.

L'un des premiers systèmes implantés possédant des objets actifs est ARGUS (LIS82). Il représente les objets actifs comme des « guardians ».

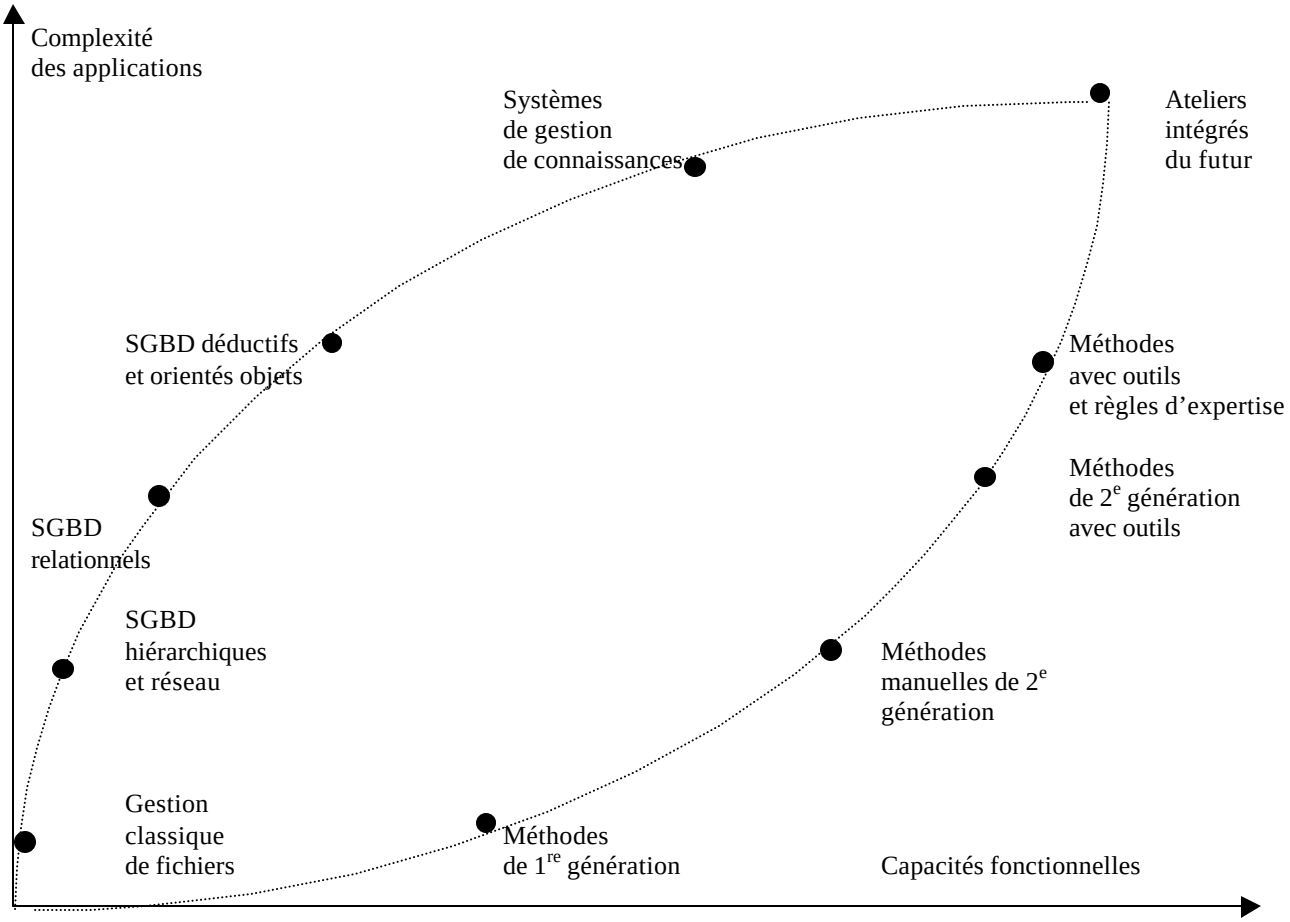
Un « guardian » fournit un ensemble d'opérations qui peuvent être déclenchées par d'autres « guardians ». ARGUS fournit également un mécanisme pour la synchronisation des processus.

Un autre système, appelé KNOS (TS187) développé à l'Université de Genève, utilise un état des données internes, un ensemble d'opérations et un ensemble de règles. KNOS a pour cible les applications dont les objets doivent adapter ou apprendre leur comportement.

Problèmes liés à la conception

Lorsque l'on parle de l'approche objet dans la conception, il est nécessaire de bien différencier deux démarches :

- l'une, qui utilise une approche traditionnelle (c'est-à-dire, sans retenir les concepts objets dans la méthode elle-même) pour construire une base de données qui, elle, est orientée objet;
- la deuxième, qui utilise les apports de l'approche objet dans le processus même de conception; dans ce dernier cas, on imagine que cette approche pourrait aussi bien être utilisée pour construire une base objet que pour une base relationnelle.



La première démarche, est plus simple, en l'état actuel des connaissances et l'on assiste aujourd'hui à la création de projets allant dans ce sens. On parle, par exemple, d'une méthode MERISE pour les bases objets.

On peut noter à ce jour, une approche proposée par CAUVET *et al*. (CAU84) d'une méthodologie ciblée sur les bases objets.

La réflexion sur la deuxième approche ne fait que commencer (voir dans ce sens l'article de Flory et Rochfeld (FLO89).) Savoir comment utiliser les concepts objets dans la méthodologie elle-même est un problème ouvert. On peut imaginer, par exemple, de considérer les phases d'une méthode comme autant d'objets qui se transmettent des messages. On voit alors que le gestionnaire d'objets d'un SGBD objet, pourrait être lui-même utilisé pour concevoir les applications qu'il va gérer. On pourra alors qualifier de tels systèmes d'outils intelligents.

3.3 Les outils intelligents

Les outils actuels assistent le concepteur dans la gestion des spécifications qu'il élabore au fur et à mesure qu'il progresse dans la conception du système. Les outils de demain devraient être plus actifs, plus « intelligents » et apporter une aide véritable et efficace pendant le processus de conception.

La nouvelle génération d'outils relève d'une approche intelligence artificielle/génie logiciel (Arango 1987) qui se caractérise par une re-formulation du processus de conception dans les termes de l'intelligence artificielle et une tentative de résolution des problèmes qui y sont rencontrés, fondée intégralement sur les techniques de l'I.A.

Les techniques les plus importantes de ce point de vue sont :

- le raisonnement autour de spécifications qui peut mettre en évidence des propriétés indésirables des spécifications telles que l'ambiguïté, la contradiction ou la redondance;
- l'utilisation de connaissances heuristiques;
- le raisonnement flou permettant de bâtir des spécifications à partir d'énoncés incomplets;
- l'apprentissage qui permet à l'outil d'exploiter automatiquement sa connaissance d'un domaine applicatif donné.

L'utilisation de ces techniques doit permettre de construire des outils CASE qui :

- prennent en compte des spécifications; les outils doivent, à la fois, palier l'incomplétude et aider le concepteur à la gérer;
- règlent le problème des spécifications incohérentes, c'est-à-dire qui soient capables de détecter les incohérences et de guider le concepteur dans leur résolution;
- soient capables de réutiliser la connaissance acquise au cours de projets de conception précédents; la conception n'est pas un cas strict de résolution de problème et la résolution est toujours présente implicitement ou explicitement dans l'activité de spécification;
- expliquent au concepteur le pourquoi d'un certain résultat et justifient leurs décisions.

On commence successivement trois des aspects de la conception où les techniques de l'I.A. peuvent avoir un impact. Ce sont :

- l'automatisation du processus de conception,
- la réutilisation de connaissances liées au domaine,
- l'explication et le tutorage.

L'automatisation du processus de conception

Le processus de conception est aujourd'hui conduit par l'homme. Il est long, itératif et souvent créatif. L'activité de conception est caractérisée par un certain indéterminisme :

- dans la définition du problème lui-même, les limites du domaine d'application ne sont pas clairement définies; les objectifs du S.I. sont souvent flous;
- dans la manière de choisir un schéma conceptuel, car plusieurs sont possibles pour le même domaine d'application;
- dans le passage du schéma conceptuel au schéma physique.

L'expert humain maîtrise l'activité de conception car il conjugue sa connaissance des techniques à un savoir expérimental. Il connaît et maîtrise les concepts et règles des modèles avec lesquels il raisonne.

Mais complémentairement, il utilise son expérience pour reconnaître des cas typiques, résoudre des problèmes par analogie, être vigilant à certaines étapes du processus qu'il sait délicates. Par exemple, un expert relationnel n'utilise pas la normalisation de la même manière qu'un débutant : il sait d'emblée se placer au niveau de la 3^{e} forme normale, mais prendre quelques précautions dans l'emploi des multivaluées et des dépendances de jointure.

Prenant en compte ces deux aspects (l'indétermination du processus et le comportement des concepteurs), il ressort que l'automatisation de la conception ne peut être algorithmique. Elle doit être fondée sur la *formalisation de la connaissance formelle et expérimentale* nécessaire à la maîtrise de l'activité de conception. L'approche système expert semble adaptée et des prototypes d'outils tels que SECSI (Bouzeghoub 85) et OICSI (Rolland 86, Cauvet 88) en sont une illustration.

Dans un tel système expert, la *base de connaissances* représente la formalisation des connaissances formelles et heuristiques de conception, tandis que la *base de faits* mémorise les spécifications du S.I. à différents niveaux d'abstraction.

Il peut y avoir plusieurs attitudes de construction de la base de connaissances. Une typologie des activités de conception peut servir de support à la formalisation : reconnaître, par exemple, des activités d'*abstraction* permettant de passer de « besoins » exprimés en langue naturelle à un schéma de nature conceptuelle.

Les solutions proposées aujourd'hui ne sont que des ébauches d'une approche qu'il semble, en revanche, important de poursuivre et d'approfondir.

La vérification de connaissances liées au domaine

Il est bien connu dans le monde professionnel que les concepteurs deviennent experts dans un domaine applicatif donné par leur expérience. Leur expertise résulte d'une accumulation de connaissances acquises au cours du déroulement de projets de développement de S.I. liés à ce domaine. La réutilisation des connaissances acquises leur permet d'être plus efficaces dans la conception d'un nouveau système de cette même catégorie.

Partant de cette considération, l'idée de développer un outil qui apprenne au fur et à mesure de ses expérimentations, utilise ce qu'il a appris pour un nouveau projet et s'en souvienne pour de futurs projets, vient naturellement à l'esprit.

L'intelligence artificielle apporte :

- des techniques de représentation des connaissances pour construire des bases de connaissances pragmatiques;
- des techniques d'apprentissage utiles pour mettre en œuvre la démarche de réutilisation.

Les techniques de représentation de connaissances sont les règles de production (Prost 1943), (Pinson 1981), (Boley 1983), les réseaux sémantiques (Quilan 1968) et les « frames » (Winston 1975).

Les techniques d'apprentissage ont jusqu'ici été peu utilisées dans le monde des outils d'aide à la conception de systèmes d'information. Celles qui paraissent le mieux adaptées à la nature des problèmes à résoudre sont celles de l'apprentissage par l'exemple (Quilen 1967), (Michaloky 1983) et les techniques d'apprentissage par observation et découverte (Langly 1983).

L'explication et le tutorage

Les outils actuels ont au mieux une touche HELP qui ne peut fournir que des explications ponctuelles et locales au concepteur en désaccord avec le comportement de l'outil. En réalité, les outils n'ayant pas intégré la démarche méthodologique qu'ils supportent, c'est au concepteur d'acquérir la démarche pour être en mesure de se servir correctement de l'outil.

L'efficacité des concepteurs et la qualité du travail de conception seraient sans nul doute accrus par l'existence dans un outil CASE :

- d'un module d'explication capable d'expliquer au concepteur pourquoi un certain résultat de conception a été atteint;
- d'un tuteur qui enseigne au concepteur l'emploi des formalismes, des modèles, des langages, des démarches et des techniques utilisés par l'outil.

Les spécialistes de l'intelligence artificielle ont, dans ce domaine, une grande expérience dont il peut être intéressant de tirer parti.

Les travaux de Clancey (1983), Sleeman (1982), Nicaud (1988) proposent une architecture de tutoriel à quatre composantes :

- le tuteur,
- le module expert,
- la base de connaissances,
- l'interface.

Cette architecture semble adaptée et facilement intégrable dans un système expert d'aide à la conception.

Les techniques d'explication telles que les « textes imbriqués » de Fox (1970) et Simmon (1970), et les traces, Shortliffe (1976), Clancey (1981), Sartaout (1983), semblent transportables au domaine des outils d'aide à la conception surtout s'ils sont construits selon les principes de l'architecture d'un système expert d'aide.

4. CONCLUSION

Nous venons de présenter dans ce papier l'état existant des systèmes d'information d'aujourd'hui, ainsi que les perspectives d'ouverture qu'offrent les recherches actuelles.

Les nouvelles approches, en particulier les approches objets et les nouvelles technologies, laissent entrevoir des bouleversements considérables, tant du point de vue de la conception que de celui de la gestion des systèmes d'information.

La décennie 90 va voir entrer le domaine des systèmes d'information dans une ère nouvelle par le recours systématique à des « outils intelligents ». On entre aussi, petit à petit, dans la troisième génération des méthodologies.

BIBLIOGRAPHIE

[ALL81] Allen, J.F. (1981) : An Initial-based Representation of Temporal Knowledge, 7th Int. Point Conf. On Artificial Intelligence, Vancouver, Canada, 1981

[ALL83] Allen, J.F. (1983) : Maintaining Knowledge about Temporal Intervals, *Communication ACM*, Vol 26, No 11, 1983

[ALL82] Allen, J.F.(1984) : Towards a General Theory of Action and Time, Artificial Intelligence, Vol. 23, No 2, 1984

[AND82] Anderson, T.L. (1982) : Modeling Time at the Conceptual Level, *Improving Database Usability and Responsiveness*, P. Schewrmann (ed), Academic Press, 1982

[ARA87] Arango, G., Baxter, I., Freeman, P. (1987) : A Framework for Incremental Progress, the Application of AI to Software Engineering, Research Report, Departement of Information and Computer Science, University of California, Irvine, USA, May 1987

[ASS89] Van Assche, F., *et al.* (1989) : RUBRIC, Esprit Project 928, Final Report, 1989

[BAN87] Banerjee et all « Semantics and implementation of schema evolution in O.O databases » A.C.M. SIGMOD – San Francisco – 1987

[BAN88] Bancilhon F *et al.* « The design and implementations of 02 an 00 DB System » Workshop on OODBS – Badmunster W. Germany 1988

[BEE78] Beeri, C., Berstein, P.A., Goodman, N. (1978) : A sophisticate's Introduction to Normalization Theory, 4th Int. Conf. on VLDB, Berlin, 1978

[BER76] Bernstein, P.A. (1976) : Synthetising third normal Form Relations from functional Dependencies, *ACM TODS*, Vol 1, No 4, 1976
Boley, H. (1983) : Artificial Intelligence Languages and Machines, TSI *Technique et Science Informatique*, Vol. 2, No 3, 1983

[BOL83] Bolour, A., Dekeyser, L.J. (1983) : Abstractions in Temporal Information, *Information Systems*, Vol 8, No 1, 1983

[BOR85] Borgiba, A., Greenspan, S., Mylopoulos, J. (1985) : Knowledge Representation as the Basis for Requirements Specifications, IEEE Computer, 1985

[BOU56] Boulding, K. (1956) : General Systems Theory, the Skeleton of Science, Management Science, 1956

[BOU85] Bouzeghoub, M., Gardarin, G., Metais, E. (1985) : Database Design Tool : an Expert System Approach, Proc. of the 11th VLDB Conference, Stockolm, August 1985

[BRE89] Bretl R. *et al.* – The GemStone Data management system in (KIM 89)

[BRO82] Brodie, M.L., Silva, E. (1982) : Active and Passive Component Modelling : ACM/PCM in Olle (1982)

[BRO86] Brodie, M.L., Mylopoulos, J., Schmidt, J.W. (eds), (1986) : *On Conceptual Modelling*, Springer-Verlag, New York Berlin Heidelberg Tokyo, 1986

[CAU88] Cauvet, C., Rolland, C., Proix, C. (1988) : Information Systems Design : an Expert System Approach in Proc. of the Int. Conf. on Extending Database Technology, Venice, March 1988

[CAU89] Cauvet C. Rolland C. : O* un modèle pour la conception de bases de données orientées objet. Congrès INFORSID – Nancy 1989.

[CHE76] Chen, P. (1976) : The Entity-Relationship Model : Toward a Unified View of Data, *ACM TODS,* Vol 11, No 1, 1976

[CLA81] Clancey, J.W. (1981) : NEOMYCIN : reconfiguring a rule based expert system for application to teaching, in Proc. IJCAI 1981

[CLA83] Clancey, J.W. (1983a) : The Epistemology of a Rule Based Expert System : A Framework for Explanation, *Artificial Intelligence,* No 20, 1983

[CLA83] Clancey, J.W. (1983b) : the GUIDON system, in *int. Journal of computer based instruction* 1983

[CLI87] Clifford, J., Rau, A. (1987) : A Simple General Structure for Temporal Domains, *Temporal Aspects in Informations Systems*, Rolland (ed), North Holland, 1987

[COD70] Codd, E.F. (1970) : A Relational Model of Data for Large Shared Data Banks, *Communication ACM,* Vol 13, No 6, 1970

[COD72] Codd, E.F. (1972) : Further Normalization of the Data Base. Relational Model in Database Systems, Prentice-Hall, Englewood Cliffs, 1972

[DIG89] Dignum, F. (1989) : A Language for Modelling Knowledge Bases, PhD Thesis, Department of Mathematics and Computer Science, Vriji Universiteit, 1989

[DUB86] Dubois, E., Hagelstein, J., Lahou, E., Ponsart, F., Rifault, A. (1986) : A Knowledge Representation Language for Requirements Engineering, Proc. of the IEEE, Vol 14, No 10, 1986

[DUB87] Dubois, E., Hagelstein, J. (1987) : Reasoning on Formal Requirements : A Lift Control System, 4th Int. Workshop on Software Specification and Design, Monterey, USA, 1987

[FAG77] Fagin, R. (1977) : Multivalued Dependencies and a New Normal Form for Relational Databases, *ACM TODS,* Vol 2, No 3, 1977

[FIS89] Fishman D.H. *et al.* – An overview of the IRIS DBMS in (KIM 89)

[FLO86] Floyd, C. (1986) : A Comparative Evaluation of System development methods in Olle 1986

[FLO89] Flory A., Morejon J., Rochfeld A. : « Toward a new generation of design tools : some basic ideas. First int conf. on TOOLS – Paris – 1989.

[FOX70] Fox, A.J. (1970) : Survey of Question Answering Systems, in *Medical Computing American Elsevier*, New York, 1970

[GAZ90] Gardarin G., Valduriez P. : SGBD avancés – Éditions Eyrolles – Paris 1990

[GRI82] Van Griethuysen (ed) : Concepts and Terminology for the Conceptual Schema and the Information Base, ISO/TC97/SC5-N695, 1982

[GRA88] GRAPHAEL S.A. « G base 3.2. manual » sun version. 1988. University de Compiègne

[GUS82] Gustafsson, M.R., Karlsson, T., Bubenko, J.A. (1982) : A Declarative Approach to Conceptual Information Modeling in Olle (1982)

[HAM81] Hammer, M., McLeod, D. (1981) : Database Description with SDM : A Semantic Model, *ACM TODS*, Vol 6, No. 3, 1981

[HAR84] Harel, D. (1984) : Dynamic Logic in D.M. Gabbay, F. Guenthu (ed), *Handbook of Philosophical Logic,* Vol 2, Reidel, 1984

[JAC83] Jackson, M.A. (1983) : *System development*,Prentice Hall International, 1983

[KOW79] Kowalski, R.A. (1979) : *Logic for Problem Solving*, North Holland, 1979

[LAN83] Langley, P., Bradshaw, G.L., Simon H.A. (1983) : Rediscovering chemistry with the Bacon system, chapter in *Machine Learning : An artificial Intelligence approach* Michalski R.S. (eds) Tioga Publishing company 1983

[LEM77] Lemoigne J.L. *La théorie du système général-théorème de la modélisation* PUF Paris 1977

[LON86] Longworth, G., Nicholls, D. (1986) : SSADM Manual, NCC Publications, Manchester, UK, 1986

[LIS82] Liskou, Scheifler : Gardians and actions : « Linguistic support for robust, distributeol systems » ACM tran. On Prog. Lang – Vol 5 no 3 pp. 381-404 - 1982

[LUN82] Lundeberg, M. (1982) : The ISAC Approach to Specification of Information Systems in Olle (1982)

[MAR78] De Marco (1978) : *Structured analysis and system specification,* Yourdon Press, NY, 1978

[MAR86] Martin, J. (1986) : Information Engineering : An improved, automatable Methodology for the Design of Data sharing Systems, in Olle 1986

[MCA79] McArthur, J. (1979) : *Tense Logic*, Reidel, 1979
J. McCarthy, P. Hayes : Some Philosophical Problems from the Standpoint of Artificial Intelligence, *Machine Intelligence,* Vol Meltzer and Michie (eds), University Press, Edenburgh, 1969

[MCD82] McDermot, D. (1982) : A Temporal Logic for Reasoning about Processes and Plans, *Cognitive Science*, Vol 6, 1982

[MEY85] Meyer, B. (1985) : On Formalism in Specifications, IEEE Software, 1985

[MEY78] Meyers, G.J. (1978) : Composite/structured Design, New York : Van Nostrand, 1978

[MIC83] Michalski, R.S., Hoof, N., Stepp. R.E. (1983) : « INDUCE 2 : A Program for Learning Structural Destruction from Examples », Technical Report, Department of Computer Science, University of Illinois, 1983

[MIL80] Mylopoulos, J., Yang, D., Fry, J.P. (1980) : A Language Facility for Designing Interactive Database-Intensive Applications, *ACM TODS*, Vol 5, No 2, 1980

[NICA87] Nicaud, J.F., Vivet, M. (1987) : Les tuteurs intelligents : réalisation et tendances de recherche, *TSI* Vol 7, 1987

[OLI82] Olive, A (1982) : DADES : A Methodology for Specification and Design of Information Systems, in Olle (1982)

[OLI86] Olive, A. (1986) : A Comparison of the Operational and Deductive Approaches to Conceptual Information Systems Modelling, Information Processing 86, Elsevier Science Pub, North Holland 1986

[OLI89] Olive, A. (1989) : On the Design and Implementation of Information Systems from deductive Conceptual Models, 15th Int. Conf. on VLDB, Amsterdam, 1989

[OLL82] Olle, T.W., Sol, H.G., Verryn-Stuart, A.A. (eds) (1982) : *Information Systems Design Methodologies : A comparative Review*, North Holland (pub), 1982

[OLL86] Olle, T.W., Sol, H.G., Verryn-Stuart, A.A. (eds) (1986) : *Information Systems Design Methodologies : Improving the Practice*, North Holland (pub), 1986

[PAR72] Parnas, D.L. (1972) : On the Criteria to be used in decomposing Systems into Modules, *Communication ACM*, Vol 15, No 12, 1972

[PIN81] Pinson, S. (1981) : Représentation des connaissances dans les systèmes experts, *RAIRO Informatique*, Vol 15, No 14, 1981

[POS43] Post, E. (1943) : Formal Reductions of the General Combinatorial Problem, *American Journal of Mathematics*, 65, 1943

[QUI67] Quilan, R. (1967) : Word Concepts : A Theory and Simulation of some Basic Semantic Capabilities, in *Behavioral Science*, Vol 12, pp 410-430, 1967

[QUI68] Quilan, R. (1968) : Semantic Memory, Semantic Information Processing, M. Minsky (ed) MIT Press, Cambridge Mass., 1968

[RES71] Rescher, N., Viquhart, A. (1971) : Temporal Logic, Springer Verlag, New York, 1971

[ROL82] Rolland, C., Richard, C. (1982) : The Remora Methodology for Information Systems Design and Management in Olle (1982)

[ROL86] Rolland, C., Proix, C. (1986) : « An Expert System Approach to Information System Design in IFIP World Congress 86, Dublin, 1986

[ROS77A] Ross, D.T., Scholman, U.E. (1977) : Structured Analysis for Requirements Definition, *IEEE Trans. Software Eng.* Vol SE-3, No 1, 1977

[ROS77B] Ross, D.T. (1977) : Structured Analysis (SA) : A Language for communicating Ideas, *IEEE Trans. Software Eng.* Vol SE-3, no 1, 1977

[ROS85] Ross, D.T. (1985) : Applications and Extensions of SADT, Computer Vol 18, No 4, 1985

[SCH83] Schiel, U. (1983) : An Abstract Introduction to the Temporal Hierarchical Model (THM) 9th VLDB Int. Conference, Florence, 1983

[SER89] Sernadas, A., Fiedero, J., Sernadas, C., Ehrich, H.D. (1989) : The Basic Building Block of Information Systems, Information Systems Concepts : An In-depth Analysis, E. Falkenberg, P. Lindgreen (eds), North Holland, 1989

[SHO76] Shortliffe, E. (1976) : Computer based medical consultation MYCIN, Elsevier 1976

[SIM70] Simmon, R.F. (1970) : Natural Language Question Answering Systems in *Communication ACM*, No 13, pp 15-30, 1970

[SLE82] Sleeman, B.D., Brown, J.S. (1982) : *Intelligence Tutoring Systems,* Academic Press, 1982

[SMI87] Smith, J.M., Smith, D.C.P. (1977) : Data Base Abstractions : Aggregation and Generalization, *ACM TODS*, Vol 2, No 2, 1977

[SNO86] Snodgrass, R. (1986) : Research Concerning Time in Databases, ACM Sigmod Record, Vol 15, No 4, 1986

[STA76] Stay, J.F. (1976) : HIPO and integrated Program Design, *IBM Syst. Journal*, Vol 15, No 2, 1976

[SWA83] Swartout, W.R. (1983) : X PLAIN : a system for creating and explaining expert consulting programs, *AI* Vol 21 No 3, pp 285-325, Sept. 1983

[TEI77] Teichroew, D., Hershey, E. (1977) : PSL/PSA : A Computer Aided Technique for Structuring Documentation and Analysis of Information Processing Systems, IEEE Trans. Software Eng., Vol 13, 1977

[TEM89] TEMPORA, Esprit2 Project, Concepts Manual, 1989
Vilain, M.B. : A System for Reasoning about Time, Proc. AAAI 82, Pittsburg, USA, 1982

[WIN75] Winston, P.H. (1975) : the Psychology of Computer Vision, PH Winston Editor Mac Graw Hill Book Company, 1975

[WIR71] Wirth, N. (1971) : Program Development by Stepwise Refinement, *Communication ACM.* Vol 1, No 4, 1971

[YOU79] Yourdon, E., Constantine, E. (1979) : Structured Design, Englewood Cliffs, Prentice-Hall, 1979

[TSI87] Tsichritzis *et al.* Kno's Knowledge Acquisition, dissemination and manipulation objets. – ACM Tram on office I?S. Vol 5 No 4 pp 96-112 – 1987