SHIRKA, 10 ans, c'est TROPES?

Jérôme Euzenat, François Rechenmann

INRIA Rhône-Alpes - IMAG-LIFIA, 46, avenue Félix Viallet, 38031 Grenoble, France
{Jerome.Euzenat,Francois.Rechenmann}@imag.fr

Résumé

Il y a dix ans, apparaissait le système de représentation de connaissance SHIRKA. À travers la présentation de sa conception, de son évolution et de son utilisation, on tente d'établir ce que peut être, dix ans plus tard, un système de représentation de connaissance. La mise en œuvre de deux points clé de SHIRKA — la séparation programmation-représentation et l'utilisation de l'objet partout où cela est possible — est particulièrement étudiée. Ceci permet de considérer leur pertinence et leur évolution pour la représentation de connaissance.

in: Actes 2nde conférence «langages et modèles à objets», Nancy (FR), pp13-34, 1995

10 ans approximativement après son implémentation initiale, le logiciel SHIRKA est toujours utilisé. Avant de nous demander si cela est trop ou trop peu, nous revenons sur ces années de travaux autour de SHIRKA et essayons d'en tirer des enseignements pour le futur et en particulier pour son successeur TROPES.

L'exposé suivant ne correspond pas à ce qui aurait été écrit il y a quelques années: la meilleure compréhension du système et le souci de faire partager cette compréhension là, et non celle de l'époque, interdisent de décrire SHIRKA dans les termes initiaux. Du reste, le manuel [Rechenmann *et al.*, 1988] est toujours disponible.

Le contexte dans lequel SHIRKA a été créé est d'abord présenté. Puis, l'exposé porte sur le noyau initial de SHIRKA, ses extensions ultérieures et ses applications. Après cela les différents aspects conservés ou abandonnés dans TROPES sont abordés.

1. Contexte

La première version de SHIRKA était disponible en 1984 [Rechenmann *et al.*, 1984], c'est-à-dire:

- une quinzaine d'années après le début des langages de programmation à objets, alors que SMALLTALK en était le seul représentant ayant une visibilité importante;
- une dizaine d'années après les écrits sur les "frames" de Marvin Minsky [Minsky, 1974];
- quelques années après les langages de représentation de connaissance FRL [Roberts et Goldstein, 1977], KRL [Bobrow et Winograd, 1977], LOOPS [Bobrow et Stefik, 1981] ou SRL [Wright et Fox, 1983];
- À la suite des premiers articles de Ronald Brachman sur la rationalisation de la représentation de connaissance [Goodwin, 1979; Brachman 1979, 1983, 1985];
- Au même moment que les systèmes commerciaux ART [Williams 1984] et KEE [Fikes et Kehler, 1985] et que les premiers articles sur KL-ONE [Brachman et Schmolze, 1985];
- Au même moment qu'un foisonnement de systèmes développés en France et en Europe: CEYX [Hulot, 1983], MERING [Ferber, 1983], KOOL [Albert, 1984], LRO2 [Roche, 1984], SMECI, YAFOOL [Ducournau et Quinqueton, 1986].

Le contexte semble donc à l'implémentation des résultats des recherches précédentes et à la rationalisation des formalismes de représentation de connaissance. SHIRKA naît de préoccupations de modélisation dans le domaine de la biométrie et de la dynamique des populations. L'idée de créer un système de représentation de connaissance par objets vient de la lecture du chapitre 22 de la première version de [Winston et Horn, 1981] qui décrit les "frames". Il doit aussi beaucoup à [Aikins, 1983] et à l'insistance de Jacques Pitrat et de ses étudiants sur la déclarativité [Laurière, 1982]. En effet, la volonté était dès le début de ne pas mélanger langage de programmation et langage de modélisation. Cette volonté résultait des travaux menés antérieurement sur les langages de modélisation de phénomènes dynamiques, en particulier en économie et en biologie, dans lesquels trop souvent les éléments de description d'un système dynamique se retrouvaient mélangés aux ordres de simulation du modèle mathématique. C'est sans doute à partir de ce parti-pris, souligné par le vocable «centré-objet» destiné à marquer l'opposition avec «orienté-objet», que SHIRKA se démarque des expériences

menées à la même époque. Cet aspect est visible dans la présentation des attachements procéduraux, puis des tâches, où cette séparation est très nette.

La conception et l'évolution de SHIRKA ne se sont cependant pas faits dans l'ignorance des autres efforts menés à la même époque. En particulier, le second parti-pris de SHIRKA, celui de pousser jusqu'au bout le «tout est objet», n'est pas étranger aux travaux concernant les langages de programmation par objets.

La seconde version de SHIRKA, celle distribuée actuellement, a été implémentée en 1986 par François Rechenmann, Jose-Luis Aguirre et Didier Bloch. C'est elle qui est décrite dans la suite. Enfin, pour ce qui concerne l'environnement informatique, SHIRKA été développé en LE-LISP [Chailloux, 1983] sur Apple Lisa, puis porté sur des stations Unix.

2. Le modèle de SHIRKA

Initialement, SHIRKA [Bensaïd *et al.*, 1986; Rechenmann 1985; *et al.*, 1984; 1988; Rechenmann et Uvietta, 1989] tire des "frames" la notion de facette, des langages "post-frames" la rationalisation des facettes et leur utilisation quasi-exclusive à des fins de typage et d'inférence. Il tire aussi des langages de programmation par objets l'idée du «tout objet» et d'une implémentation réflexive où le système lui-même est représenté en terme d'objets [Cointe, 1985]. Mais le langage se démarque de ses semblables de l'époque par l'absence d'envoi de messages et, plus généralement, par un abandon de tout ce qui concerne la programmation pour se concentrer sur la représentation.

2.1. Le noyau de représentation

Le noyau central du système a peu évolué; il est possible de le décrire en trois temps: les objets, les classes et la spécialisation. Un objet est un nom auquel est associé un ensemble de valeurs d'attributs, ces valeurs pouvant être d'autres objets (comme dans SMALLTALK, toutes les valeurs primitives — réels, entiers ou chaînes — sont considérées comme des objets). Le langage permet de décrire un objet de la manière suivante (ι dénote l'absence de valeur, la ligne correspondante n'apparaît alors pas dans le source):

```
{AutoDeSarah
  est-un             =  Auto;
  couleur            =  rouge;
  puissance-fiscale  =  9;
  moteur             =  {Moteur#172
                            est-un    = Moteur;
                            cylindrée = 2000};
  puissance-réelle   =  ι}
```

Une classe décrit un ensemble d'objets et pour ce faire elle énumère la liste de ses attributs ainsi que leur type. Le type d'un attribut est défini au moyen de facettes (voir table 1). Enfin la classe détermine aussi des contraintes que doivent vérifier ses instances.

Ainsi, la classe `Auto` peut-elle se définir comme une sous-classe de `Véhicule` dont les attributs `couleur`, `puissance-fiscale` et `puissance-réelle` sont respectivement réduits à un `symbole` parmi `rouge`, `vert` ou `bleu`, un `entier` entre `1` et `12` et un `réel`:

```
{Auto  est-un Classe
  sorte-de           =           Véhicule;
  couleur            $un         symbole
                     $domaine    rouge vert bleu;
  puissance-fiscale  $un         entier
                     $intervalle [1 12];
  puissance-réelle   $un         réel
                     $sib-exec
                        {calc-puissance
                           auto        $var<- lui-même;
                           puissance   $var-> puissance-réelle}}
```

La spécialisation d'une classe par une autre permet de définir une classe comme un ensemble d'objets dont les éléments ont une structure plus spécifique que ceux de la sur-classe. Cette spécificité s'exprime par un ensemble d'attributs plus grand et par des types plus restreints pour les attributs déjà existants.

facette	valeur	interprétation [$I(d)$]	restriction
`$un, $liste-de`	τ	τ, τ*	τ=entier, chaîne…
	f	$E(f)$, $E(f)$*	f est un filtre
	c	$I(c)$, $I(c)$*	c est une classe
`$domaine`	$v_1 \ldots v_n$	$t \cap \{v_1 \ldots v_n\}$	t est le type initial de l'attribut
`$sauf`	$v_1 \ldots v_n$	$t - \{v_1 \ldots v_n\}$	
`$valeur`	v	$t \cap \{v\}$	
`$card`	n	$\{v \in t; \lvert v\rvert = n\}$	(pour $liste-de)
`$intervalle`	$[n_1\ n_2]$	$\{v \in t; n_1 \leq v \leq n_2\}$	(pour entier ou réel)
`$a-verifier`	p		

TAB. 1 – *Facettes de typage. Une première facette (`$un/$liste-de`) permet de spécifier le type ou la classe de la valeur. Un ensemble de facettes permet de restreindre ce type initial (seconde partie du tableau). La facette `$a-verifier` s'applique à un attribut qui peut être l'attribut `lui-même`, permettant de porter sur l'objet tout entier. La seconde colonne donne le type de valeur autorisée pour la facette; la troisième donne l'effet de la facette sur le domaine de l'attribut (voir ci-dessous). * dénote la séquence.*

2.2. Interprétation

SHIRKA est avant tout un programme et n'a pas été pourvu d'emblée d'une sémantique formelle. Cependant, ses concepteurs ont tenté de n'y utiliser que des concepts dont la signification soit suffisamment stable et raisonnée pour être rapidement intelligible. Une ébauche de traitement formel du langage utilisé permet de justifier le fonctionnement de SHIRKA (dans ses grandes lignes, car quelques points restent rebelles tels que le traitement erratique des

listes). Ainsi, partant d'un domaine du discours D, on peut établir l'ensemble des valeurs possibles d'attributs en y ajoutant les valeurs primitives (entiers, réels, chaînes…) ce qui correspond à l'ensemble D'. En y appliquant les constructeurs disponibles dans le langage (`$un` et `$liste-de`), on obtient le domaine $D''=D' \cup D'^*$ (c'est-à-dire contenant D' et toutes les séquences d'éléments de D').

Les objets s'interprètent comme des éléments du domaine (D), les classes comme des parties de ce domaine, les attributs comme des fonctions partielles d'un élément du domaine vers une valeur et les contraintes (ou prédicats) comme des n-uplets de valeurs (un élément de D''). La fonction d'interprétation I est alors telle que:

pour tout objet o, $I(o) \in D$ $\qquad\qquad (I(o)=I(o') \Rightarrow o=o')$

pour toute classe c, $I(c) \subseteq D$

pour tout attribut f, $I(f):D \rightarrow D'$

pour tout prédicat p d'arité n, $I(p) \subseteq D''^n$ (pour simplifier, on considérera que les prédicats utilisés dans les objets prennent en valeur l'objet lui-même; leur interprétation est donc un sous-ensemble du domaine).

Par ailleurs, à chaque facette d est associé un domaine $I(d) \subseteq D''$ qui correspond à la colonne 3 de la table 1. Les sous-ensembles correspondant aux classes subissent les contraintes suivantes:

pour toute sous-classe c' de c (noté $c' \leq c$), $I(c') \subseteq I(c)$

pour toute instance o de c, $I(o) \in I(c)$

pour toute de facette d d'un attribut f de c, $I(f):I(c) \rightarrow I(d)$

Ainsi, la contrainte portant sur une classe est la suivante:

$$I(c) \subseteq \bigcap_{c \leq c'} I(c') \cap \left[\bigcap_{\substack{f_i \in attributs(c) \\ d_{ij} \in facettes(f_i)}} \left\{ o \in D; I(f_i)(o) \in I(d_{ij}) \right\} \right] \cap \bigcap_{p_i \in contraintes(c)} I(p_i)$$

Le premier terme correspond à ce qui est réalisé en terme opérationnel par le mécanisme d'héritage (l'inclusion de l'interprétation d'une classe dans l'intersection de celle de ses sur-classes), le second à la prise en compte de toutes les restrictions sur les attributs de la classe et le dernier à la restriction aux objets satisfaisant les prédicats (utilisés par les facettes `$a-verifier`). Le caractère particulier de cette présentation par rapport à d'autres présentations plus classiques (telles que celles des logiques terminologiques) provient de la présence de relations d'inclusion au lieu d'égalité entre les termes. Ceci correspond à l'interprétation descriptive (et non définitionnelle) des classes qui ne sont décrites que par des conditions nécessaires à leur appartenance mais pas forcément suffisantes.

SHIRKA travaille donc constamment avec deux termes: $I(c)$ représentant l'interprétation réelle des classes qui ne lui est connue qu'au travers des instances et $E(c)$ qui représente l'extension maximale de cette interprétation, c'est-à-dire le second terme de l'inéquation ci-dessus. Elle se simplifie donc en $I(c) \subseteq E(c)$.

2.3. Mécanismes d'inférence

Une classe décrit aussi, à l'aide de facettes, des moyens permettant d'obtenir la valeur d'un attribut lorsqu'elle n'est pas connue (voir table 2).

facette	valeur
`$sib-exec`	méthode
`$sib-filtre`	filtre
`$var<-` (`$var-liste<-`)	nom-d'attribut
`$defaut`	valeur

TAB. 2 – *Les facettes d'inférence et le type de leur valeur.*

Les mécanismes d'inférence mis en œuvre par SHIRKA se distinguent par leur intégration profonde dans le langage de description de classes. Ainsi, même le mécanisme le plus étranger, l'attachement procédural, fait l'objet d'une présentation toute Shirkaïenne (inspirée de SRL). Plus surprenant, l'exécution de cet attachement est aussi très intégrée: un attachement procédural est décrit par une classe (sous-classe de `méthode`) dont les attributs sont les entrées/sortie et un attribut particulier (`fonction`) contient le nom de la fonction LISP associée à l'attachement procédural. Lorsque cet attachement apparaît dans la description d'une classe (par exemple, pour l'attribut `puissance-réelle` de la classe `Auto`, plus haut) une sous-classe est créée enregistrant comment sont obtenues les entrées/sortie (ici l'objet lui-même est l'entrée et la sortie sera la valeur de l'attribut `puissance-réelle`). Enfin, l'exécution de l'attachement se passe comme suit: lorsque la valeur de l'attribut `puissance-réelle` est demandée, le système instancie cette sous-classe, renseigne les attributs d'entrée à l'aide des méthodes correspondantes, applique la fonction désignée à l'instance, récupère le résultat dans l'attribut de sortie (s'il s'y trouve) et le retourne.

Un autre mécanisme encore plus intégré est le filtrage. Il permet de chercher des instances ayant certaines caractéristiques. Un filtre est en fait défini exactement comme une classe. Ainsi, on peut implémenter la règle «L'ensemble des fils d'un homme est l'ensemble des hommes qui l'ont pour père» par le filtre suivant la facette `$sib-filtre`:

```
{Homme
  est-un    =    Classe;
  sorte-de  =    Personne;
  lui-même  $var   lui;
  père      $un    Homme;
  fils      $liste-de   Homme
            $sib-filtre
               {Homme
                 père      $var<- lui;
                 lui-même  $var->  fils}}
```

Un tel filtre retourne la liste des objets qui le satisfont (sans distinguer entre objets qui ne peuvent le satisfaire et ceux qui sont incomplets comme dans YAFOOL).

3. Développements ultérieurs

SHIRKA fut conçu comme un système de gestion de bases de connaissance, c'est-à-dire permettant d'exprimer la connaissance sachant qu'elle doit être manipulée par des programmes créés pour une tâche précise. Il a donc été développé, au dessus de SHIRKA, un certain nombre de mécanismes de manipulation de bases de connaissance (ce que l'on ne trouve pas dans les langages de programmation par objets qui considèrent que n'importe qui peut implémenter ce dont il a besoin dans le système). Ainsi, ces extensions ont implémentés des développements successifs de la recherche en représentation de connaissance. Certaines sont maintenant indissociables de SHIRKA (classification), alors que d'autres n'ont que peu (TMS) ou jamais (règles, introduction de nouvelles classes) été utilisées réellement.

3.1. Règles

Si SHIRKA a su résister à la pression de la programmation, il n'a pas pu résister à celle des règles. À une époque où elles étaient quasi-indissociables des «systèmes-experts» et à l'instar de KOOL, SMECI ou YAFOOL, il fallait bien que SHIRKA soit doté d'un système de règles. Plusieurs expériences ont été tentées: CRIKA [Rechenmann et Vignard, 1985], SOLI [Act, 1987] et [Jean-Marie et Leclerc, 1987]. Un effort d'intégration de ces règles à Shirka a été consenti; en particulier dans le dernier exemple, les règles sont représentées par des objets comme c'était le cas à l'époque. Mais le développement de SHIRKA allait plutôt vers la promotion de mécanismes d'inférence spécifiquement objet et profondément intégrés dans le système (attachement le moins procédural possible, filtrage, classification, tâches). Ces systèmes de règles n'ont pas fait fortune (ils n'ont pas non plus été distribués avec le système).

3.2. Système de maintien du raisonnement

Un système de maintien du raisonnement à propagation [Doyle, 1979] a été ajouté à SHIRKA de manière à conserver le résultat des inférences au lieu de les refaire constamment [Euzenat et Rechenmann, 1987]. Grâce à un réseau de dépendances, il est capable d'invalider les inférences qui ne sont plus valides lors de la modification d'une valeur d'attribut, d'une classe ou de l'ajout d'une méthode. Deux remarques peuvent être faites sur ce système: il a été victime du principe du tout objet (le réseau de dépendances était implanté en SHIRKA) ce qui nuit à ses performances, mais il a très bien fonctionné dans la seule application qui l'ait utilisé [Buisson, 1990]. Les études théoriques faites à ce propos [Buisson et Euzenat, 1992] furent sans ambiguïté: l'apport du TMS dans les performances était crucial pour le système. Il n'a jamais été intégré aux versions de SHIRKA distribuées (bien qu'il soit possible de le débrayer).

3.3. Classification

La classification a été introduite dans SHIRKA en 1987 [Prokop et Pivot, 1987; Haton *et al.*, 1991]. Elle n'a, depuis lors, pratiquement pas été modifiée. Ce mécanisme était à l'époque très rare dans un langage de représentation de connaissance diffusé. Classer une instance i sous une classe c, c'est trouver les sous-classes de c auxquelles i peut appartenir compte tenu des valeurs des attributs de i. Une partition des classes en trois ensembles est alors obtenue ($\tau_{f,c}$ est le type de l'attribut f dans la classe c, valeur?(i,f) est la valeur de l'attribut f pour l'instance i et ι dénote la valeur inconnue):

Possibles: si toutes les valeurs d'attributs de l'instance satisfont les contraintes de la classe: $\forall f \in$ attributs(c), valeur?$(i, f) \in \tau_{f,c}$,

Inconnues: si aucune valeur d'attribut de l'instance ne viole une contrainte de la classe mais certaines valeurs d'attributs sont inconnues: $\forall f \in$ attributs(c), valeur?$(i, f) \in \tau_{f,c} \cup \{\iota\}$,

Impossibles: s'il existe une valeur d'attribut de l'instance qui viole une contrainte de la classe: $\exists f \in$ attributs(c), valeur?$(i, f) \notin \tau_{f,c} \cup \{\iota\}$.

À partir de deux définitions d'interprétation (correspondant aux ensembles $I(c)$ et $E(c)$ de la définition des classes), la classification est la recherche des classes auxquelles une instance pourrait appartenir en respectant $E(c)$ (c'est d'ailleurs une opération duale du filtrage qui recherche les instances pouvant appartenir à une classe). Cette caractérisation de la classification a été donnée dans le formalisme des systèmes classificatoires [Euzenat, 1993a] qui permet d'établir que les propriétés associées à SHIRKA (interprétation descriptive, pas d'exhaustivité) ne permettent pas d'assurer que la classification rendra toujours un ensemble de classes possibles muni d'un unique plus petit élément, ni que ses plus petits éléments feront forcément partie des feuilles de la taxonomie.

La classification de SHIRKA — que l'on pourrait nommer identification — possède un certain nombre d'originalités (sans égal à l'époque à notre connaissance pour la classification symbolique):

- Elle prend en compte l'incomplétude des objets (le classement d'instances incomplètes apparaît dans l'ensemble *Inconnues*).
- Parallèlement aux langages terminologiques qui ne s'autorisent pas à classifier dans les concepts primitifs (correspondant à l'interprétation descriptive des classes), SHIRKA dispose d'un mécanisme de classification agissant uniquement dans des concepts primitifs. Cependant, par rapport aux langages terminologiques, la classification est plus simple puisqu'elle ne classe que des individus (l'extension aux classes a été conçue et implémentée [Capponi, 1994], mais non distribuée).
- L'algorithme prend récursivement en compte la «classifiabilité» des objets placés en valeurs d'attributs dans la classe caractérisant le type de l'attribut. Un utilisateur demandant une classification dans SHIRKA peut la voir se dérouler en temps réel et examiner les trois ensembles sur le graphe de spécialisation lui-même.

Cette classification mériterait le traitement formel qui lui fait défaut quant à la décidabilité et complexité du problème, la complétude et la complexité de l'algorithme utilisé. On peut considérer que l'implémentation «naïve» de l'algorithme devrait conduire à une complexité exponentielle. Cependant, les nombreuses applications utilisant la classification ne semblent pas

gênées par cela. Des travaux sont en cours pour établir un algorithme dont la complexité temporelle devrait être polynomiale pour une complexité spatiale linéaire.

3.4. Langages de tâches

Les travaux sur l'utilisation de bases de connaissance pour exploiter les bibliothèques de programmes existants ont conduit à la notion d'environnement de résolution de problèmes. Dans le cadre de SHIRKA, c'est la notion de tâche qui s'est trouvée la meilleure instanciation de ce principe. Les tâches apparaissent à la croisée de différentes réflexions comme:

- représentation d'une classe de problèmes [Wielinga et Breuker, 1986],
- élément de contrôle dans les systèmes à base de connaissance [Chandrasekaran, 1986],
- généralisation d'une fonctionnalité [Rousseau, 1988].

La notion de tâche exécutable a donnée lieu à deux implémentations successives au dessus de SHIRKA: SCAI [Poncabarré et Rechenmann, 1991] et SCARP[1] (Système Coopératif d'Aide à la Résolution de Problèmes [Willamowski, 1994; *et al.*, 1994]). Ce sont les principes du second qui sont — très brièvement — présentés ici.

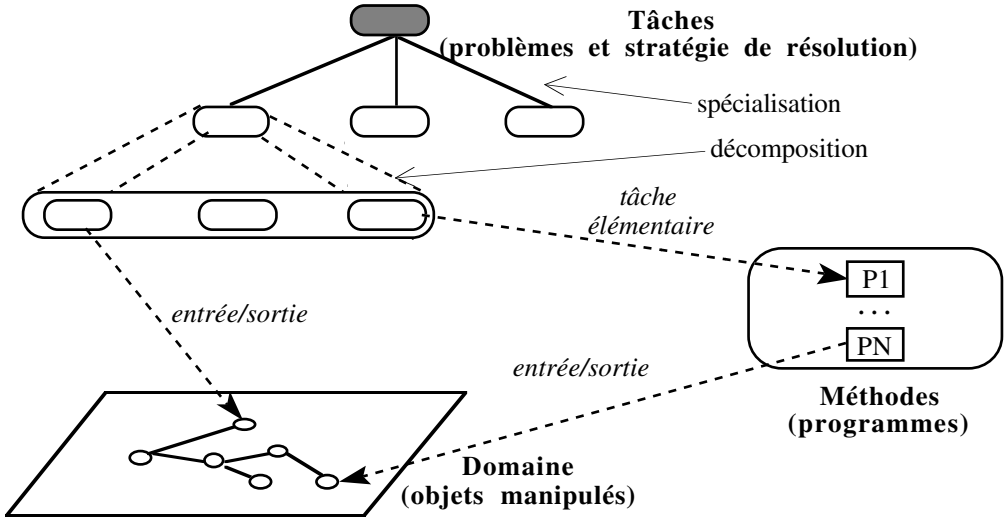

FIG. 1 (d'après [Willamowski et al., 1994) – *L'organisation des entités dans un environnement de résolution de problèmes. Les tâches sont reliées entre elles par la spécialisation et la décomposition. Elles prennent en entrée et produisent en sortie des objets du domaine (représentés par des objets SHIRKA). Elles se résolvent finalement en tâches élémentaires exécutables prenant les mêmes entrées/sorties.*

Une tâche définit un problème à résoudre. Pour cela elle utilise une représentation qui se partage en:

- ses entrées (les données à traiter) qui sont exprimées sous formes d'objets ou de valeurs;
- ses sorties (le résultat à obtenir) qui sont exprimées sous formes d'objets ou de valeurs;
- une stratégie de résolution (les tâches sans stratégie étant inapplicables).

[1] Développé en coopération avec Cap Gemini Innovation et cofinancé par le Ministère de la Recherche et de la Technologie.

Les tâches sont décrites par des classes. Elles appartiennent donc à une hiérarchie de spécialisation dans laquelle plus une classe est spécialisée plus elle permet de résoudre un problème précis. Cette précision s'exprime alors par un affinement des entrées et des sorties du système ainsi qu'une adaptation plus poussée de la stratégie de résolution. La stratégie de résolution d'une tâche peut être principalement de deux natures:

- un programme exécutable (la tâche est alors qualifiée de méthode) qui résout le problème; ces tâches permettent en particulier d'intégrer des bibliothèques de programmes externes [Prévosto et Rechenmann, 1989a, b, 1991; Bassot 1994a, b];
- une décomposition en fonction d'autres tâches construites à l'aide d'opérateurs classiques de programmation (séquence, itération, choix, parallélisme…) et des flots de données entre les entrées et sorties des tâches impliquées.

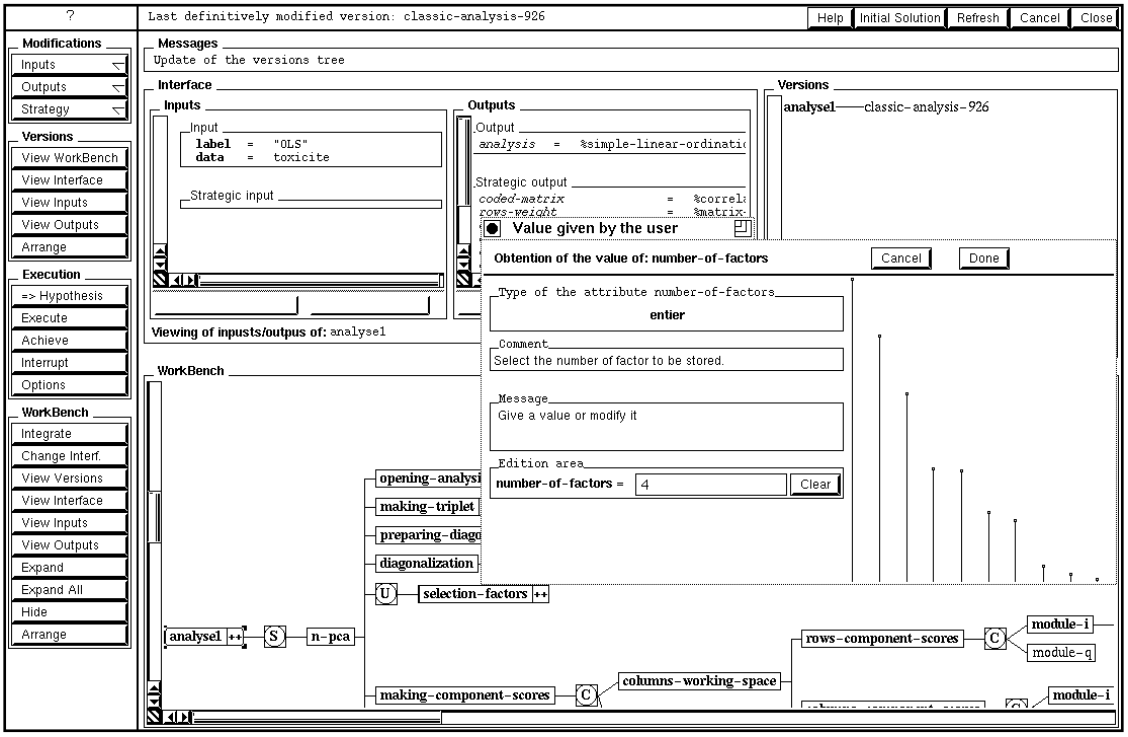

FIG. 2 (d'après [Willamowski, 1994]) – *L'interface de SCARP. On y voit en haut les entrées/sorties de la tâche en cours d'exécution, en bas l'arbre de décomposition des tâches et par dessus un interacteur demandant son avis à l'utilisateur.*

L'«exécution» d'une tâche s'accomplit comme suit:

(1) une instance de la classe représentant la tâche est créée;
(2) les entrées sont d'abord obtenues, soit par le flot de données, soit par la définition de la tâche (à l'aide des mécanismes d'inférence);
(3) la classification est utilisée pour déterminer, en fonction des entrées, les tâches applicables;
(4) les entrées manquantes sont obtenues par demande à l'utilisateur;
(5) si la tâche est une méthode, elle est exécutée, sinon la décomposition est exécutée en revenant en (1) pour chaque sous-tâche;

(6) les sorties sont obtenues grâce au flot de données et validées par l'utilisateur.

Bien entendu, ce processus est non déterministe, car la classification peut donner plusieurs résultats et il peut y avoir des opérateurs de choix dans les stratégies. Le système utilise donc des stratégies par défaut et procède à un retour-arrière en cas d'échec. En fait, le système est beaucoup plus versatile que cela puisqu'il est conçu pour être mis à la disposition d'un utilisateur qui peut, à tout moment, interrompre le processus, changer un choix par défaut, ou un choix fait précédemment par lui, et relancer l'exécution. Pour cela, il faut que l'utilisateur ait à tout moment une image du processus en cours, ce qui est offert par l'interface graphique (voir figure 2).

La sémantique opérationnelle de ce système a été étudiée dans [Crampé, 1994], elle s'apparente clairement à celle de PROLOG… qui se retrouve donc intégré plus naturellement (que par des règles) au sein des objets.

3.5. Hypertexte

SHIRKA était à l'origine utilisable en mode alphanumérique (et il l'est toujours). Mais très tôt le problème des interfaces s'est posé. On a, dans un premier temps, développé une interface permettant d'utiliser SHIRKA comme un tableur alphanumérique (l'interface privilégiée de l'époque; il est toujours livré avec SHIRKA) [Demuyter et Gonguet, 1986]. Puis, avec la diffusion d'X-windows sont apparues les interfaces graphiques que l'on connaît maintenant [Cruypenninck et Ziébelin, 1992].

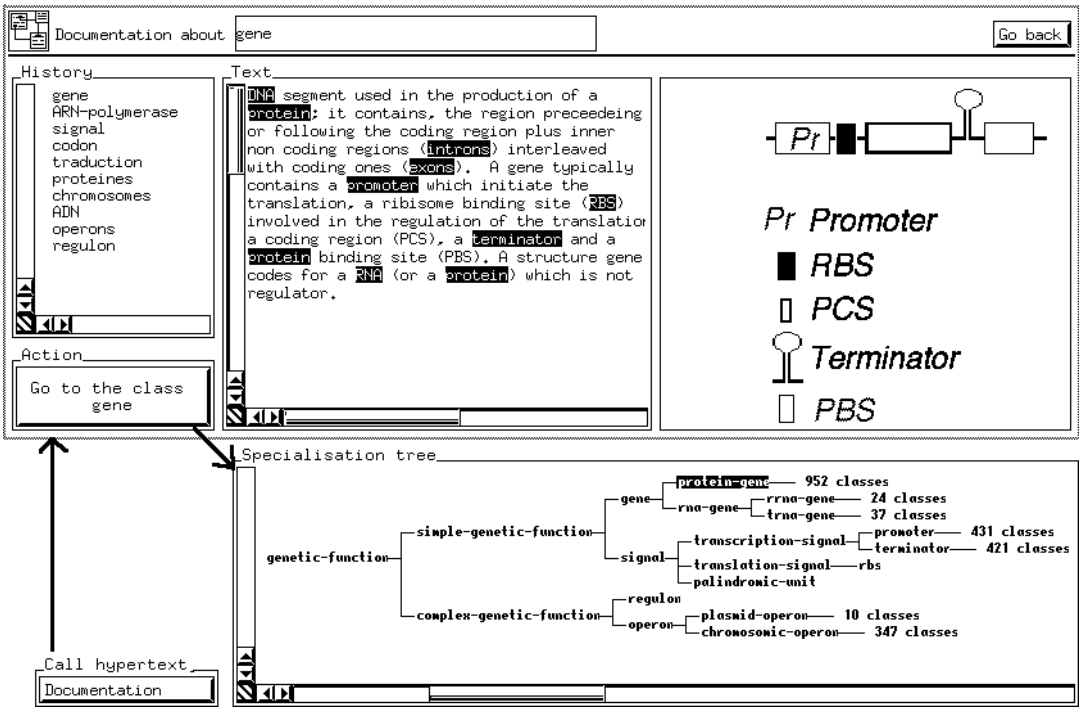

FIG. 3 (d'après [Euzenat, 1995]) – *Liaison SHIRKA-hypertexte permettant de passer du texte aux objets et vice versa.*

Mais l'expression de la connaissance sous un formalisme particulier, si elle permet sa manipulation par un ordinateur, a ses limites. En particulier, elle ne permet pas de tout exprimer et n'est pas forcément intelligible à un utilisateur qui n'a pas participé à l'élaboration de la base. Ceci a justifié l'adjonction à SHIRKA d'un système d'hypertexte permettant d'exprimer informellement des informations [Grivaud et Rechenmann, 1992]. L'aspect original du système — qui a profité de l'intérêt porté aux hypertextes dans les années 80 — est non seulement de pouvoir accéder au texte à partir des éléments de connaissance (classes, attributs…), mais aussi de pouvoir revenir aux éléments de connaissance à partir de l'hypertexte (voir figure 3).

4. Applications

C'est sous ces diverses formes que SHIRKA a été utilisé dans un certain nombre d'applications. La table 3 inventorie les applications (portées à notre connaissance) utilisant SHIRKA et SCARP. En général, beaucoup d'entre elles ont été réalisées indépendamment de l'équipe qui a développé SHIRKA et SCARP, et dont la contribution n'a consisté qu'à assister par des conseils et à prendre en compte d'éventuels problèmes.

Une telle utilisation d'un logiciel développé en laboratoire est encourageante. Au delà du nombre, on peut cependant noter deux aspects concernant ces applications:

- elles ont principalement été réalisées dans des laboratoires de recherche, par des spécialistes du domaine: elles n'ont donc pas bénéficié du support d'un intermédiaire informaticien ou «cogniticien», mais au contraire du lien direct avec le «producteur de connaissance».
- le choix de SHIRKA a pu être influencé par le fait qu'il était distribué gratuitement (bien que des formalités devaient être initialement remplies pour utiliser le système).

Du premier point vient sans doute un tournant des activités du projet SHERPA vers «ceux qui produisent la connaissance», c'est-à-dire les utilisateurs dont le métier est d'exprimer la connaissance qu'ils acquièrent (traditionnellement par des livres ou des cours), plutôt que vers «ceux qui sont chargés d'accoucher cette connaissance». Cela passe, bien entendu, par une sémantique plus claire (elle ne l'est jamais assez) du système, une définition de cette sémantique et des outils confortables d'exploitation.

5. TROPES

Le SHIRKA actuel n'a pas beaucoup changé en 10 ans. Depuis longtemps une nouvelle version est envisagée qui a pris le nom de TROPES en 1990. TROPES [Mariño 1993; Mariño *et al.*, 1990] reprend un certain nombre d'idées fortes de SHIRKA tout en changeant radicalement certains aspects. Les aspects conservés sont les suivants:

nom	destination	promoteurs	références
ÉDORA	Modélisation de systèmes dynamiques en biologie des populations	INRIA Sophia-Antipolis	Pavé et Rechenmann, 1986; Rechenmann et Rousseau, 1988; Rousseau *et al.*, 1986, Rousseau et Rechenmann, 1989; Pierret-Goldbreich, 1988a; b
ELSA	Analyse de site avalancheux	CEMAGREF Grenoble	Buisson, 1989; 1990; et Charlier, 1988
EVE	Résolution d'équations aux dérivées partielles	TIM3 Grenoble	Barras *et al.*, 1990; Rechenmann, 1991b; d
CHIMÈNE	Aide à la régression non linéaire	INRA Jouy-en-Josas	INRA, 1988
Subacosyas	Commande	Lille	Bennani, 1989
APEX	Diagnostic technique	Apside Sèvres	Billoir, 1987
SAFIR	Analyse financière	ARTEMIS Grenoble	Colin et Doize, 1986; Doize, 1986; Rechenmann et Doize, 1987
	Traitement de la parole	ICP Grenoble	Caelen *et al.*, 1988
GIDE	Gestion de dossiers médicaux	IRISA Rennes	Noussi, 1988; 1989
	Détermination d'espèces	LBGBP Lyon	Perrenou, 1986; Sieffer, 1988; Gautier *et al.*, 1992
COLIGÈNE, MULTIMAP	Connaissance sur le génome	LBGBP Lyon	Perrière *et al.*, 1993; Dorkeld 1994; Schmeltzer *et al.*, 1993
FLORIAN	Filtrage flou et analogie	LRI Orsay	Salloti, 1992
SADIG	Classification de gisement d'or	BRGM	Marcoux *et al.*, 1989
	Gestion de troupeaux	INRA Avignon	Girard, 1995
SAID	Diagnostic de plates-formes en mer par traitement du signal	IFREMER Brest	Prévosto et Rechenmann, 1989a; b; 1991; Jean-Marie *et al.*, 1991
DANAÏDE, SLOT	Analyse de données	LBGBP Lyon	Chevenet *et al.*, 1993a, b; Chevenet 1994
ANAÏS	Automatique	LAG Grenoble	Bassot 1994a; b
	Analyse de séquences génomiques	Institut Pasteur Paris, LBGBP Lyon	Uvietta et Willamowski, 1991; Médigue *et al.*, 1993; 1995
MYOSYS	diagnostic médical en électro-myographie	CHRU Grenoble	Vila *et al.*, 1987; 1990; Ziébelin *et al.*, 1994

TAB. 3 – *Applications de SHIRKA et de SCARP (sous la double barre) connues.*

- La distinction classe/instance est renforcée par un troisième niveau, le concept, qui permet de partitionner l'univers du discours de manière étanche: ceci a des incidences sur la signification et l'implémentation des objets.
- Le typage fort des attributs se voit renforcé par une disparition des types primitifs (décrits au travers de types abstraits et par conséquent extensibles) et par un système de normalisation de types permettant de faire une classification de classes correcte [Capponi, 1995].
- La connexion avec un système hypertexte repose simplement sur HTML. En plus de ce qui a été présenté plus haut, TROPES est capable de présenter la connaissance sous forme de documents HTML à la volée (c'est-à-dire que TROPES se transforme en serveur et engendre les pages à la demande au lieu de les stocker au préalable).

TROPES ajoute à cela deux nouveautés principales:

- la représentation des taxonomies de classes sous plusieurs points de vue (inspirée de [Carré, 1989]), qui rend compte de la multiplicité des approches possibles lorsqu'il s'agit de structurer un ensemble d'objets [Mariño, 1993];
- l'utilisation d'un système de résolution de contraintes, en remplacement des anciennes contraintes de SHIRKA, mais aussi en prolongement du système de types [Gensel, 1995].

Certains aspects de SHIRKA ont été abandonnés, soit parce qu'ils n'ont jamais été utilisés (règles), soit parce qu'il est délicat de leur associer une sémantique s'intégrant dans la philosophie générale du système (c'est le cas pour la réflexivité qui ne s'accorde pas uniformément avec la sémantique de la classification). Mais d'autres aspects déjà abordés seront à poursuivre:

- le maintien du raisonnement et la gestion d'hypothèses qui permettent de traiter un raisonnement disjonctif [Gensel, 1990; Gensel *et al.*, 1993] (le système de maintien du raisonnement présenté plus haut constitue une expérience à reconduire mais en reconsidérant son intégration au sein du système de satisfaction de contraintes);
- la construction automatique de taxonomies par diverses méthodes d'analyse de données [Euzenat, 1993b] ou d'apprentissage symbolique [Aguirre, 1989; Valtchev et Euzenat, 1995; Bisson, 1995];
- l'intégration d'équations qualitatives sur le comportement des objets;
- l'intégration de connaissances linguistiques sur la connaissance afin de traiter la déconnexion entre le langage courant et le langage standardisé dans une base de connaissance [Lemaire, 1995] (ceci permet aussi de traiter le multi-linguisme);
- la conception de mécanismes de construction coopérative de bases de connaissance [Euzenat, 1995].

TROPES, ainsi que sa documentation, sont disponibles sur le serveur ftp de l'IMAG (`ftp.imag.fr`) dans le catalogue `/imag/SHERPA/logiciels` (voir aussi `http://everest.imag.fr`).

6. Conclusion

L'exposé qui précède n'introduit pas d'originalité. Son but est de récapituler l'évolution d'un langage sur 10 ans. L'évolution technique de SHIRKA se traduit aussi par une évolution théorique.

Le parti-pris de déclarativité va dans le sens de l'histoire et s'est révélé payant en terme d'utilisabilité par les applications. Celui du «tout objet» a montré ses limites (réflexivité, règles, TMS) mais aussi ses avantages (filtres, classification, tâches). Ce dernier point est donc à évaluer de manière très précise avant de le réutiliser. On peut dire que plus une notion (ici objet) a d'utilisations différentes et variées, moins elle a de signification propre. Ainsi, l'exploitation tout azimut d'un concept — perçu comme un gage de sa généralité — n'est pas forcément si positive. Par exemple, l'interprétation des objets donnée ci-dessus autorise la classification telle qu'elle a été présentée mais devrait interdire l'utilisation des objets SHIRKA en tant que dépendances dans le TMS (voire même en tant que tâches). Mais pour s'en rendre compte, il est nécessaire de s'entendre au préalable sur la sémantique des notions utilisées. C'est pourquoi il est apparu petit à petit (nous ne sommes pas les premiers à nous en rendre compte, voir par exemple [Levesque et Brachman, 1987]) que la notion de déclarativité utilisée initialement devait être remplacée par celle d'interprétation. D'une manière générale, ces principes se sont donc dégagés au cours de ces dix ans comme antagonistes. Mais dix ans, c'est à la fois trop — en tout cas suffisant pour se rendre compte de cet antagonisme — et trop peu — car le développement d'une sémantique englobant la totalité du monde objet (y compris la programmation) n'a pas eu le temps de voir le jour.

Au delà des aspects conceptuels, les utilisations d'un système comme SHIRKA montrent qu'il y a réellement un besoin. Mais ces outils sont principalement utilisés par des ingénieurs ou des chercheurs non informaticiens: ils doivent donc bénéficier d'une simplicité d'utilisation (uniformité) et d'une sémantique claire.

Remerciements

Les auteurs remercient Roland Ducournau et Amedeo Napoli pour leur exigence de précision et de vérité historique, ainsi que Jean-François Perrot pour être le premier à avoir rédigé une mise en perspective des travaux du projet SHERPA.

Références

[Act, 1987] *Manuel de référence de SOLI*. Act informatique, Paris, FR, 450p., 1987

[Aguirre, 1989] J. L. Aguirre. *Construction automatique de taxinomies à partir d'exemples dans un modèle de connaissance par objet*. Thèse d'informatique, INP, Grenoble, FR, 1989

[Aikins, 1983] J. Aikins. Prototypical knowledge for expert systems. *Artificial intelligence* 20:163-210, 1983

[Albert, 1984] P. Albert. *KOOL at a glance*. Actes 6th ECAI, Pisa, IT, p345, 1984

[Barras *et al.*, 1990] P. Barras, J. Blum, J.-C. Paumier, F. Rechenmann et P. Witomski. EVE: an object-centered knowledge-based PDE solver. Dans J. Rice, R. Vichnevetsky (éds.). *Actes 2nd international conference on expert systems for numerical computing*. rapport de recherche CSD-TR-963, Purdue university, West-Lafayette, IN US, pages 1-3, 1990

[Bassot, 1994a] C. Bassot et F. Michau. Aide à l'interprétation de courbes de simulation pour l'analyse de systèmes dynamiques. *Actes 9ième RFIA*, Paris, FR, pages 721-726, 1994

[Bassot, 1994b] C. Bassot. *ANAÏS: un outil d'aide à l'analyse interactive de simulations*. Thèse d'automatique, INPG, Grenoble, FR, 1994

[Bennani, 1989] M. Bennani. *Subacosyas: un superviseur à base de connaissances en synthèse d'asservissement*. Thèse d'informatique, université des sciences et technologies de Lille Flandre artois, Compiègne, FR, 1989

[Bensaid *et al.*, 1986] A. Bensaid, F. Rechenmann, A. Simonet et P. Vignard. Mécanismes d'inférence et d'explication dans des bases de connaissances centrées-objet. *Actes 8èmes journées francophones sur l'informatique «Bases de données et bases de connaissances»*, Grenoble, FR, 1986

[Billoir, 1987] T. Billoir. Élaboration de diagnostics techniques par génération et propagation de suspicions et accusations. *Actes 7ièmes journées internationales sur les systèmes experts et leurs applications*, Avignon, FR, pages 867-885, 1987

[Bisson, 1995] G. Bisson. Why and how to define a similarity measure for object-based representation systems. Dans Nicolaas Mars (éd.). *Towards very large knowledge bases*. IOS press, Amsterdam, NL, pages 236-246, 1995

[Bobrow et Stefik, 1981] D. Bobrow et M. Stefik. *The LOOPS manual*. Memo KB-VLSI-81-13, Xerox PARC, Palo Alto, CA US, 1981

[Bobrow et Winograd, 1977] D. Bobrow et T. Winograd. An overview of KRL: knowledge representation language. *Cognitive science* 1(1):3-45, 1977

[Brachman et Schmolze, 1985a] R. Brachman et J. Schmolze. An overview of the KL-one knowledge representation system. *Cognitive science* 9(2):171-216, 1985

[Brachman, 1979] R. Brachman. On the epistemological status of semantic networks. Dans N. Findler (éd.). *Associatives networks: representation and use of knowledge by computers*. Academic Press, New-York, NY US, pages 3-50, 1979

[Brachman, 1983] R. Brachman. What is-a is and isn't: an analysis of semantics links in semantic networks. *IEEE Computer* 16(10):30-36, 1983

[Brachman, 1985] R. Brachman. "I lied about the trees" or, defaults and definitions in knowledge representation. *AI magazine* 6(3):80-93, 1985

[Buisson et Charlier, 1988] L. Buisson et C. Charlier. Avalanche starting zone analysis with a knowledge-based system. *Actes International Glaciological Society symposium on snow and glacier research relating to human living conditions*, Lom, NW, 1988

[Buisson et Euzenat, 1992] L. Buisson et J. Euzenat. The ELSA avalanche path analysis system: an experiment with reason maintenance and object-based representations (extended abstract).

Actes ECAI 92 workshop on «Applications of Reason Maintenance Systems», Vienne, OS, pas de pagination, 1992

[Buisson, 1989] L. Buisson. Reasoning on space with knowledge object centered representation. *Lecture notes on computer science* 409:325-344, 1990

[Buisson, 1990] L. Buisson. *Le raisonnement spatial dans les systèmes à base de connaissances — application à l'analyse de sites avalancheux*. Thèse d'informatique, université Joseph Fourier, Grenoble, FR, 1990

[Caelen *et al.*, 1988] J. Caelen, O. Cervantes, J.-F. Serignat et Y. Fernandez, Data and knowledge for speech processing. Dans J. Demongeot, T. Hervé, V. Rialle et C. Roche (éds.). *Artificial intelligence and cognitive sciences*. Manchester University Press, Manchester, GB, pages 23-48, 1988

[Capponi, 1994] C. Capponi. Interactive class classification using types. Dans E. Diday, Y. Lechevallier, M. Schader, P. Bertrand et B. Burtschy (éds.). *New approaches in classification and data analysis*. Springer-Verlag, Berlin, DE, pages 204-211, 1994

[Capponi, 1995] C. Capponi. *Identification et exploitation des types dans un modèle de connaissances à objets*. Thèse d'informatique, université Joseph-Fourier, Grenoble, FR, 1995 en préparation

[Carré, 1989] B. Carré, *Méthodologie orientée-objet pour la représentation des connaissances: concepts de points de vue, de représentation multiple et évolutive*, Thèse d'informatique, université de Lille, FR, 1989

[Chailloux, 1983] J. Chailloux. *Le_Lisp 80, version 12: le manuel de référence*. Rapport technique 27, INRIA Rocquencourt, FR, 1983

[Chandrasekaran, 1986] B. Chandrasekaran. Generic tasks in knowledge-based reasoning: high-level building tools for expert-system design. *IEEE Expert* [1(4)]:23-30, 1986

[Chevenet *et al.*, 1993a] F. Chevenet, F. Jean-Marie et J. Willamowski. A development shell for cooperative problem solving enviroments. Dans E. Houstis, J. Rice et R. Vichnevetsky (éds.). *Actes 3rd international conference on expert systems for numerical computing*. Rapport technique CSD-TR-93-028, Purdue university, West Lafayette, IN US, pages 1-7, 1993

[Chevenet *et al.*, 1993b] F. Chevenet, F. Jean-Marie et J. Willamowski. SLOT: a cooperative problem-solving environment in explanatory data analysis. *Actes 49th session of the International Statistical Institute*, Firenza, IT, pages 255-256, 1993

[Chevenet, 1994] F. Chevenet. *Environnements coopératifs de résolution de problèmes pour l'analyse statistique: représentation objets et analyse des données multivariées en écologie*. Thèse de biométrie, université Claude-Bernard, Lyon, FR, 1994

[Cointe, 1985] P. Cointe. Le modèle objVlisp: une plate-forme pour l'expérimentation des formalismes objets. *Actes 5ième RFIA*, Grenoble, FR, pages 737-754, 1985

[Colin et Doize, 1986] C. Colin et M.-S. Doize. *SAFIR: Système-expert d'Analyse Financière par Interprétation de Ratios*. Mémoire de projet ENSIMAG, Grenoble, FR, 1986

[Crampé, 1994] I. Crampé. *Sémantique des tâches décomposables et spécialisables*. Mémoire de DEA d'informatique, INPG-université Joseph Fourier, Grenoble, FR, 1994

[Cruypenninck et Ziébelin, 1992] F. Cruypenninck et D. Ziébelin. Classification d'objets complexes dans les bases de connaissances. *Actes 1ères journées «représentations par objets» (RPO)*, La Grande Motte, FR, pages 59-72, 1992

[Demuyter et Gonguet, 1986] S. Demuyter et J.-F. Gonguet. *Un tableur comme interface utilisateur d'un environnement de développement de systèmes experts*. Mémoire de projet ENSIMAG, Grenoble, FR, 1986

[Doize, 1986] M.-S. Doize. *Évaluation de modèles pour la représentation de connaissances. Application à la réalisation de SAFIR. Système d'Analyse Financière par Interprétation de Ratios*. Mémoire de DEA informatique, UJF-INP, Grenoble, FR, 1986

[Dorkeld, 1994] F. Dorkeld. *MULTIMAP: un modèle objet dédié à la cartographie comparée des génomes de mammifères*. Thèse de biométrie, université Claude Bernard, Lyon, FR, 1994

[Doyle, 1979] J. Doyle. A truth maintenance system. *Artificial intelligence* 12(3):231-272, 1979

[Ducournau et Quinqueton, 1986] R. Ducournau et J. Quinqueton. *YAFOOL: encore un langage objet à base de frames! version 2.1*. Rapport technique 72, INRIA, Rocquencourt, FR, 1988

[Euzenat et Rechenmann, 1987] J. Euzenat et F. Rechenmann. Maintenance de la vérité dans les systèmes à base de connaissance centrée-objet. *Actes 6ième RFIA*, Antibes, FR, pages 1095-1109, 1987

[Euzenat, 1993a] J. Euzenat. Une définition abstraite de la classification et son application aux taxonomies d'objets. *Actes 2ndes journées «représentations par objets» (RPO)*, La Grande Motte, FR, pages 235-246, 1993

[Euzenat, 1993b] J. Euzenat. Brief overview of T-tree: the Tropes Taxonomy building Tool. *Actes 4th ASIS SIG/CR classification research workshop*, Columbus, OH US, pages 69-87, 1993

[Euzenat, 1995] J. Euzenat. Building consensual knowledge bases: context and architecture. Dans Nicolaas Mars (éd.). *Towards very large knowledge bases*. IOS press, Amsterdam, NL, pages 143-155, 1995

[Ferber, 1983] J. Ferber. *MERING: un langage d'acteurs pour la représentation et la manipulation des connaissances*. Thèse d'informatique, université Pierre et Marie Curie, Paris, FR, 1983

[Fikes et Kehler, 1985] R. Fikes et T. Kehler. The role of frame-based representation in reasoning. *Communication of the ACM* 28(9):904-920, 1985

[Gautier *et al.*, 1992] N. Gautier, A. Pavé et F. Rechenmann. Object centered representation and fish identification in Antarctica. Dans R. Fortuner (éd.). *Advanced computer methods for systematic biology*. The Johns Hopkins university press, Baltimore, ML US, pages 181-195, 1993

[Gensel *et al.*, 1993] J. Gensel, P. Girard et O. Schmeltzer. Integrating constraints. composite objects and tasks in a knowledge representation system. *Actes 5th international conference on tools with artificial intelligence*, Cambridge, MA US, pages 127-130, 1993

[Gensel, 1990] J. Gensel. *Gestion des dépendances et des hypothèses dans un modèle de connaissances à objets*. Mémoire de DEA informatique, UJF-INP, Grenoble, FR, 1990

[Gensel, 1995] J. Gensel. *Contraintes et représentation de connaissances par objets: application au modèle TROPES*. Thèse d'informatique, université Joseph Fourier, Grenoble, FR, 1995 en préparation

[Girard, 1995] N. Girard. Structurer la description des cas et élaborer un langage pour guider l'acquisition des connaissances: une expériences en agronomie. *Actes 6ièmes Journées «acquisition de connaissance»*, Grenoble, FR, pages 287-300, 1995

[Goodwin, 1979] J. Goodwin. *Taxonomic programming with KLONE*. Rapport de recherche LITH-MAT-R-79-5, Linköping university, Linköping, SE, 1979

[Grivaud et Rechenmann, 1992] S. Grivaud et F. Rechenmann. Navigation dans les bases de connaissances associant objets et hypertexte. *Actes 1ères journées «représentations par objets» (RPO)*, La Grande Motte, FR, pages 262-280, 1992

[Haton *et al.*, 1991] J.-P. Haton, N. Bouzid, F. Charpillet, M.-C. Haton, B. Lâasri, H. Lâasri, P. Marquis, T. Mondot et A. Napoli. *Le raisonnement en intelligence artificielle*. InterÉditions, Paris, FR, 1991

[Hullot, 1983] J.-M. Hulot. *Ceyx: a multi-formalism programming environment*. Rapport de recherche 210, INRIA, Rocquencourt, FR, 1983

[INRA88] *Le projet CHIMENE*, document ronéoté, INRA, Jouy-en-Josas (FR), 1988

[Jean-Marie *et al.*, 1991] F. Jean-Marie, B. Rousseau, F. Rechenmann et M. Prevosto. Embedding knowledge within scientific computing codes: an application to offshore structure vibration analysis. *Actes EUROCAIP*, Rueil-Malmaison, FR, 1991

[Jean-Marie et Leclerc, 1987] F. Jean-Marie et A. Leclerc. *Quand les règles rencontrent les objets*. Mémoire de projet ENSIMAG, Grenoble, FR, 1987

[Laurière, 1982] J.-L. Laurière. Représentation et utilisation des connaissances. *Techniques et science informatique* 1(1):25-42 & 1(2):109-133, 1982

[Lemaire, 1995] F. Lemaire. Improving the legibility of a knowledge base. *Actes 1st International Symposium on Knowledge Retrieval, Use, and Storage for Efficiency*, Santa Cruz, CA US, pages 102-111, 1995

[Levesque et Brachman, 1987] H. Levesque et R. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational intelligence/intelligence informatique* 3(2):78-93, 1987

[Marcoux *et al.*, 1989] E. Marcoux, P. Puvilland et F. Rechenmann. SADIG: un système expert de classification précoce de gisements d'or. *Actes 1ère convention IA*, Paris, FR, pages 607-620, 1989

[Mariño *et al.*, 1990] O. Mariño, F. Rechenmann et P. Uvietta. Multiple perspectives and classification mechanism in Object-oriented Representation. *Actes 9th ECAI*, Stockholm, SE, pages 425-430, 1990

[Mariño, 1993] O. Mariño. *Raisonnement classificatoire dans une représentation à objets multi-points de vue*. Thèse d'informatique, université Joseph-Fourier, Grenoble, FR, 1993

[Médigue *et al.*, 1993] C. Médigue, J. Willamowski, O. Schmeltzer, P. Uvietta, F. Rechenmann, F. Chevenet, G. Perrière et C. Gauthier. Modeling tasks for problem solving in molecular biology.

Actes IJCAI workshop on «artificial intelligence and genome» (rapport technique LAFORIA 93/30, Paris, FR, 1993), Chambéry, FR, pages 67-76, 1993

[Médigue *et al.*, 1995] C. Médigue, T. Vermat, G. Bisson, A. Viari et A. Danchin. Cooperative Computer System for Genome Sequence. *Actes 3ième ISMB*, Cambridge, GB, pages 249-258, 1995

[Minsky, 1974] M. Minsky. *A framework for representing knowledge*. Rapport technique AI-memo 306, MIT, Cambridge, MA US, (rep. dans P. Winston (éd.). *The psychology of computer vision*. Mac Graw-Hill, New York, NY US, pages 211-277, 1975), 1974

[Noussi, 1988] R. Noussi. GIDE: a system for intelligent handling of medical files of epileptics. Dans J. Demongeot, T. Hervé, V. Rialle et C. Roche (éds.). *Artificial intelligence and cognitive sciences*. Manchester University Press, Manchester, GB, pages 361-374, 1988

[Noussi, 1989] R. Noussi. *Un modèle objectal pour la synthèse de dossiers médicaux*. Thèse d'informatique, université de Rennes 1, Rennes, FR, 1989

[Pavé et Rechenmann, 1986] A. Pavé et F. Rechenmann. Computer aided modelling in biology: an artificial intelligence approach. Dans E. Kerckhoffs, G. Vansteenkiste et B. Ziegler (éds.). *Artificial Intelligence and Simulation*. SCS series 18, pages 52-66, 1986

[Perrenou, 1986] C. Perrenou. *Détermination spécifique et système expert*. Mémoire de DEA Analyse et modélisation des systèmes biologiques, université Claude Bernard, Lyon, FR, 1986

[Perrière *et al.*, 1993] G. Perrière, F. Dorkeld, F. Rechenmann et C. Gautier. Object-oriented knowledge bases for the analysis of prokaryotic and eukaryotic genomes. *Actes 1st ISMB*, Bethesda, MD US, pages 319-327, 1993

[Pierret-Golbreich, 1988a] C. Pierret-Golbreich. *Structuration des connaissances et raisonnement à l'aide d'objets*. Rapport de recherche 847, INRIA, Rocquencourt, FR, 1988

[Pierret-Golbreich, 1988b] C. Pierret-Golbreich. *Vers un système à base de connaissances centrée-objet pour la modélisation de systèmes dynamiques en biologie*. Thèse d'informatique, université technologique de Compiègne, Compiègne, FR, 1988

[Poncabaré et Rechenmann, 1991] T. Poncabaré et F. Rechenmann. SCAI: un environnement de développement de systèmes à base de connaissances en calcul scientifique et technique. *Actes 3ième convention intelligence artificielle*, Paris, FR, pages 491-509, 1991

[Prevosto et Rechenmann, 1989a] M. Prevosto et F. Rechenmann. Présentation et démonstration du logiciel SAID d'aide au traitement du signal. *Actes journée de clôture des préétudes C2A*, CNET, Issy-les-Moulineaux, FR, 1989

[Prevosto et Rechenmann, 1989b] M. Prevosto et F. Rechenmann. Système d'aide à l'inspection de structures en génie mécanique. *Actes séminaire automatique et intelligence artificielle*, Groupement de Recherche Automatique, Pôle «Automatisation Intégrée», CNRS, Paris, FR, 1989

[Prevosto et Rechenmann, 1991] M. Prevosto et F. Rechenmann. SAID: a knowledge-based system in signal processing. *Actes conférence européenne d'automatique*, Grenoble, FR, 1991

[Procop et Pivot, 1987] M. Procop et B. Pivot. *Définition et réalisation d'une fonctionnalité de classification dans SHIRKA*. Mémoire d'année spéciale «Intelligence Artificielle», ENSIMAG, Grenoble, FR, 1987

[Rechenmann *et al.*, 1984] F. Rechenmann, A. Bensaid et D. Granier. SHIRKA: des systèmes experts centrés-objet. *Actes 4ièmes journées internationales sur les systèmes-experts et leurs applications*, actes polycopiés, Avignon, FR, 1984

[Rechenmann *et al.*, 1988] F. Rechenmann, P. Fontanille et P. Uvietta. *SHIRKA: manuel d'utilisation*. Rapport interne, INRIA — Laboratoire ARTEMIS, Grenoble, FR, 1988

[Rechenmann et Doize, 1987] F. Rechenmann et M.-S. Doize. SAFIR-SHIRKA: un système à base de connaissance centrée-objet pour l'analyse financière. *Actes 7ièmes journées internationales sur les systèmes-experts et leurs applications*, Avignon, FR, pages 949-967, 1987

[Rechenmann et Rousseau, 1988] F. Rechenmann et B. Rousseau. ÉDORA: an object-centered knowledge-based approach to dynamic modelling. Dans J. Demongeot, T. Hervé, V. Rialle et C. Roche (éds.). *Artificial intelligence and cognitive sciences*. Manchester University Press, Manchester, GB, pages 107-120, 1988

[Rechenmann et Uvietta, 1989] F. Rechenmann et P. Uvietta. SHIRKA: an object-centered knowledge bases management system. Dans A. Pavé et G. Vansteenkiste (éds.). *Artificial intelligence in numerical and symbolic simulation*, ALEAS, Lyon, FR, pages 9-23, 1991

[Rechenmann et Vignard, 1985] F. Rechenmann et P. Vignard. *CRIKA: quand les règles rencontrent les objets*. Rapport de recherche 468, INRIA, Sophia-Antipolis, FR, 1985

[Rechenmann, 1985] F. Rechenmann. SHIRKA: mécanismes d'inférence sur une base de connaissance centrée-objet. *Actes 5ième RFIA*, Grenoble, FR, pages 1243-1254, 1985

[Roberts et Goldstein, 1977] R. Roberts et I. Goldstein. *The FRL manual*. Rapport de recherche AIM-409, MIT, Cambridge, MA US, 1977

[Roche, 1984] C. Roche. *LRO: génération de systèmes experts.* Thèse d'informatique, INPG, Grenoble, FR, 1984.

[Rousseau *et al.*, 1986] B. Rousseau, A. Pavé, F. Rechenmann et M. Landau. ÉDORA Project: artificial intelligence approach and workstation concept to aid dynamic modelling in biology and ecology. *Actes Summer Computer Simulation conference*, Reno, Nevada US, pages 14-20, 1986

[Rousseau et Rechenmann, 1989] B. Rousseau et F. Rechenmann. Apports des techniques de l'intelligence artificielle au développement d'environnements de résolution de problèmes. *Actes 1ère convention intelligence artificielle*, Paris, FR, pages 129-155, 1989

[Rousseau, 1988] B. Rousseau. *Vers un environnement de résolution de problèmes en biométrie — apport des techniques de l'intelligence artificielle et de l'interaction graphique.* Thèse de biométrie, université Claude Bernard, Lyon, FR, 1988

[Salotti, 1992] S. Salotti. *Filtrage flou et représentation centrée-objet pour raisonner par analogie: le système FLORIAN*. Thèse d'informatique, université Paris-Sud, Orsay, FR, 1992

[Schmeltzer *et al.*, 1993] O. Schmeltzer, C. Médigue, P. Uvietta, F. Rechenmann, F. Dorfeld, G. Perrière et C. Gautier. Building large knowledge bases in molecular biology. *Actes 1st ISMB*, Bethesda, MD US, pages 345-353, 1993

[Sieffer, 1988] A. Sieffer. *Constitution d'une base de connaissances centrée-objet en vue de l'identification de poissons des Kerguelen à l'aide du système SHIRKA*. Mémoire de DEA Analyse et modélisation des systèmes biologiques, université Claude Bernard, Lyon, FR, 1988

[Uvietta et Willamowski, 1991] P. Uvietta et J. Willamowski. Modélisation de connaissances en biologie moléculaire. *Actes 8ième RFIA*, Villeurbanne, FR, pages 1067-1078, 1991

[Valtchev et Euzenat, 1995] P. Valtchev et J. Euzenat. Classification of concepts through products of concepts and abstract data types. *Actes 1st international conference on data analysis and ordered structures*, Paris, FR, pages 131-134, 1995

[Vila *et al.*, 1987] A. Vila, L. Kress, V. Rialle, C. Robert et D. Ziébelin. Are expert systems an aid for diagnosing neuropathies?. *Electroencephalography and clinical neurophysiology* 66, 1987

[Vila *et al.*, 1990] A. Vila, D. Ziébelin et V. Rialle. Which Expert Systems for which diagnosis?. *Actes 4th international congress on computerized and quantitative EMG*, Mainz, DE, 1990

[Wielinga et Breuker, 1986] B. Wielinga et J. Breuker. A model of expertise. *Actes 7th ECAI*, Brighton, GB, pages 306-318, 1986

[Willamowski *et al.*, 1994] J. Willamowski, F. Chevenet et F. Jean-Marie. A development shell for cooperative problem-solving environments. *Mathematics and computers in simulation* 36(4-6):361-379, 1994

[Willamowski, 1994] J. Willamowski. *Modélisation de tâches pour la résolution de problèmes en coopération système-utilisateur*. Thèse d'informatique, université Joseph Fourier, Grenoble, FR, 1994

[Williams, 1984] C. Williams. ART: *The advanced reasoning tool, conceptual overview*. Inference Corporation, Los Angeles, CA US, 1984

[Winston et Horn, 1981] P. Winston et B. Horn. *Lisp*. Addison-Wesley, Reading, MA US, 1981

[Wright et Fox, 1983] J. Wright et M. Fox. *SRL/1.5 user manual*. Carnegie-Mellon university, Pittsburg, PA US, 1983

[Ziébelin *et al.*, 1994] D. Ziébelin, P. Valtchev, I. Iordanova et A. Vila. Classification of complex objects in a knowledge-based diagnosis system. *Actes 6th international conference on artificial intelligence: methodology, systems, applications*, Sofia, BG, pages 269-279, 1994