Jean-Pierre GOULETTE

Laboratoire d'Informatique Appliquée
à l'Architecture,
Ecole d'Architecture de Toulouse

Menon : un système expert orienté objet pour l'analyse de scènes en architecture

Résumé

Nous présentons tout d'abord la première version d'un système d'analyse de scènes en architecture néo-classique : Menon. Ce logiciel s'inspire des mécanismes de reconnaissance des formes issus des techniques de résolution de l'intelligence artificielle et se propose de réduire la complexité de communication homme-machine dans les logiciels de CAO.

Nous décrivons ensuite l'évolution du logiciel : une seconde version s'intéresse à la mise au point d'un moteur d'inférence permettant une interaction souple entre l'utilisateur et les mécanismes déductifs (un moteur d'inférence de type "Prolog" sera retenu).

Enfin, une troisième version nous permet de décrire une possibilité de structuration des données à partir du concept de "frame". Cette "orientation objet" vient alors colorer l'ensemble du logiciel où la connaissance (symbolisée par des règles de production) et les mécanismes exploitant cette connaissance sont distribués, à travers les objets, vers des lieux pertinents.

Abstract

At first, we present the first version of a software wich analyses scenes in architecture : Menon. This software draws its inspiration from mechanisms wich identify forms sprung from technics of resolution for the artificial intelligence and, purposes to reduce the complexity of the communication between man and machine in the C.A.A.D. softwares.

Then, we describe the evolution of the software : a second version has an interest in the tuning of an engine with inference wich allows a flexible interaction between the user and the deductive mechanisms (a "Prlog" like engine will be choiced).

At last, a third version allows us to describe a possibility of structuring the data from the concept of "frame". So, this "object orientation" comes to colour the whole software where the knowledge (symbolized by production rules) and the mechanisms exploiting this knowledge are distributed, through the objects, towards pertinent places.

1 Introduction

Nos précedents travaux de recherche nous ont amenés à nous intéresser plus particulièrement aux possibilités d'assistance informatique dans les premiers instants d'une démarche de conception en architecture. Notre souci n'est donc pas d'étudier l'assistance à l'instrumentation du projet, mais bien plutôt de poser les bases d'un système souple et non contraignant pouvant être utilisé par l'architecte dès les premières esquisses du projet architectural. Au stade de l'esquisse, le concepteur manipule en effet des éléments graphiques imprécisement dessinés : son but n'est pas alors de décrire précisement le projet, mais seulement "d'esquisser" rapidement sa morphologie globale à partir de signifiants malléables (il s'agit moins là de représenter sans ambiguïté le bâtiment que d'imprimer les différentes traces d'une pensée conceptuelle). Ainsi un même élément d'architecture peut avoir, au fur et à mesure de l'avancement du projet, diverses représentations pour finalement être dessiné selon les normes du rendu traditionnel d'architecture. Il devient alors nécessaire que les premiers dessins de l'acte de conception puissent être réalisés facilement et rapidement sans aucune contrainte quant à la précision des tracés et des signifiants manipulés.

Nous avons donc orienté nos recherches vers la définition d'un système permettant au concepteur d'utiliser plusieurs représentations (en particulier des représentations floues et rapidement exécutées) pour un même élément. Ce système devra posseder la faculté de reconnaître un élément à travers ses divers signifiants (par le biais d'informations de type contextuel et non iconographique), et de lui associer les attributs correspondants. Un tel objectif nous a amenés à l'étude des mécanismes de reconnaissance des formes issus des techniques de résolution de l'intelligence artificielle. Un protoype d'un tel système à été mis au point au laboratoire LI2A sous la forme d'un "mini système expert" visant à réduire la complexité des problèmes de communication homme-machine dans les logiciels de CAO en architecture.

2 Première version du logiciel Menon : la formalisation des règles

Menon opère dans un cadre très particulier : les façades des porches de l'architecture néo-classique française telles qu'elles ont été définies par J.N.L. Durand dans son traité du début du XIX^{e} siècle. Les éléments d'architecture rentrant dans la composition d'un porche sont : les colonnes, le ou les entablements (dans le cas de deux rangées de colonnes superposées), le fronton, le tympan, la statue (posée au sommet du fronton), les croisées, les portes et, éventuellement le piedestal (les colonnes du porche pouvant reposer directement sur le socle de l'édifice). Un porche est donc défini par combinaison de ces éléments (la combinaison minimum regroupant deux colonnes et un entablement).

Nous avons choisi d'orienter tout d'abord le processus d'analyse de scènes vers la reconnaissance des éléments porteurs et des éléments portés (que nous nommerons éléments soutenus dans la suite de ce texte). La première étape de l'analyse sera donc d'établir (et de mémoriser) les relations porteur et soutenu entre tous les éléments.

Nous définissons avant tout une méta-règle générale : une règle ne s'applique qu'aux éléments non encore reconnus. Ainsi, les éléments totalement identifiés étant soustraits au mécanisme d'inférence, certaines règles pourront procéder par élimination.

Voici l'ensemble des règles utilisées :

Règle 1
Si l'élément étudié est dessous un autre élément et
est en contact avec cet élément
Alors l'élément étudié est porteur de cet élément.

Règle 2
Si l'élément étudié est dessus un autre élément et
est en contact avec cet élément
Alors l'élément étudié est soutenu par cet élément.

(La pertinence de ces deux règles est bien entendu limitée au cas précis qui nous intéresse... Nous admettrons, pour la suite du processus, que ces relations constituent une base de données propre à chaque élément et disponible pour la définition d'autres règles.)

L'élément colonne est (parmi l'ensemble des éléments définis ci-dessus) le seul qui est à la fois

porteur et "debout" (très allongé dans le sens vertical si l'on préfère). Donc :

Règle 3
Si l'élément étudié est porteur (quelque soit l'élément porté) et
est debout
Alors l'élément étudié est une colonne

Il nous semble maintenant naturel de définir les règles relatives aux deux éléments qui encadrent verticalement la colonne : le piedestal et l'entablement. L'un porte les colonnes, l'autre est soutenu par elles. Un problème se pose dans le cas d'une superposition de colonnes : l'élément d'articulation entre les deux rangées de colonnes porte bien des colonnes mais n'est pas un piedestal. Nous posons donc deux conditions à la reconnaissance d'un élément piedestal :

Règle 4
Si l'élément étudié est porteur d'au moins une colonne et
n'est pas soutenu par une colonne
Alors l'élément étudié est un piedestal.

Nous pouvons alors définir l'entablement et le fronton, puis le tympan et la statue :

Règle 5
Si l'élément étudié est soutenu par au moins une colonne
Alors l'élément étudié est un entablement.

Règle 6
Si l'élément étudié est soutenu par l'entablement et
couvre tout l'entablement
Alors l'élément étudié est un fronton.

Règle 7
Si l'élément étudié est au-dessus de l'entablement et
est situé dans les limites verticales d'un fronton
Alors l'élément étudié est un tympan.

Règle 8
Si l'élément étudié est soutenu par un fronton
Alors l'élément étudié est une statue.

Une règle intermédiaire nous permet de reconnaître les ouvertures :

Règle 9
Si l'élément étudié est au-dessus d'un piedestal ou
est au-dessous de l'entablement
Alors l'élément étudié est une ouverture.

La porte est alors une ouverture qui "découpe" le mur jusqu'au niveau où reposent les colonnes :

Règle 10
Si l'élément étudié est une ouverture et
est au même niveau inférieur qu'une colonne
Alors l'élément étudié est une Porte.

Les croisées sont alors les ouvertures restantes (les portes sont éliminées par la méta-règle générale) :

Règle 11
Si l'élément étudié est une ouverture
Alors l'élément étudié est une croisée.

Il nous faut maintenant concevoir et réaliser un moteur d'inférence capable d'utiliser l'ensemble des règles que nous venons de définir. Nous avons plusieurs contraintes :

- Le moteur d'inférence doit pouvoir manipuler des variables (ordre 1)
- La conclusion de l'activation d'une règle doit être mémorisée dans une base de données propre à chaque élément (cette conclusion pouvant être utilisée ultérieurement par d'autres règles).
- Les règles doivent être employées dans l'ordre dans lequel elle ont été définies.
- Un élément reconnu devra être ôté de la liste des éléments à reconnaître (certaines règles

procédant par élimination).

De plus il nous faut bien entendu écrire l'ensemble des procédures permettant l'établissement des relations (nous emploierons dans la suite de ce texte le terme "slot" par référence aux langages objets) entre éléments : "dessous", "en contact avec", "dessus", "debout", "couvre tout", "situé dans les limites de", "au même niveau inférieur que". Nous utiliserons, pour écrire notre logiciel, le langage de programmation Tangram que notre laboratoire a développé (interprète Lisp interfaçé avec un editeur graphique 2D ; voir le rapport final de recherche (LI2A-1984)). Celui-ci nous permet en effet de définir et de manipuler des atomes graphiques, et d'obtenir un ensemble d'informations sur la représentation graphique de ces atomes.

Nous regroupons l'ensemble des éléments constituant le porche à étudier dans une liste d'atomes (un atome par élément) possédant chacun une propriété graphique mémorisant le dessin de l'élément. De plus nous attachons à ces atomes une expression fonctionnelle lui permettant de constituer et d'explorer "lui-même" sa propre base de données pour répondre à toute requête concernant celle-ci. Voici le squelette de cette expression :

Si la réponse à la requête est présente dans la base de données de l'élément,
  Alors renvoyer cette réponse
Sinon, Si la requête concerne un slot,
  Alors
    Caculer la réponse (par le biais de la définition fonctionnelle du slot),
    Insérer cette réponse dans la base de données de l'élément (à la propriété "slot"),
    Renvoyer cette réponse
Sinon, renvoyer une liste vide.

Ainsi les slots de chaque élément ne seront calculés qu'au plus une fois. Par exemple, une requête du type (ELEMENT-1 DESSUS) retournera toujours une liste d'éléments situés en dessous de ELEMENT-1 (celui-ci sera donc dessus chaque élément de la liste retournée). De plus les différents slots de chaque élément ne seront calculés que si nécessaire.

Le résultat de l'activation d'une règle sera, au même titre que le résultat d'un calcul de slot, introduit dans la base de données de l'élément. Une requête du type (ELEMENT-1 PORTEUR) pourra donc être formulée par le système (ou l'opérateur) aprés l'activation de la règle 1 et renverra la liste (éventuellement vide) des éléments portés par ELEMENT-1.

Le moteur d'inférence procéde en appliquant chaque règle sur chaque élément non encore reconnu (création de requête pour chaque condition de la règle et soumission de cette requête à chaque élément). L'ordre des règles est bien entendu respecté, le moteur procéde par chaînage avant, en régime irrévocable, dans le cadre d'une logique monotone (le dessin n'évoluant pas pendant l'activation des règles).

Dans cette première version, nous avons exploité le résultat de l'analyse en remplaçant le dessin de chaque élément ébauché par l'utilisateur par le tracé exact de l'élément défini en bibliothèque. Ainsi, l'opérateur dessine tout d'abord une ébauche de son porche, active le mécanisme de reconnaissance des éléments et peut ensuite obtenir un dessin exact de son porche (dessin composé à l'aide des éléments de bibliothèque placés automatiquement sur le dessin à la position et aux dimensions des éléments ébauchés).

Une des utilisations possibles de cet utilitaire pourrait être son intégration dans l'éditeur d'un programme de CAO, cet éditeur devenant ainsi "intelligent".

3 Deuxième version : l'étude du moteur d'inférence

La première version de Menon répond à notre objectif : la reconnaissance des éléments constituant un porche à partir d'une ébauche réalisée par l'utilisateur. Elle posséde tout de même une certaine "rigidité", et ceci pour deux raisons :

- le moteur d'inférence procédant en chainage avant et en largeur d'abord, la totalité des éléments dessinés est obligatoirement étudié par Menon à chaque session. L'utilisateur ne peut donc restreindre le champ d'investigation du logiciel à quelques éléments particuliers (au détriment d'autres jugés inintéressants).
- l'ensemble des règles devant respecter un ordre strict (l'ordre d'utilisation de ces règles par

le moteur d'inférence), l'utilisateur se trouve dans l'impossibilité de définir interactivement une nouvelle règle ou de modifier une règle existant déjà.

Nous nous sommes donc intéressés à la réalisation d'un moteur d'inférence plus souple. En effet, la totalité des éléments reconnaissables (l'ensemble des diagnostics) étant connus à l'avance, un moteur d'inférence du type chaînage arrière semble approprié à la reconnaissance de ces éléments. Nous avons développé, pour le logiciel Menon, un moteur d'inférence de type Prolog procédant en chaînage arrière, avec filtrage des règles, et en régime par tentatives.
Dans un tel système, les règles et les faits sont regroupés dans une base commune : la base de connaissances. Celle-ci comprend donc les faits établis :

```
((DESSOUS OBJ1 OBJ2))
((DEBOUT OBJ1)) ...
```

et les règles :

```
((PORTEUR ?X ?Y)(DESSOUS ?X ?Y)(CONTACTY ?X ?Y))
((COLONNE ?X)(DEBOUT ?X)(PORTEUR ?X ?Y))
((ENTABLEMENT ?X)(PORTEUR ?Y ?X)(COLONNE ?X)) ...
```

Les variables utilisées doivent impérativement commencer par "?". Le système se charge de renommer les variables lors de l'étape d'unification.

Notre logiciel devient alors plus souple, et l'insertion de nouvelles règles (ou de nouveaux faits) dans la base de connaissances peut être faite interactivement par l'utilisateur. De plus, les règles peuvent être exploitées dans un ordre quelconque et des requêtes du type ((ENTABLEMENT ?X)) peuvent être proposées au système qui activera alors une chaîne de déduction sur un ensemble réduit d'éléments (l'utilisation du paramètre cut : "/" permet de limiter le mécanisme d'inférence à la première solution rencontrée.

Toutefois, si le regroupement des connaissances (établies à priori ou déduites) permet une certaine souplesse dans la déclaration des faits et règles, elle complexifie l'étape de restriction et de filtrage de ces faits et règles, et tend à alourdir le système lorsque cette base de connaissance devient importante. Nous allons tenter de remédier à ceci par le biais de l'utilisation d'une structure de données de type "frame".

4 Troisième version : la structuration des données

4.1 Les frames

Nous utilisons ici le concept de "frame" développé par Minsky (Minsky-1975) (LI2A-1984) (LI2A-1985). Nous rappelons la définition de certains termes qui seront utilisés par la suite :

- Les champs

Sorte-de : lien reliant une frame "fille" à sa frame "mère". Il permet, entre autre, l'héritage des propriétés des frames "mères", "grand-mères", etc... Ce lien est unique pour chaque frame.
Spécialisation : lien inverse du lien "Sorte-de". Il met en relation une frame "mère" avec ses frames "filles". Il peut y avoir plusieurs liens de spécialisation pour une même frame (les caractéristiques du lien "Sorte-de" et "Spécialisation" définissent donc une structure arborescente).
Est-un : lien associant un objet à la frame représentant sa classe. Ce lien est unique pour chaque objet.
Instance : il associe une frame à des objets "réalisations de cette frame". Plusieurs liens de type instance peuvent associer une frame à des objets. Nous distinguerons les "Instances directes" (le lien inverse de "Est-un") des "Instances" (tout court), ce dernier terme regroupant les "Instances directes" et les "Instances" des frames "filles" de la frame considérée.
D'autres champs concernant des propriétés plus spécifiques à notre domaine seront définis en temps utile.

- Les aspects

Valeur : caractérise une valeur terminale au niveau d'une frame ou d'un objet.
Référence : caractérise un lien entre objet et frame et inversement. Cet aspect est le seul admis pour les champs "Sorte-de", "Spécialisation", "Est-un", "Instance".
Cardinalité : Si l'aspect "valunique" est associé à un champ, celui-ci ne pourra admettre qu'une valeur unique ; sinon il pourra admettre plusieurs valeurs.

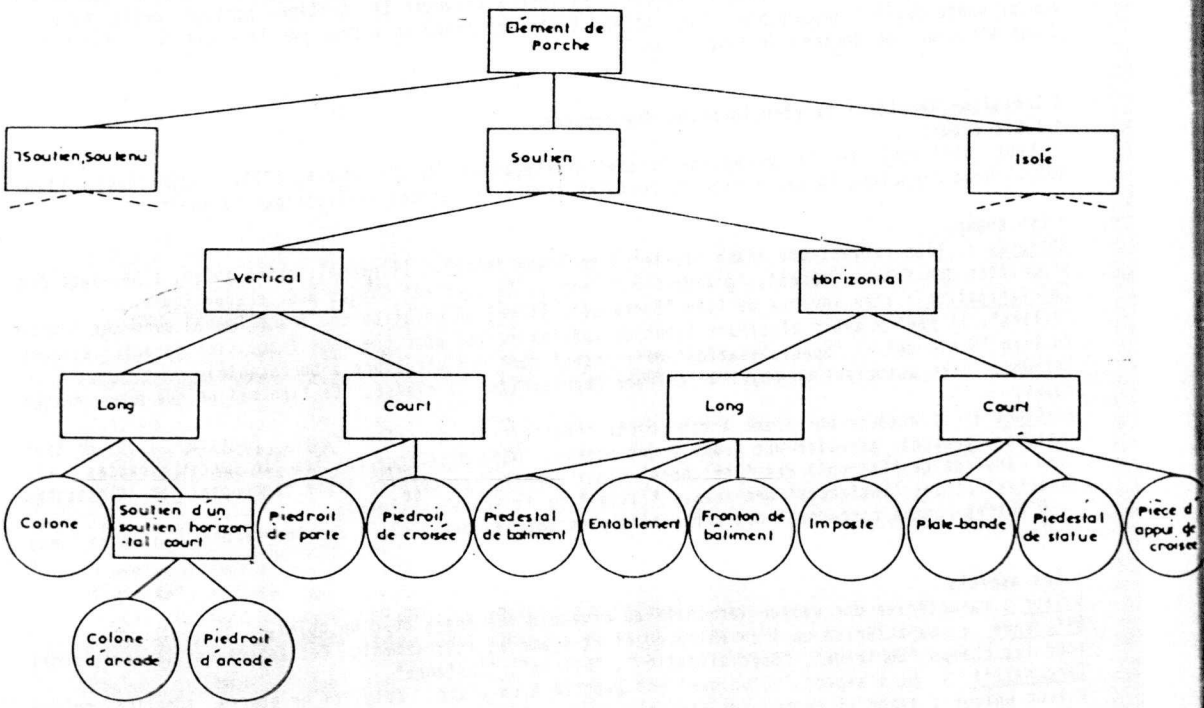
Domaine : donne la liste des valeurs possibles pour un champ.

-Les démons
Si-ajout : active une procédure "parallèle" lorsqu'on ajoute une valeur au champ concerné.
Si-besoin : indique les modalités de calcul pour la détermination de la valeur du champ concerné.

4.2 Le but d'une structuration à base de frames

Pour faciliter la compréhension de ce qui suit, nous indiquons grossièrement le but de notre étude :
soit une arborescence quelconque de frames (bâtie autour du lien "Spécialisation") et
un objet pointant au départ (par son lien "Est-un") sur la racine de l'arborescence ;
il devient alors intéressant de faire progresser (dans le sens d'une "Spécialisation") ce lien "Est-un" dans l'arborescence. L'objet devient alors de plus en plus "spécialisé" au fur et à mesure de sa progression dans l'arbre, pour finalement (dans le cas idéal) être associé à une frame terminale (une feuille de l'arborescence) et donc être totalement reconnu.

(Remarque : dans la phrase ci-dessus, nous avons fait une confusion entre l'objet et sa frame (c'est en effet celle-ci qui devient de plus en plus "spécialisée"). Nous nous permettrons d'utiliser à nouveau cette confusion dans la suite de ce texte pour simplifier des phrases du type "la frame associée à l'objet par le lien Est-un est à un niveau supérieur de spécialisation" par la phrase "l'objet se spécialise".)
Le schéma suivant résume cette démarche :



4.3 Une réalisation du schéma de spécialisation

Le schéma de la page précédente décrit partiellement une réalisation possible de "l'arborescence de spécialisation" appliquée à notre domaine d'étude. Nous insistons sur le fait qu'il s'agit d'une réalisation "possible". Nous avons élargi l'ensemble des éléments de porches étudiés (et donc légérement modifié les règles de définition de ces éléments). Ainsi sont présents dans ce schéma des éléments tels que : piédroit de porte, plate-bande, imposte... Nous n'avons pas fait figurer la totalité des élément "reconnaissables" sur ce schéma ; voici la liste des éléments absents :
- éléments rattachés à la spécialisation "Soutenu" : porte, fronton d'ouverture, archivolte, fronton de batiment, plate-bande, statue ;
- éléments rattachés à la spécialisation "Isolé" : tympan, croisée.

Nous avons fait figurer l'élément croisée sous deux formes :
- l'élément croisée proprement dit (un élément "isolé" sur une façade),
- une décomposition de la croisée en : pièce d'appui, piédroit, plate-bande et optionnellement fronton (si ce fronton est présent sur le dessin, l'élément plate-bande sera orienté vers une spécialisation "Soutien", sinon il sera orienté vers une spécialisation "Soutenu").

Le logiciel peut maintenant s'adapter à divers niveaux de définition d'un élement et l'utilisateur peut ainsi moduler son degré de symbolisation, reflet d'approches conceptuelles à plusieurs niveaux.

4.4 Le fonctionnement du système

Les entités du système :

Les frames

Elles possédent une suite de champs auxquels peuvent être affectés les démons "Si-besoin" et "Si-ajout". Le champ "Spécialisation" comporte des règles sous forme de clauses (exemple : ((SOUTIEN OBJ ?Y)(SOUTHORILONG ?Y)) ; nous verrons la signification de cette clause, ainsi que le rôle des démons associés plus tard). Le champ "Instance" est aussi valué sous forme de clauses (exemple : si la frame considérée est la frame "Soutvertilong" - soutien vertical long - ((SOUTVERTILONG OBJ2)) représente le fait que OBJ2 est une instance de "Soutvertilong").

Les objets

Ils possédent une première méthode leur permettant de valuer les champs de leurs frames "mères" (connus par le champ "Est-un"). Les valeurs de ces champs (obtenues par le biais des fonctions de calcul) sont placées soit dans une base de données numérique, soit dans une base de données organisées sous forme de clauses quand il s'agit de données "de référence" (exemple : ((DESSUS OBJ1 OBJ2)) signifie que OBJ1 est dessus OBJ2). Ces champs sont valués par appel au fonctions de calcul, chaque objet possédent ses propres bases de données (contenant uniquement les informations intéressant cet objet).

Une deuxième méthode propose au moteur d'inférence les règles de spécialisations de la frame "mère". Le moteur d'inférence "applique" ces règles sur la BD de référence de l'objet ; si une unification est possible, l'objet se spécialise en progressant vers la frame "fille" concernée.

Une autre entité du système est le superviseur général. Celui-ci assure la gestion d'une liste d'objets. Au lancement du système, tous les objets dessinés par l'utilisateur sont insérés dans cette liste et le contrôle est donné au superviseur. Celui-ci ôte le premier objet de la liste et "l'active". L'objet applique alors les méthodes 1 et 2 sur lui-même et obtient, ou non, une spécialisation. Si l'objet est spécialisé, il est inséré en queue de la liste, sinon il est mis en "sommeil". Ce processus est réitéré sur chaque objet et s'arrête lorsque la liste est vide.

Nous prendrons l'exemple de la spécialisation d'un objet vers la frame "Colonne" :
L'objet OBJ1 est au départ un élément de porche.
Il est d'abord spécialisé par la règle ((SOUTIEN OBJ1 ?Y)), ?Y référençant un objet quelconque, puis par les règles ((VERTICAL OBJ1)) et ((LONG OBJ1)). L'objet pointe alors vers la frame "Soutvertilong" (ce qui signifie qu'il appartient à la classe des soutiens verticaux longs). Pour progresser vers la frame "Colonne", il lui faut alors vérifier la règle ((SOUTIEN OBJ1 ?Y)(SOUTHORILONG ?Y)), c'est à dire qu'il doit être porteur d'un objet appartenant à la classe des soutiens horizontaux longs (un entablement pour être plus précis). Il se peut qu'à ce stade, aucun objet intéressant notre objet OBJ1, n'ait été spécialisé en "Southorilong". OBJ1 ne peut donc être spécialisé et est donc mis en sommeil.

Mais ce sommeil peut ne pas être éternel... En effet, imaginons un objet OBJ2 progressant dans la "branche des soutiens horizontaux". Il pourra, par le biais de règles de spécialisation, "devenir un Southorilong". Le démon associé à la régle de spécialisation de "Southorilong" réveillera alors la frame mère de OBJ1. Ce réveil consistera simplement à insérer les instances directes de cette frame "mère" (et donc OBJ1) en queue de la liste du superviseur. La règle de spécialisation ((SOUTIEN OBJ1 ?Y)(SOUTHORILONG ?Y)) sera de nouveau appliquée sur OBJ1 qui pourra donc (si l'unification OBJ2 / ?Y est possible) être spécialisé.

Nous présentons sous forme de dialogue le rôle des démons de spécialisation :
OBJ1 - "Je suis un objet possédant une spécialisation Spé1. Je dois, pour pouvoir être spécialisé à nouveau, entretenir une relation RX avec un objet de spécialisation Spé2. Hèlas, pour l'instant, je ne connais aucun objet de spécialisation Spé2 ; je vais donc être mis en sommeil."

OBJ2 - "Je viens d'acquérir une spécialisation Spé2. Je lance donc le démon associé à Spé2."

Le démon (à la frame Spé1) - "Debout là dedans !!" (il insère les instances directes de Spé1 en queue de liste du superviseur.)

OBJ1 (et d'autres) - "Enfin on peut continuer..."

En règle générale, le démon associé à une règle de spécialisation (c'est à dire contrôlant le passage à une frame fille que nous nommerons FF) réveillera toutes les frames utilisant, dans leurs clauses de spécialisations, une référence à FF. Les instances directes de ces frames seront donc insérées dans la liste (si elles ne s'y trouvent pas déjà) et pourront donc être spécialisées sur la base du nouveau fait établi (une nouvelle instance de FF).

Cette possibilité de réveil des objets entre eux nous permet de définir un type d'inférence particulier (que nous avons nommé "inférence coopérative") qui, dès le début de nos recherches nous avait semblé primordial en analyse de scènes en architecture : deux éléments possédant des inter-relations très fortes, sont reconnus à peu près simultanément. Reprenons notre exemple des objets OBJ1 et OBJ2 (et supposons que OBJ1 est une colonne ébauchée par l'utilisateur, et OBJ2 un entablement). OBJ1 est, pour l'instant, un "Soutvertilong" ; pour "devenir Colonne", il lui faut porter un soutien horizontal long. Justement, OBJ2 devient un "Southorilong", il va donc réveiller OBJ1. Mais pour "devenir Entablement", OBJ2 doit être porté par une colonne (règle de la forme ((SOUTENU OBJ2 ?X)(COLONNE ?X))). La spécialisation d'OBJ1 va, à son tour, réveiller OBJ2 (ainsi que tous les objets qui ont "buté" sur une clause faisant référence à Colonne). OBJ2 pourra alors progresser vers la frame "Entablement".

La longueur de la liste gérée par le superviseur peut donc varier à chaque étape. Lorsqu'au cours d'une session (l'activation séquentielle de tous les éléments de la liste), aucun objet n'a été spécialisé, cette liste devient vide (aucun objet n'ayant été réveillé par une spécialisation) et notre système d'analyse de scènes s'arrête. A ce stade, les objets étudiés possèdent une "certaine spécialisation" (qui peut ne pas être terminale).

5 Conclusion

Nous avons développé un logiciel d'analyse de scènes en architecture dont les différentes versions ont abouti à une structure de données particulière. Les caractéristiques des frames utilisées ici sont bien entendu dédiées à notre sujet d'étude : la reconnaissance des éléments architecturaux manipulés par l'utilisateur.

Toutefois, l'intégration d'une telle structure dans un logiciel de CAO ne semble pas devoir rencontrer de difficulté majeure : on peut en effet, sans perturber notre système, élargir l'ensemble des champs, aspects et démons utilisé pour définir d'autres utilitaires d'assistance à la conception. Ainsi, des "attachements procéduraux" permettant la vérification de cohérence, des calculs techniques... peuvent être envisagés. Un tel système possédera alors l'avantage de pouvoir déterminer les fonctions d'assistance pertinentes sur la base d'une "certaine spécialisation" des éléments (on n'activera pas les mêmes procédures de calcul technique pour un soutien horizontal ou un soutien vertical), tout en permettant à l'utilisateur une grande souplesse de tracé.

Le fonctionnement général du système étant indépendant de la nature des règles employées, celles-ci peuvent être redéfinies pour s'appliquer à un autre domaine d'analyse. De plus, les règles étant "dispersées" à l'intérieur du système et uniquement associées aux objets intéressés, le moteur d'inférence peut être soulagé des étapes de restriction et de filtrage : la connaissance est en quelque sorte distribuée vers les lieux où elle se révèle pertinente

Les possibilités d'intégration des mécanismes déductifs, mis en oeuvre par Menon, dans un logiciel de CAO constitueront l'objet de nos recherches futures.

6 Bibliographie

Charniac E. "Artificial Intelligence Programming". Lawrence Erlbraum Associates Publishers, 1980.

Cullen Y. "Expert Systems in Architectural and Planning Education" Proceeding of the eCAADe, BRUSSEL 1983.

Durand J.N.L. "Précis des leçons d'architecture données à l'Ecole Royale Polytechnique". Ed. Verlag Dr. Alfons Uhl, Nördlingen, 1981.

Dugerdil P. Une méthodologie orientée objet pour la représentation des connaissances en C.A.O Architecture G.R.T.C. Marseille 1985

Goulette J. P. "Le dessin d'Architecte et l'Informatique". Travail personnel de fin d'études. Ecole d'Architecture de Toulouse, Novembre 1984

LI2A-83 "Les utilitaires et l'intelligence artificielle pour un système d'aide à la conception en Architecture" Rapport final d'une recherche financée par le Secrétariat de la Recherche Architecturale. LI2A Juin 1983.

LI2A-84 "Les utilitaires et l'intelligence artificielle pour un système d'aide à la conception en Architecture" Rapport final d'une recherche financée par le Secrétariat de la Recherche Architecturale. LI2A Avril 1985.

LI2A-85 "Les utilitaires et l'intelligence artificielle pour un système d'aide à la conception en Architecture" Rapport final d'une recherche financée par le Secrétariat de la Recherche Architecturale. LI2A Avril 1985.

LI2A-84
Caradant D. "Les utilitaires et l'intelligence artificielle pour un système d'aide à la conception en Architecture". Rapport final de recherche. LI2A, Juin 1984.

LI2A-85
J.P. Goulette
P. Pérez "Tangram, Manuel de l'utilisateur", in "Aides Intelligentes au dessin d'Architecte", Rapport final de la Convention AdI 84/935, LI2A, Sept. 85.

Minsky M. "A Framework for Representing Knowledge". in The Psychology of Computer Vision, McGraw-Hill, NY, 1975.

Rich E. A. "Artificial Intelligence". Ed. McGraw-Hill in AI, 1983.

Waterman
Hayes-Roth "Pattern-Directed Inferences Systems". Ed Academic Press, 1977.

Auteur : Jean-Pierre Goulette

Laboratoire d'Informatique Appliquée à l'Architecture,
Ecole d'Architecture de Toulouse
Chemin Aristide Maillol, 31000 Toulouse.
Téléphone : 61 40 47 28 (poste 226).