



Université de Caen / Basse-Normandie
U.F.R. de Sciences
École doctorale Structure, Information, Matière et Matériaux

Thèse

présentée par

Valérie Ficet-Cauchard

en vue de l'obtention du

Doctorat de l'Université de Caen

Spécialité : Informatique

Réalisation d'un système d'aide à la conception d'applications de Traitement d'Images : une approche basée sur le Raisonnement à Partir de Cas.

soutenue le 14 janvier 1999

devant le jury composé de

M ^{me} Catherine Garbay	Rapporteur
M. Gilles Kassel	Examineur
M. Alain Mille	Rapporteur
M ^{lle} Christine Porquet	Examineur
M ^{me} Marinette Revenu	Directeur de thèse
M. Khaldoun Zreik	Examineur



Groupe de recherches en informatique, image et instrumentation de Caen
GREYC – UPRESA CNRS 6072
Institut des sciences de la matière et du rayonnement
ISMRA de Caen

Remerciements

Je voudrais tout d'abord témoigner ma reconnaissance aux personnes avec qui j'ai plus particulièrement travaillé durant ces trois années de thèse :

- Madame Marinette Revenu, Professeur à l'ISMRA de Caen, mon directeur de thèse, pour m'avoir fait confiance dès le stage de DEA et pour m'avoir guidée et encouragée tout au long de ces trois années,
- Mademoiselle Christine Porquet, Maître de Conférence à l'ISMRA de Caen, pour ses précieux conseils et remarques, en particulier lors de la rédaction de ce manuscrit,
- Monsieur Régis Clouard, Maître de Conférence à l'IUT d'Alençon, pour avoir tenté de me faire partager ses vastes connaissances sur les systèmes de Traitement d'Images à Base de Connaissances en particulier, et sur le Traitement d'Images en général,

Je voudrais aussi remercier les personnes qui m'ont fait l'honneur de juger mon travail :

- Madame Catherine Garbay, Chargée de Recherche au CNRS, et Monsieur Alain Mille, Maître de Conférence à l'Université de Lyon et tout nouvellement Habilité à Diriger des Recherches, qui ont accepté la charge de rapporteur, pour leurs remarques pertinentes et leurs conseils,
- Monsieur Gilles Kassel, Professeur à l'Université de Picardie, pour l'intérêt qu'il a manifesté envers mes travaux en acceptant de participer à ce jury,
- Monsieur Khaldoun Zreik, Professeur à l'Université de Caen, pour avoir présidé ce jury et pour les discussions intéressantes que nous avons eu, en particulier dans le cadre du groupe apprentissage,

Je tiens également à remercier Abderahim Elmoataz et François Angot pour s'être pliés amicalement à l'explicitation du savoir-faire qu'ils mettent en œuvre dans leurs applications.

J'associe également à ces remerciements tous les autres membres de l'Equipe Image : Alexandre, Bruno, Christophe, Cyril, Daniel, Hubert, Jalal, Michel, les deux Nicolas, Rémy, Ronan, Sébastien, Serge, Sophie et Su. Je pense en particulier aux croissants du matin, aux discussions du déjeuner et aux blagues de la pause café (je suis sûre que les plus concernés se reconnaîtront ...).

Enfin, je remercie mes collègues du « groupe apprentissage » pour l'ambiance sympathique mais non moins travailleuse qui règne lors des réunions de travail.

« Last but not least », je remercie de tout cœur Christophe, ma famille et tous nos amis pour le soutien affectif et moral qu'ils ont su m'apporter tout au long de ces trois années.

Sommaire

Thèse 1

Valérie Ficet-Cauchard.....	1
Doctorat de l'Université de Caen.....	1
Réalisation d'un système d'aide à la conception d'applications de Traitement d'Images : une approche basée sur le Raisonnement à Partir de Cas.....	1
Introduction	10
I. Ingénierie des connaissances et systèmes de Traitement d'Images	16
I.1. Introduction.....	18
I.2. Etude sur les méthodes d'acquisition de connaissances.....	19
I.2.1. Modèles de représentation des connaissances.....	21
I.2.2. Intervention de l'utilisateur dans le contrôle de l'acquisition et de l'opérationnalisation ..	29
I.3. Représentation des connaissances dans les systèmes de TI	31
I.3.1. Connaissances statiques en TI et connaissances du domaine de l'image.....	32
I.3.2. Connaissances stratégiques du TI.....	39
I.3.3. Résolution de problèmes de TI.....	44
I.3.3.1 Définition d'un problème	44
I.3.3.2 Mise en œuvre des connaissances stratégiques	45
I.3.3.3 Représentation des résultats.....	52
I.3.4. Rôle de l'utilisateur dans les systèmes de TI.....	53
I.3.5. Conclusion sur les systèmes de TI	55
II. Le Raisonnement à Partir de Cas pour l'aide à la réutilisation des connaissances	58
II.1. Introduction.....	60
II.2. Représentation et organisation des cas	63
II.2.1. Problèmes de conception	64
II.2.2. Problèmes de planification.....	66

II.2.3. Autres types de problèmes.....	67
II.2.4. Bilan sur la représentation des cas.....	68
II.3. La phase de remémoration	69
II.3.1. Remémoration en deux étapes.....	69
II.3.2. Remémoration multi-étapes.....	71
II.3.3. Remémoration à options.....	73
II.3.4. Bilan sur la phase de remémoration	75
II.4. La phase d'adaptation	75
II.4.1. Systèmes de conception.....	76
II.4.2. Systèmes de planification	79
II.4.3. Bilan sur la phase d'adaptation.....	80
II.5. Mémorisation d'un nouveau cas	81
II.5.1. Mémorisation à partir des traces de résolution.....	81
II.5.2. Mémorisation par abstraction	81
II.5.3. Mémorisation à partir de modèles	82
II.5.4. Bilan sur l'apprentissage des cas	84
II.6. Conclusion sur les systèmes de RàPC.....	84
III. Le système TMO.....	86
III.1. Motivations et objectifs	88
III.1.1. Nos motivations.....	88
III.1.2. Objectifs de notre système.....	89
III.2. Architecture du système TMO	93
III.2.1. Une architecture à trois niveaux	93
III.2.2. Articulation des trois niveaux	94
III.3. Modélisation des connaissances TMO	95
III.3.1. Notions de Tâche, Méthode et Outil	96
III.3.2. Représentation de l'architecture à l'aide du modèle	100
III.3.2.1 Exemple de tâche de métacontrôle.....	101
III.3.2.2 Exemple de tâche de contrôle	101
III.3.2.3 Exemple de tâche du domaine.....	102
III.3.2.4 Exemple d'articulation entre les trois niveaux.....	103

III.3.3. Intérêt du modèle	104
IV. Aide à la réutilisation des connaissances	106
IV.1. Motivations et objectifs	108
IV.1.1. Nos motivations.....	108
IV.1.2. Objectifs du module de RàPC	109
IV.2. Représentation des cas	110
IV.2.1. Choix des critères de sélection	111
IV.2.2. Calcul de la similarité entre deux cas.....	115
IV.3. L'algorithme de sélection/adaptation	119
IV.3.1. Principe général de l'algorithme	119
IV.3.2. Sélection d'un cas source	120
IV.3.3. Adaptation interactive d'un plan	121
IV.4. La mémorisation d'un nouveau cas	123
IV.5. Conclusion	124
V. Implémentation et expérimentation	126
V.1. Implémentation du système interactif et de la base de cas.....	128
V.1.1. Les éléments de l'architecture	128
V.1.2. Les fonctionnalités de l'interface graphique.....	135
V.2. Déroulement d'une session de création d'un plan à l'aide de l'interface.....	139
V.3. Déroulement d'une session d'exécution d'un plan	144
V.4. Déroulement d'une session de création d'un plan par RàPC.....	148
V.5. Quelques exemples de plans intégrés	154
Conclusion	176
Bibliographie.....	182

Introduction

Le travail décrit dans ce mémoire a été réalisé au sein de l'Equipe Image du GREYC, dans le cadre de « l'Atelier Logiciel d'Intégration de Connaissances pour le Traitement et l'Interprétation d'Images » dont le but est la création d'un environnement informatique qui permette d'acquérir et d'intégrer des connaissances en TI au fur et à mesure de leur mise en évidence par la réalisation d'applications. La motivation sous-jacente est non seulement de résoudre des applications réelles, mais aussi de formaliser, c'est-à-dire de représenter et de structurer, la connaissance des experts.

En TI, la mise au point d'une application est une activité longue et complexe : des difficultés surviennent dans les trois phases que constituent la définition du problème, l'élaboration d'une solution et l'évaluation des résultats. Les difficultés de la première et de la dernière étape proviennent essentiellement du fait qu'elles doivent être réalisées en collaboration entre l'expert du domaine de l'image et l'expert en TI (appelé aussi traiteur d'images dans la suite du mémoire). Les difficultés de la deuxième étape sont dues d'une part à la multiplicité et la fragilité des solutions, et d'autre part au fait que les connaissances du traiteur d'images reposent beaucoup sur un savoir-faire expérimental.

Le type d'approche choisi dans l'Atelier Logiciel pour la mise au point d'applications de TI s'appuie sur l'exploitation « intelligente » de bibliothèques de programmes, appelés opérateurs de TI. Suivant cette approche, construire une application de TI consiste à enchaîner et à paramétrer un ensemble d'opérateurs d'une bibliothèque donnée. Parmi les systèmes de TI qui utilisent de telles bibliothèques, on distingue les environnements de programmation graphique grâce auxquels l'utilisateur peut sélectionner et enchaîner des opérateurs (mais pas expliquer ni modéliser son raisonnement) et les planificateurs automatiques qui, ayant une représentation et une explicitation des connaissances, permettent leur réutilisation (mais qui permettent peu à l'utilisateur d'intervenir dans la résolution).

Une première approche a été étudiée dans la thèse de Régis Clouard [Clouard 94]. Cette thèse avait pour but la réalisation d'un générateur automatique de plans de TI basé sur une architecture de Tableau Noir : BORG. Les difficultés rencontrées pour l'écriture des sources de connaissances de BORG ont conduit vers une démarche plus pragmatique de construction interactive d'applications. C'est cette nouvelle démarche qui est à la base des travaux présentés dans ce mémoire.

En effet, notre objectif est la réalisation d'un système qui bénéficie des avantages dus à l'interactivité des environnements de programmation graphique pour la sélection et l'enchaînement des opérateurs, mais qui donne également à l'utilisateur la possibilité de modéliser et d'expliquer le

raisonnement qui l'a amené à cet enchaînement. Le but est de réutiliser la stratégie mise en œuvre, à l'instar des systèmes automatiques.

Une application réelle peut nécessiter l'enchaînement de plusieurs dizaines d'opérateurs. Pour expliciter le raisonnement utilisé lors de la mise au point de cet enchaînement, nous proposons de modéliser les applications sous forme de plans. Un plan de TI est obtenu par décomposition hiérarchique du problème posé en problèmes plus simples. Chaque problème ou sous-problème est associé à une tâche de TI, qui, suivant son niveau dans le plan, exprime le but recherché, la technique à employer ou l'algorithme à appliquer.

La modélisation sous forme de plans d'applications de TI dans divers domaines fait apparaître rapidement le besoin de pouvoir réutiliser des parties de plans construits auparavant. En effet, on peut se retrouver en face d'un problème qui a déjà été résolu dans un domaine d'application différent. Par exemple, on voudra réutiliser la stratégie mise en œuvre pour résoudre un problème proche ou une partie d'un plan construit précédemment qui correspond à un sous-problème et à des conditions similaires.

Pour mettre en œuvre ce type de réutilisation, nous avons choisi d'employer des techniques de Raisonnement à Partir de Cas (RàPC). Le RàPC résout un nouveau problème en retrouvant et en adaptant des solutions ou des éléments de solution d'un problème précédemment résolu. Il utilise un raisonnement assez proche du raisonnement humain pour pouvoir coopérer avec l'utilisateur et il permet de se focaliser sur plusieurs catégories d'informations concernant le plan, que ce soient des informations sur le problème posé, sur les images traitées ou sur la stratégie à adopter.

Le mémoire est composé de cinq chapitres.

Le premier chapitre est consacré à l'étude de l'Ingénierie des Connaissances (IC) et des systèmes de TI. Nous commençons par décrire des concepts et méthodologies proposés par l'IC pour la modélisation et l'acquisition de connaissances dans des domaines mal structurés tels que le TI. Nous mettons alors en évidence les caractéristiques que doit posséder un Système à Base de Connaissances pour faciliter l'acquisition interactive des connaissances et permettre leur opérationnalisation. Puis nous passons en revue un certain nombre de systèmes à base de connaissances dédiés aux TI dans le but de répertorier les connaissances intervenant dans une application et de particulariser les modèles nécessaires.

Le deuxième chapitre est entièrement consacré au RàPC. Nous montrons en quoi il peut améliorer l'aide apportée à l'utilisateur par notre système et nous décrivons les modèles et

techniques utilisés dans ce domaine. Cette description nous permet de comparer la représentation des cas et la mise en œuvre des différentes phases du RàPC sur différents systèmes et en particulier sur des systèmes de conception et de planification dont les objectifs se rapprochent des nôtres.

Dans le troisième chapitre, nous présentons notre système de conception et d'exécution interactives d'applications de TI. Nous commençons par exposer en détail les objectifs de ce système et les fonctionnalités nécessaires à sa mise en œuvre. Puis, nous décrivons l'architecture et le modèle utilisés. Ce modèle est basé sur les notions de tâches (représentation d'un but), de méthodes (représentation d'une stratégie) et d'outils (représentation d'un code informatique) et propose une représentation des problèmes à différents niveaux d'abstraction.

Le quatrième chapitre est consacré au module de RàPC. Nous montrons d'abord en quoi le RàPC est bien adapté à notre système et à nos besoins. Puis, nous spécifions la représentation choisie pour les cas, et expliquons en particulier comment ont été déterminés les critères de caractérisation d'un cas et leur utilisation. Enfin nous décrivons l'algorithme et les techniques adoptés pour la mise en œuvre des différentes phases du RàPC : sélection d'un cas, adaptation d'un cas et mémorisation d'un cas.

Le cinquième et dernier chapitre traite de l'implémentation et de l'expérimentation du système. En premier lieu, nous détaillons les principales classes définies ainsi que les différentes fonctionnalités proposées par l'interface graphique. Ensuite nous donnons des exemples de déroulement d'une session pour la création d'une application par définition interactive d'un nouveau plan, l'exécution interactive d'un plan sur une image et la création d'un nouveau plan par RàPC. La dernière partie de ce chapitre montre quelques exemples de plans de TI mis en œuvre à l'aide de notre système dans divers domaines d'application. Les expérimentations présentées nous permettent d'illustrer l'intérêt et de confirmer la validité de l'approche choisie.

I. Ingénierie des connaissances et systèmes de Traitement d'Images

I.1. Introduction

L'Ingénierie des Connaissances (IC) propose des concepts, méthodes et techniques permettant de modéliser et d'acquérir des connaissances dans des domaines ne se formalisant pas directement. Le Traitement d'Image (TI) est bien un tel domaine : le développement d'une application nécessite des connaissances détenues par plusieurs types d'experts et la connaissance de ces experts est principalement acquise par expérience, il n'y a pas de théorie notifiant le lien entre l'image initiale et le résultat à obtenir.

Pour appréhender la complexité du processus d'acquisition des connaissances, quel que soit le domaine visé, l'IC propose d'introduire des modèles qui ont pour objectif :

- d'une part, de jouer un rôle de médiateur en étant lisibles et compréhensibles par tous les acteurs du projet, avec en particulier des modèles du « niveau connaissance » [Newell 82],
- d'autre part, de faciliter le codage de la base de connaissances car on veut réaliser toutes les étapes de la spécification jusqu'à l'implémentation finale. Ces modèles doivent permettre de modéliser, de représenter puis d'opérationnaliser les connaissances en définissant des liens entre les différents niveaux de représentation.

Dans la deuxième section de ce chapitre, nous présentons une étude sur l'IC sans nous focaliser sur un domaine d'application particulier. Cette section concerne plus précisément les techniques d'acquisition et d'opérationnalisation des connaissances issues, soit directement d'un système, soit d'une méthodologie de développement de système.

Enfin la troisième section est une étude sur la représentation des connaissances dans les systèmes de TI. Cette section présente, suivant plusieurs points de vue, différents systèmes de TI, dédiés à un type d'images ou de problèmes ou non dédiés.

I.2. Etude sur les méthodes d'acquisition de connaissances

L'acquisition des connaissances se définit comme l'ensemble des processus nécessaires à la conception d'un SBC à partir de sources de connaissances. Ces sources peuvent être d'origine humaine ou documentaire. Dans cette thèse, nous nous intéressons essentiellement à l'acquisition de connaissances auprès d'experts humains.

Depuis le début des travaux sur les SBC, cette forme d'acquisition reste un problème difficile : l'expert d'un domaine n'est pas toujours en mesure de décrire et d'expliquer toutes les phases de son raisonnement, en particulier lorsque ce raisonnement relève plus de l'intuition et de l'expérience que d'un « savoir écrit » théorisé.

Les premiers travaux dans cette direction, réalisant un simple « transfert d'expertise », ont abouti à la création de systèmes experts. Cependant, ces systèmes proposent peu d'explication sur le raisonnement qu'ils utilisent et le manque de structuration des connaissances qu'ils détiennent permet difficilement la maintenance de la base. L'analyse de ces faiblesses a conduit à rechercher des techniques d'acquisition prenant plus en compte l'expert du domaine et à essayer de structurer les connaissances recueillies.

Un autre problème qui se pose lors de la construction d'un SBC est le passage des données recueillies auprès de l'expert à une forme implémentable. Pour pouvoir enrichir et modifier la base de connaissances d'un SBC, il est nécessaire que cette phase se fasse automatiquement sans passer par l'intervention d'un « traducteur humain ».

La construction de la base de connaissances passe par la définition d'un modèle des connaissances de l'expert, appelé Modèle Conceptuel (MC), qui doit prendre en compte toutes les contraintes énoncées ci-avant. Pour étudier la construction et l'opérationnalisation de ce modèle, N. Aussenac [Aussenac 92] propose un cadre méthodologique schématisé par la fig.1.

Il s'agit du cadre méthodologique d'acquisition et de modélisation des connaissances défini dans MACAO (Méthode d'Acquisition des Connaissances Assistée par Ordinateur) [Aussenac 89]. MACAO propose une démarche méthodologique pour le développement de SBC qui décompose le processus de modélisation en un ensemble d'étapes qui, par leur succession, guide la construction d'un modèle explicite de résolution de problème. La démarche proposée favorise l'analyse des processus de résolution de l'expert en décrivant les étapes de ces processus sous forme de tâches.

MACAO-II [Matta 95], la deuxième version de cette méthodologie, préconise l'association de la méthode MACAO à la réutilisation de modèles génériques. Elle propose des indications méthodologiques et la modélisation du raisonnement de l'expert par catégories de problèmes pour établir un lien entre des modèles génériques et le futur modèle de l'application. Pour représenter une méthode de résolution de problème, MACAO-II utilise le formalisme de représentation des connaissances conceptuelles MONA [Matta 95], à l'aide duquel les connaissances sont décrites sous forme de frames annotés par des descriptions en langage naturel et contenant un lien vers le niveau opérationnel.

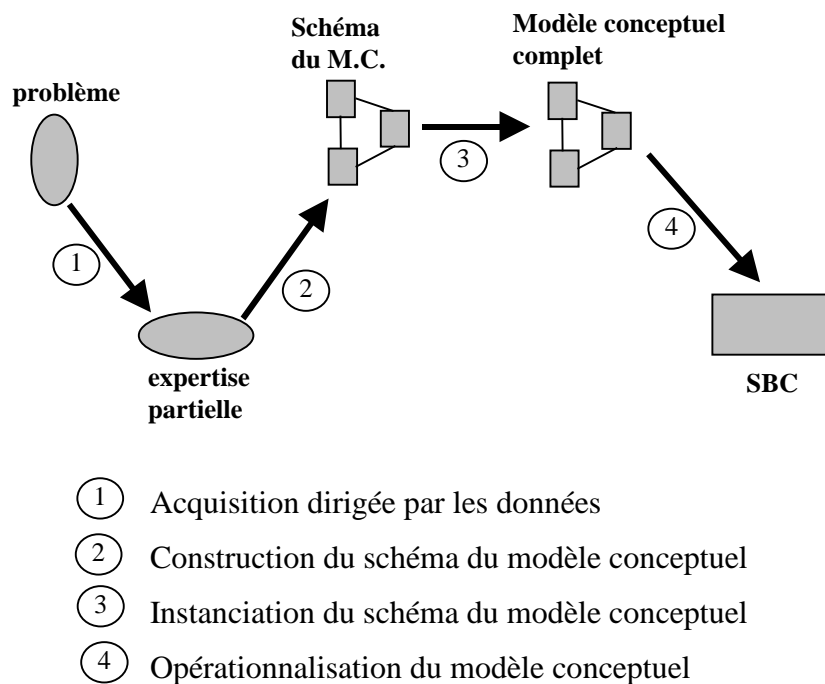


Fig.1 les différentes phases de l'acquisition des connaissances d'après Aussenac

Le système que nous proposons se veut être un outil d'expérimentation et d'acquisition interactive des connaissances. L'instanciation et l'opérationnalisation des connaissances (phase 3 et 4) doivent donc être accessibles à un utilisateur expert du domaine. Pour cela, d'une part, le Modèle Conceptuel complet doit être compréhensible par l'utilisateur et directement opérationnalisable. D'autre part, il faut déterminer les modules de contrôle nécessaires à la mise en œuvre de ces différentes phases en interaction avec l'utilisateur.

Dans la suite de ce chapitre, nous commençons par présenter plusieurs modèles d'acquisition des connaissances depuis des modèles théoriques vers des modèles de plus en plus tournés vers l'opérationnalisation. Puis nous nous intéressons plus particulièrement à l'intervention de l'utilisateur dans le contrôle de l'acquisition des connaissances et de leur opérationnalisation.

I.2.1. Modèles de représentation des connaissances

Generic Tasks

Chandrasekaran propose de construire les SBC à l'aide de blocs de base, qu'il appelle des « tâches génériques » [Chandrasekaran 87] [Chandrasekaran 94] [Chandrasekaran 96], chacun de ces blocs étant adapté à la résolution d'un type de problème. Pour cela, il part du principe qu'une tâche de raisonnement complexe d'un SBC peut souvent être décomposée en un certain nombre de tâches génériques ; chacune de ces tâches est associée à certains types de connaissances et à une famille de stratégies de contrôle mais est indépendante d'un domaine d'application particulier.

Une tâche générique est définie à l'aide des informations suivantes :

- la fonction de la tâche : le type de problème qu'elle résout, la nature de ses entrées et de ses sorties,
- la représentation et l'organisation des connaissances nécessaires à la réalisation de la tâche,
- la stratégie de contrôle qui doit être appliquée pour que la tâche réalise sa fonction.

A titre d'illustration, prenons l'exemple d'une tâche de diagnostic. Le but est d'expliquer les observations faites sur un système en termes de dysfonctionnements pouvant être à l'origine de ces observations. Ces dysfonctionnements peuvent interagir et sont organisés en hiérarchies. Une stratégie respectant les contraintes du domaine est définie pour la tâche de diagnostic. Cette stratégie décompose le problème en deux sous-problèmes : *classer hiérarchiquement les hypothèses* et *assembler abductivement les hypothèses*. Puis des tâches génériques résolvant chacun de ces deux sous-problèmes sont définies et ainsi jusqu'à l'obtention de buts auxquels on peut associer directement une procédure.

Six tâches génériques ont ainsi été définies pour résoudre des problèmes de diagnostic, de conception et de planification : *classification hiérarchique*, *synthèse d'objets par sélection et affinage de plans*, *transfert d'information dirigé par les connaissances*, *vérification d'hypothèse*, *assemblage abductif d'hypothèses* et *abstraction d'état*. Pour chacune de ces tâches, un outil gérant les connaissances et la résolution de problème a été mis au point. Ces outils permettent d'acquérir et d'organiser les connaissances de la tâche et de choisir des stratégies de résolution. Ils fournissent donc des modèles pour définir les différentes tâches génériques qui seront ensuite reliées pour créer le résolveur, mais ils ne permettent pas de réutiliser les tâches définies pour des applications précédentes.

KADS

KADS (Knowledge based system Analysis and Design Structure methodology) [Benjamins 92] [Wielinga 92] [Breuker 94a] [Schreiber 94] a été développé dans le cadre d'un projet de recherche européen ESPRIT. Il propose des méthodes et techniques qui couvrent tous les aspects de l'analyse et de la conception d'un SBC.

Le processus de conception y est décomposé en constructions successives de plusieurs modèles :

- le modèle organisationnel qui étudie les conséquences de la mise en place d'un SBC sur l'organisation et le travail des personnes,
- le modèle d'application qui définit la fonction du SBC dans l'organisation et les problèmes qu'il doit résoudre,
- le modèle de tâche qui précise comment la fonction du système est accomplie par la réalisation d'un ensemble de tâches,
- le modèle de la coopération qui contient les sous-tâches nécessitant un effort de coopération entre le système et l'utilisateur,
- le modèle d'expertise qui définit l'expertise de résolution de problème nécessaire à l'exécution des tâches de résolution de problème,
- le modèle conceptuel qui est une description abstraite des objets et des opérations que le système doit connaître ; il est défini par le modèle d'expertise et le modèle de coopération,
- le modèle d'implémentation qui décrit les techniques d'implémentation et de représentation nécessaires à la mise en œuvre du système.

Nous nous sommes intéressés plus particulièrement au « modèle conceptuel » qui est l'élément clé de la méthodologie KADS. Nous décrivons ici la partie « modèle d'expertise » du modèle conceptuel, la partie « modèle de coopération » fera l'objet du paragraphe sur le contrôle (§ I.2.2).

Pour modéliser l'expertise, KADS propose de structurer les connaissances suivant quatre niveaux, chaque niveau manipulant des objets du niveau inférieur (fig.2).


Couche	Relation	Objets manipulés
Stratégie		Plans, métarègles
Tâche		Tâches, termes de contrôle, structures de tâches
Inférence		Inférences, métaclases, vues du domaine
Domaine		Concepts, relations, propriétés

Fig.2 : structuration des connaissances dans KADS

Le niveau domaine conceptualise le domaine d'une application particulière sous la forme d'une théorie du domaine. Les connaissances du domaine sont définies à l'aide de concepts qui sont les objets centraux identifiés par leur nom, des propriétés de ces concepts (une propriété est définie par son nom et son domaine de valeur), des relations entre les concepts (relation « sous-classe » ou « partie-de »), des relations entre les propriétés (relations causales et temporelles) et des structures (représentation d'un objet complexe composé d'un ensemble de concepts et de relations).

Le niveau inférence caractérise les actions réalisables sur les objets du niveau domaine. Une inférence correspond à une primitive entièrement définie par son nom, la spécification de ses entrées/sorties et une référence au domaine qu'elle utilise. Une inférence primitive est décrite par une « source de connaissances » qui stipule l'action accomplie par la primitive sur les données d'entrées pour produire les sorties, par des « métaclases » qui décrivent le rôle que les objets manipulés jouent dans la résolution du problème et par une « vue du domaine » qui spécifie comment certaines parties de la théorie du domaine peuvent être utilisées par les sources de connaissances. Les inférences primitives peuvent être combinées en « structures d'inférences » définissant ainsi les enchaînements d'actions à effectuer dans un cadre donné pour atteindre un objectif précis. Un exemple d'inférence et de structure d'inférence concernant le diagnostic de panne dans des systèmes audio sont donnés figure 3 et figure 4.

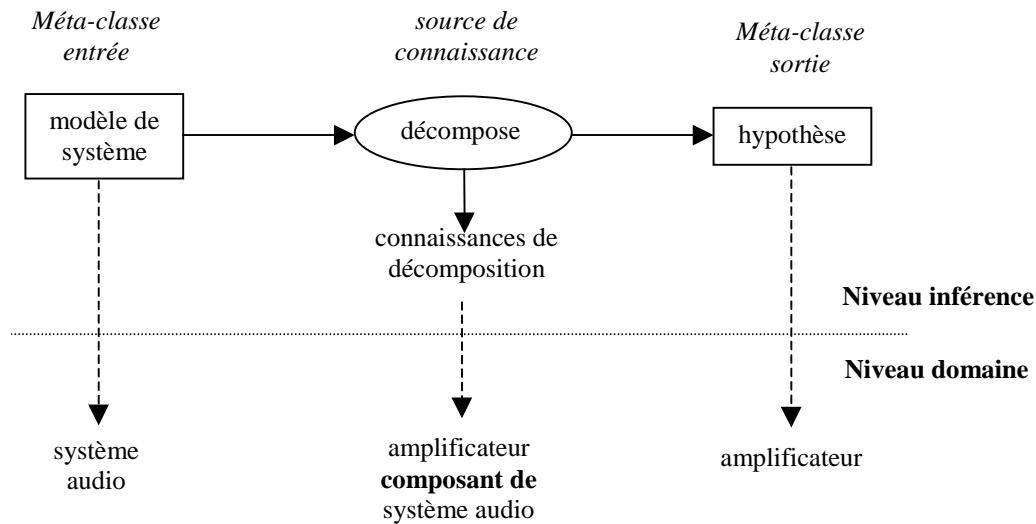


Fig. 3 : inférence primitive réalisant une action de décomposition.

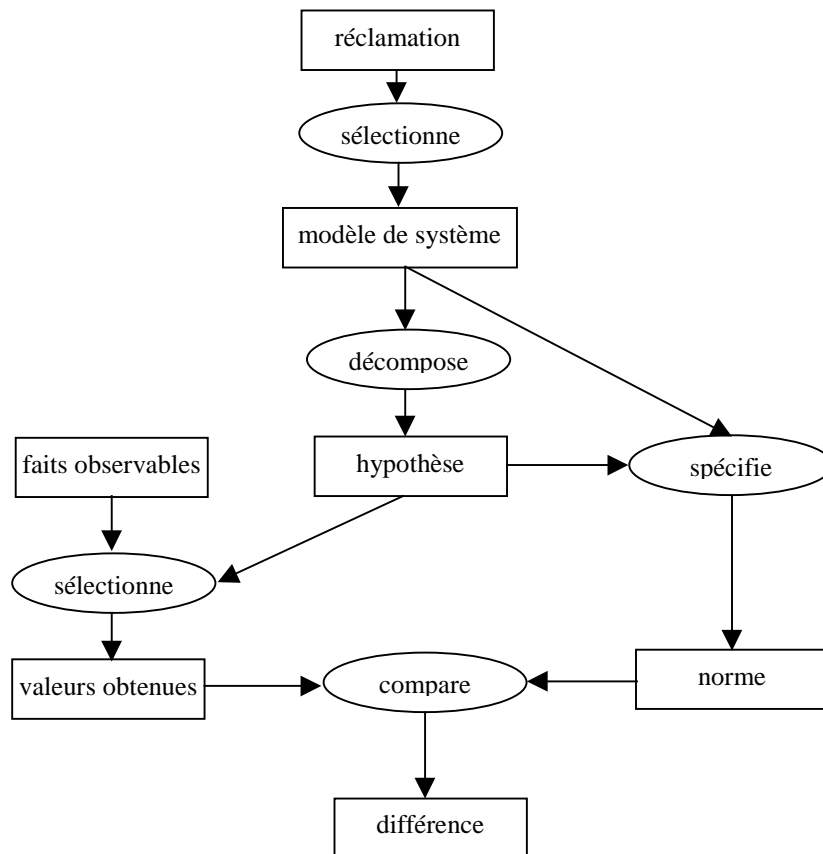


Fig. 4 : structure d'inférence pour le diagnostic de panne dans les systèmes audio.

Le niveau tâche décrit comment les inférences élémentaires peuvent être combinées pour atteindre un but donné. Ce niveau comporte plusieurs types de connaissances. Le premier type de connaissances est celui des tâches : une tâche correspond à un ensemble d'actions élémentaires, qu'elle décompose en sous-tâches. Le deuxième type de connaissances utilisé correspond aux termes de contrôle : il définit le vocabulaire utilisé, en particulier pour étiqueter les ensembles

d'éléments des méta-classes (ex. différenciation, focalisation). Le troisième et dernier type de connaissance utilisé est la structure de tâche : une structure de tâche spécifie une décomposition en sous-tâches et les relations de dépendances qui existent entre ces sous-tâches.

Enfin, le niveau stratégie contient les connaissances permettant de déterminer le but le plus approprié à la résolution d'un problème donné et de planifier dynamiquement les tâches à réaliser.

En partant du principe qu'une grande partie des informations contenues dans un modèle défini pour une application particulière sont réutilisables dans d'autres applications, KADS propose également de définir des modèles d'expertise réutilisables sur les niveaux stratégie, tâche et inférence. Il propose pour cela des ensembles de modèles d'expertise pré-définis servant de blocs de base et évitant les réinventions. Leur réutilisation est envisagée suivant deux axes : la typologie des sources de connaissances et les modèles d'interprétation. La typologie des sources de connaissances consiste en une classification des actions élémentaires de résolution de problème telles que « spécifier », « abstraire », « décomposer », « assembler », etc. Les modèles d'interprétation sont des modèles d'expertise partiels (abstraits du niveau domaine), qui permettent de guider l'acquisition des connaissances auprès de l'expert. Ils correspondent à des modèles d'expertise avec un niveau domaine vide, et sont donc indépendants du domaine d'application. La description des connaissances d'inférences et des connaissances de tâches y est réalisée à l'aide d'une terminologie indépendante du domaine d'application, ce qui permet leur réutilisation dans d'autres domaines.

SCARP

Dans SCARP (Systèmes Coopératifs d'Aide à la Résolution de Problèmes) [Willamowski 94a] [Willamowski 94 b], Willamowski propose de modéliser un problème sous la forme d'un plan hiérarchique, c'est à dire sur différents niveaux d'abstraction et de décomposition. Une base de connaissances SCARP (fig. 5) contient trois types d'éléments :

- les tâches modélisent les problèmes existants : une tâche définit un problème par un ensemble d'entrées et de sorties et lui associe une stratégie de résolution (comment la tâche peut être décomposée en tâches de plus en plus élémentaires),
- les méthodes définissent les inférences ou actions élémentaires dans lesquelles les tâches sont finalement décomposées : une méthode résout un problème élémentaire, soit par une inférence (méthode interne), soit par appel à un programme d'une bibliothèque (méthode externe),

- les entités représentent les objets manipulés dans le domaine d'application, en particulier ceux utilisés en entrée et en sortie des tâches et des méthodes.

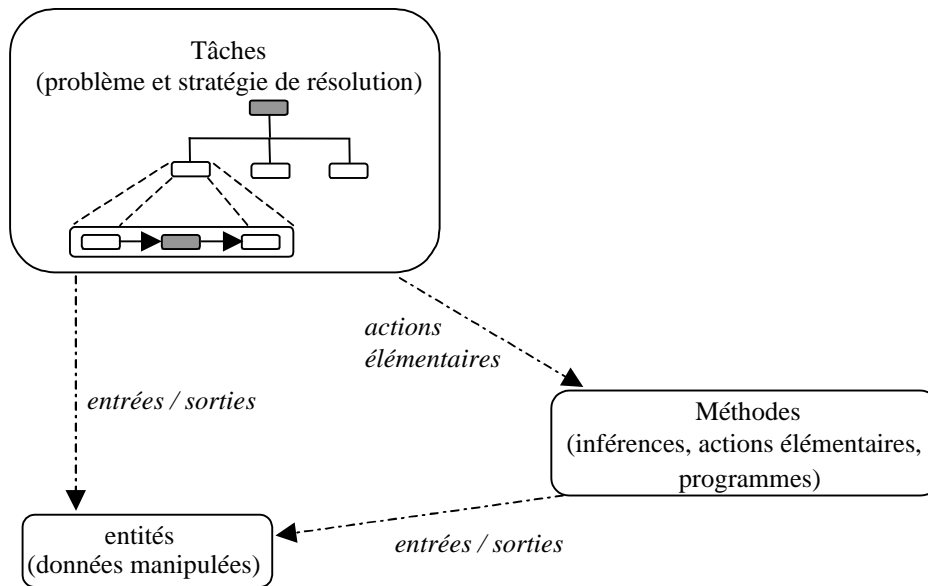


Fig.5 : éléments d'une base de connaissances SCARP

Toutes les connaissances contenues dans la base sont représentées à l'aide du modèle de représentation des connaissances par objets Shirka [Rechenman 90], qui est un modèle inspiré des frames. Dans Shirka, une classe ou une instance est représentée par un schéma regroupant un ensemble d'attributs. Un attribut est défini par une liste de facettes et une facette par une liste de valeurs (fig. 6).

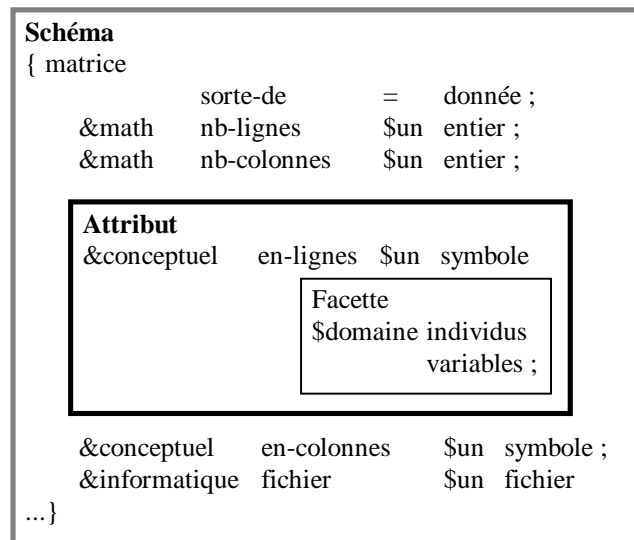


Fig.6 : schéma d'une classe ou d'une instance dans Shirka

Les classes d'objets sont organisées en hiérarchies de façon à ce qu'une hiérarchie regroupe les classes correspondant à un même concept. A chaque niveau dans la hiérarchie, le concept est décrit

à un niveau d'abstraction différent. Lorsqu'une nouvelle instance est créée, SCARP utilise des mécanismes de classification pour l'insérer dans une hiérarchie.

SCARP est un modèle opérationnel non dédié à un type de problème, il a été utilisé pour développer des systèmes dans différents domaines d'application : Anaïs est un système d'aide à l'interprétation de courbes de simulation pour l'analyse de systèmes dynamiques, Saïd un système de diagnostic vibratoire de plates-formes en mer, Myosis un système de diagnostic électromyographique et Slot un système d'aide à l'analyse de données.

LISA

LISA [Delouis 94] [Jacob-Delouis 96] est un langage réflexif de représentation des connaissances qui met l'accent sur l'opérationnalisation. Il permet de passer d'un modèle d'expertise (modèle papier structuré autour d'un certain nombre de concepts) à un système opérationnel, tout en conservant un lien entre les concepts du modèle d'expertise et les primitives d'implémentation. Il correspond donc à la mise en œuvre de l'étape 4 du schéma de Aussenac (fig. 1).

Ce langage propose un formalisme pour représenter les composants du modèle d'expertise, ainsi qu'un moteur utilisant ces connaissances pour résoudre des problèmes concrets. Il repose sur la définition des concepts suivants :

- les buts définissent l'ensemble des problèmes et sous-problèmes à résoudre ; ils sont caractérisés par des conditions de satisfaction et de terminaison (pour évaluer l'atteinte d'un but et la solution proposée) et possèdent un ensemble de méthodes et des règles de choix entre ces méthodes,
- les méthodes caractérisent l'ensemble des mécanismes permettant d'atteindre un but, une méthode peut être une séquence ordonnée de buts à réaliser, une procédure (appel à un code exécutable), une base de règles heuristiques (utilisée pour passer d'un type d'information à un autre) ou un appel à l'utilisateur,
- les connaissances du domaine décrivent les différents concepts manipulés par les méthodes ou utilisés dans la définition des buts. Parmi ces connaissances, LISA distingue les données relatives au cas en cours de traitement et les connaissances nécessaires à la création ou à la modification de ces données. Chaque concept du domaine est représenté par une classe et les exemples concrets de ce concept sont modélisés par des instances de cette classe,

- les connaissances de contrôle sont les connaissances nécessaires à l'exploitation des buts, méthodes et connaissances du domaine. Le contrôle est représenté à l'aide d'un modèle conceptuel équivalent à celui utilisé pour représenter le domaine, c'est à dire, sous forme de buts prédéfinis (recherche de méthodes, choix de méthodes, gestion de situation d'échec) et de méthodes disponibles pour atteindre ces buts (fig.7).

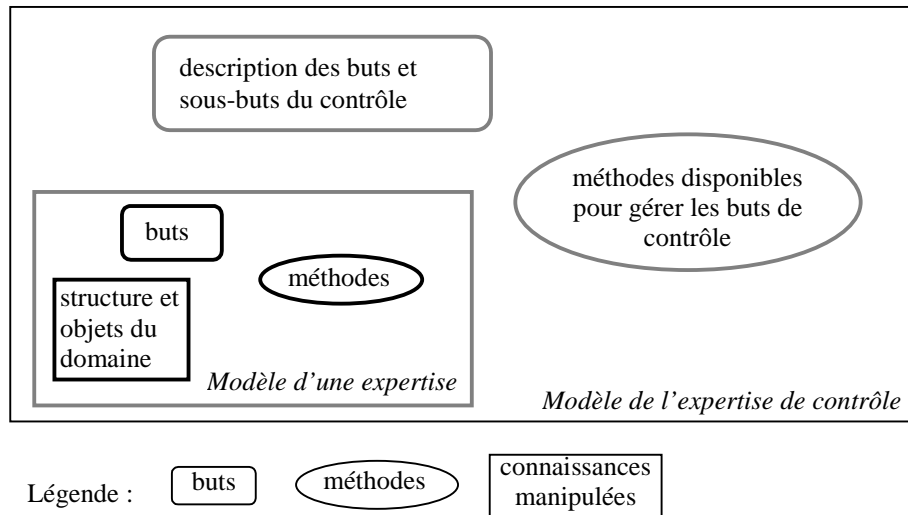


Fig.7 : organisation des connaissances dans LISA

Le langage LISA a été expérimenté au travers de deux projets : COPILOTE, un environnement d'aide à la planification de réseaux électriques et AUSTRAL, un système permettant de gérer le réapprovisionnement électrique du maximum de clients en cas de panne.

Bilan sur les modèles de représentation

Comme nous avons pu le voir dans les modèles présentés, l'acquisition de connaissances auprès d'un expert humain est facilitée par la structuration des connaissances à acquérir : en particulier, la séparation des connaissances portant sur les objets du domaine, sur le raisonnement et sur les actions ou les buts permet de guider la phase de modélisation.

Pour que les connaissances acquises puissent être utilisées en coopération avec l'utilisateur, d'une part, il est nécessaire d'introduire des connaissances sémantiques qui permettront au système d'expliquer son raisonnement. D'autre part, nous avons vu que la modélisation sur plusieurs niveaux d'abstraction facilite cette explication du raisonnement.

Enfin, dans des domaines tels que le TI, où les besoins ne sont mis en évidence que lors de la réalisation d'applications, l'acquisition de nouvelles connaissances doit pouvoir être réalisée interactivement, ce qui nécessite un modèle directement opérationnalisable. Ce type de modèle

facilite la maintenance de la base de connaissances en autorisant la définition d'une boucle permettant de modifier et de tester directement les connaissances acquises.

I.2.2. Intervention de l'utilisateur dans le contrôle de l'acquisition et de l'opérationnalisation

La mise en œuvre de l'acquisition des connaissances et de leur opérationnalisation passe par la définition d'un ensemble de tâches de contrôle. Pour réaliser un SBC, il faut donc définir les différentes tâches de contrôle nécessaires et déterminer celles dont la mise en œuvre est autorisée à l'utilisateur. Si l'on dispose de plusieurs catégories d'utilisateurs, on peut créer différents ensembles de tâches utilisateur correspondant chacun à une catégorie.

KADS propose de définir un modèle de la coopération entre le système et l'utilisateur. L'objectif de ce modèle est de décomposer les tâches réelles en un ensemble de tâches de base et de les répartir entre les agents intervenants. Ce modèle contient une spécification des fonctionnalités des tâches nécessitant un effort de coopération. Ces tâches concernent essentiellement des transferts de données et l'explicitation du raisonnement et peuvent agir dans le sens système vers utilisateur ou dans le sens inverse. Par exemple, dans une tâche de diagnostic audio, le système peut proposer que certains tests soient effectués par l'utilisateur ; l'utilisateur réalise alors ces tests et retourne les résultats au système. Inversement, l'utilisateur peut proposer volontairement une solution à un problème de diagnostic, le système critiquera alors la solution en la comparant aux siennes.

Dans LISA, l'implémentation des mécanismes de coopération se fait au travers de la définition de méthodes utilisateur (l'utilisateur construit les résultats pour le but courant à l'aide d'une interface adaptée) et de règles de préférence entre les méthodes système et les méthodes utilisateur. L'utilisateur peut traiter des buts du domaine (donner un résultat) mais il peut également traiter des buts de contrôle (choix d'une stratégie). Cependant, dans [Jacob-Delouis 96], Delouis précise que la réalisation de systèmes coopératifs nécessite la détermination de mécanismes de contrôle spécifiques (adaptation dynamique de la coopération) qui ne sont pas réalisables en LISA.

Avec KADS comme avec LISA, la répartition des tâches entre le système et l'utilisateur est figée. Une tâche donnée est allouée soit au système soit à l'utilisateur lors de sa création, mais cette attribution ne peut pas être modifiée dynamiquement pendant la résolution, comme cela pourrait être nécessaire suivant le contexte d'exécution ou suivant les désirs de l'utilisateur.

Dans SCARP, la coopération avec l'utilisateur est réalisée à l'aide de tâches de visualisation et d'opérateurs de description des stratégies de résolution. Les tâches de visualisation permettent de définir, pour chaque tâche, des moyens pour visualiser l'ensemble de ses entrées et de ses sorties. Les opérateurs de description de stratégie permettent de pré-définir les interactions système/utilisateur a priori nécessaires pendant la résolution. Les informations nécessaires au contrôle de l'exécution d'une tâche figure dans le contexte d'exécution. Il détient en particulier des moyens pour modéliser la coopération pendant la résolution (visualisation, demande de valeur, contrôle de la boucle d'exécution). L'utilisateur peut également remettre en cause toutes les décisions prises par le système et tous les résultats et définir de nouvelles tâches de façon interactive (définition des entrées, des sorties et de la stratégie de résolution adoptée).

Reprenant les travaux sur LISA, Monclar s'est intéressé plus particulièrement aux problèmes de coopération système/utilisateur avec la réalisation d'ELICO [Monclar 94] [Monclar 96], un environnement pour le développement de SBC coopératifs. Un SBC coopératif doit être capable de distribuer dynamiquement les tâches entre le système et l'utilisateur. Pour cela, il définit les notions de « rôle » et de « mode de coopération ». Un rôle attribué à un agent (système ou utilisateur) correspond à l'ensemble des problèmes que cet agent est capable de résoudre. Il propose ainsi quatre rôles principaux : décision (choix de résolution, validation, ...), exécution (des méthodes du domaine), assistance (conseils sur les choix et sur les validations) et critique (conseils sur les choix et les validations déjà effectués), un rôle pouvant ensuite être découpé en un ensemble de rôles plus fins. Le mode de coopération indique des règles d'attribution des différents rôles aux agents en tenant compte du contexte d'exécution. Par exemple, dans le système AUSTRALI dédié à la gestion de défauts dans les réseaux électriques, quatre modes de coopération ont été définis à l'aide d'ELICO : le mode « automatique » attribue tous les rôles au système, le mode « orienté utilisateur » attribue le rôle de décision à l'utilisateur et les autres au système, le mode « contrôle validation » attribue les rôles de validation et de décision à l'utilisateur et les autres au système, et le rôle « manuel » attribue tous les rôles à l'utilisateur.

Ces notions de rôles et de modes de coopération sont particulièrement intéressantes lorsque le système final doit servir à plusieurs types d'utilisateurs (utilisateur novice dans le domaine, utilisateur expert dans le domaine, concepteur du système, ...).

I.3. Représentation des connaissances dans les systèmes de TI

Une étude de plusieurs systèmes à base de connaissances en TI est présentée dans cette section. Deux points de vue sont étudiés, d'une part celui des connaissances mises en jeu, et d'autre part celui de la représentation de ces connaissances. Ceci nous a permis de déterminer les connaissances qui interviennent dans une application de TI et de définir les modèles nécessaires à leur acquisition.

Parmi les systèmes présentés, certains sont dédiés à un type d'image particulier tel que le système MVP (Multivision Vicar Planner) [Chien 96] [Chien 97] qui traite des images satellites de la Terre (correction des erreurs dues à la transmission et à la compression, combinaison de plusieurs images partielles, ...). D'autres sont dédiés à un type de traitement particulier tels que VSDE (Vision System Development Environment) [Theot 92] [Bodington 95] qui fait de la détection de défauts dans des composants ou VISIPLAN [Gong 95] qui détecte des objets en se basant sur des objets de référence. Cependant la plupart des systèmes étudiés sont non dédiés à un type d'image ou de traitement. La résolution d'un problème y est plus ou moins automatisée suivant qu'il s'agisse de systèmes de supervision de bibliothèques d'opérateurs tels que les planificateurs automatiques BORG (Blackboard orienté Objet pour la Résolution de problèmes par Génération de plans) [Clouard 94] [Clouard 95] [Clouard 98] et OCAPI (Outil de Contrôle Automatique de Procédures Images) [Clément 93] [Thonnat 93] [Thonnat 95] [Van den Elst 94] [Van den Elst 95] [Vincent 94] ou de systèmes d'aide à la conception d'applications tels que les systèmes ExTI-SATI [Dejean 96a] [Dejean 96b] et IMPRESS [Hasegawa 86]. Nous présentons également le projet BBI (BlackBoard Image) [Lefevre 93] [Lefevre 95] [Lefevre 96] dont l'objectif est la construction d'un système multi-agents pour l'interprétation d'images de différentes origines.

Ces différents systèmes sont décrits suivant plusieurs points de vue. Tout d'abord, nous parlons de la façon dont y sont représentées les connaissances, en distinguant les connaissances « stratégiques » des connaissances du domaine de l'image (pour les systèmes qui les prennent en compte) et des connaissances dites « statiques ». Les connaissances statiques regroupent les différents algorithmes de TI utilisés, le contexte des applications et le contexte des images, alors que les connaissances stratégiques décrivent le savoir-faire utilisé pour mettre au point une application particulière. Ensuite, nous décrivons comment ces systèmes résolvent des problèmes de TI en précisant comment est défini un problème, comment une solution est mise au point et comment sont

représentés les résultats. Enfin, nous nous intéressons au rôle que chacun de ces systèmes accorde à l'utilisateur. Tous ces aspects ne sont pas explicites dans tous les systèmes : nous avons choisi d'approfondir uniquement les points les plus intéressants de chaque système (certains systèmes ne font presque pas intervenir l'utilisateur) et notamment ceux qui peuvent être présentés de façon suffisamment indépendante du reste (certains systèmes ne distinguent pas les connaissances statiques des connaissances stratégiques).

Le tableau récapitulatif de la figure 8 sert de guide de lecture du chapitre : pour chacun des systèmes, il rappelle le domaine d'application ou les objectifs et l'ensemble des points de vue présentés dans la suite de ce chapitre.

Système	Domaine d'application/objectif	Points de vue présentés
MVP	Détection de défauts dans des composants industriels	STR, RP, RU
VSDE	Correction d'erreurs de transmission et combinaison d'images satellites de la Terre	STA, STR, RP, RU
VISIPLAN	Détection d'objets à l'aide d'objets de référence	STA, STR, RP, RU
BORG	Planification non dédiée à un type d'image ou de problème	STA, STR, RP, RU
OCAPI	Pilotage de programmes, non dédié à un type d'image ou de problème	STA, STR, RP, RU
ExTI-SATI	Aide à la conception d'application, non dédié à un type d'image ou de problème	STA, RP, RU
IMPRESS	Planification à partir du dessin des résultats attendus	RP, RU
BBI	Système multi-agents pour l'interprétation d'images	STA, STR, RP, RU
Légende : STA = représentation des connaissances statiques ; STR = représentation des connaissances stratégiques ; RP = résolution d'un problème; RU = rôle de l'utilisateur		

Fig.8 : tableau récapitulatif sur les systèmes de TI

I.3.1. Connaissances statiques en TI et connaissances du domaine de l'image

Pour réaliser un système de conception d'applications, on doit commencer par déterminer les connaissances nécessaires à la représentation de l'application elle même. Nous présentons ici la façon dont sont modélisées les différentes entités utilisées pour représenter une application dans les

systèmes étudiés. Une application comporte deux types de connaissances : les entités manipulées et les actions effectuées. En TI, les principales entités manipulées sont des images, mais il peut également s'agir des objets présents (ou recherchés) dans l'image ou des caractéristiques associées aux images (conditions d'acquisition, qualité de l'image) ou encore de caractéristiques calculées (à l'aide d'un histogramme par exemple). Les actions effectuées sur les images correspondent à des algorithmes dont la plupart sont connus et répertoriés dans des bibliothèques de programmes ou dans la littérature.

VSDE

Une hiérarchie de concepts est utilisée pour représenter les connaissances a priori sur le domaine d'application et les connaissances en TI. Cette hiérarchie de concepts est associée à un réseau sémantique dont les nœuds sont des concepts et les arcs des relations sémantiques entre ces concepts. Un nœud peut représenter un élément de la description du contexte de l'application ou une tâche de TI et possède un ensemble d'attributs le décrivant plus précisément. Il existe deux types de relation : les relations de type « partie-de » correspondant à la décomposition d'un concept en un ensemble de sous-concepts et les relations de type « spécialisation ». Cette hiérarchie regroupe l'ensemble de toutes les solutions possibles, une solution particulière correspondant à une instantiation de la hiérarchie.

Sur l'exemple présenté figure 9, la tâche « détection de défauts » est décomposée en deux tâches plus simples « réduction de bruit » et « élimination de défauts ». La tâche « réduction de bruit » est une tâche optionnelle et la tâche « élimination de défauts » se spécialise en « défauts niveau de gris » ou « défauts image texturée ». Lors de l'instanciation de cette hiérarchie, la tâche optionnelle n'a pas été conservée et un choix a été fait entre les deux spécialisations possibles.

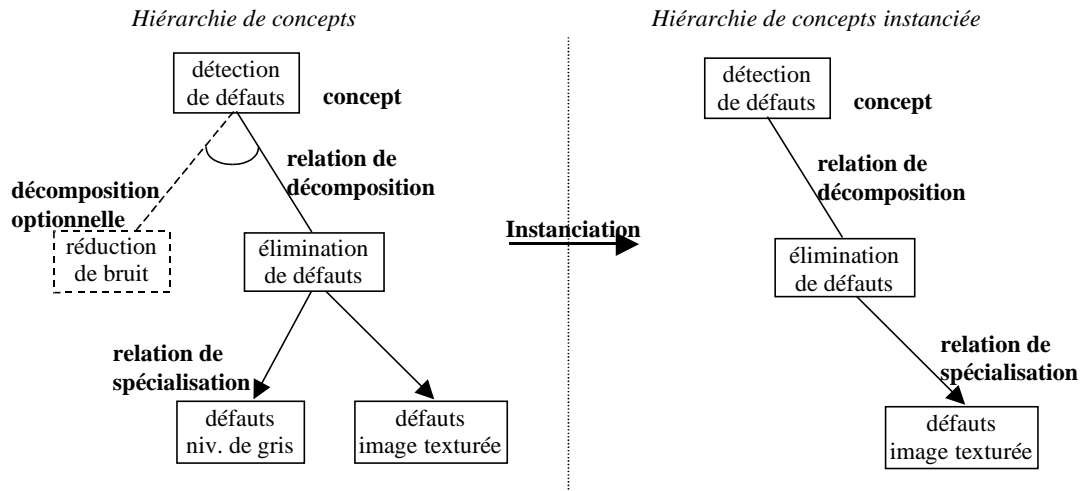


Fig.9 : hiérarchie de concepts dans VSDE

VISIPLAN

Pour détecter des objets dans des images à partir d'objets de référence, VISIPLAN utilise un « modèle de l'objet ». Ce modèle comprend :

- des connaissances sur les relations structurelles entre les objets et leurs composants : ce sont des relations « partie-de » qui vont guider la classification des objets recherchés,
- des connaissances sur les relations spatiales entre les objets ou leurs composants : ce sont des connaissances qui vont guider la décomposition en sous-buts,
- des connaissances sur les caractéristiques morphologiques et physiques des objets et de leurs composants, telles que la forme, la taille ou le contraste,

Ces connaissances définissent les modèles des objets recherchés et sont représentés sous la forme d'un réseau sémantique.

Dans ce réseau, les nœuds représentent des objets auxquels sont attachées des propriétés et les arcs les relations entre ces objets (fig. 10).

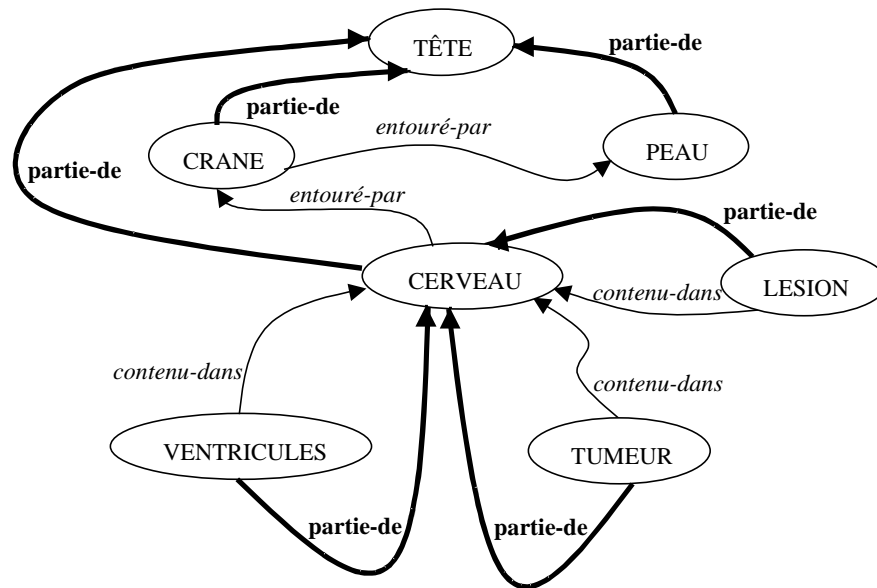


Fig.10 : réseau sémantique représentant un modèle d'objet dans VISIPLAN

OCAPI

Les connaissances descriptives d'OCAPI sont représentées sous forme d'objets structurés, elles correspondent à toutes les informations disponibles sur les programmes, les liens entre les différentes étapes, etc. Cette structuration est basée sur la définition des cinq concepts suivants (fig. 11) : but, opérateur, argument, contexte et requête.

- Un but représente une fonctionnalité du TI qui peut être réalisée par un algorithme ou un traitement plus complexe.
- Un opérateur représente une action qui peut être exécutée. Les programmes de TI sont des opérateurs élémentaires, les combinaisons d'opérateurs élémentaires forment des opérateurs complexes.
- Des arguments d'entrée et de sortie sont associés aux buts et aux opérateurs. Ces arguments définissent les données dont les valeurs sont fixes et les paramètres dont les valeurs peuvent être ajustées. Une description des données permet de conserver les informations sur les arguments eux-mêmes (taille de l'image, valeur maximum des pixels, ...) et sur leur contenu (objets présents dans l'image, ...).
- Un contexte d'utilisation fournit des informations symboliques concernant, soit les données (conditions d'acquisition, domaine de l'image, contenu de la scène, ...), soit l'environnement du système.

- Requête : une requête permet d'exprimer la demande de résolution de problème en précisant le but à atteindre, les objets à traiter et la qualité des résultats attendus.

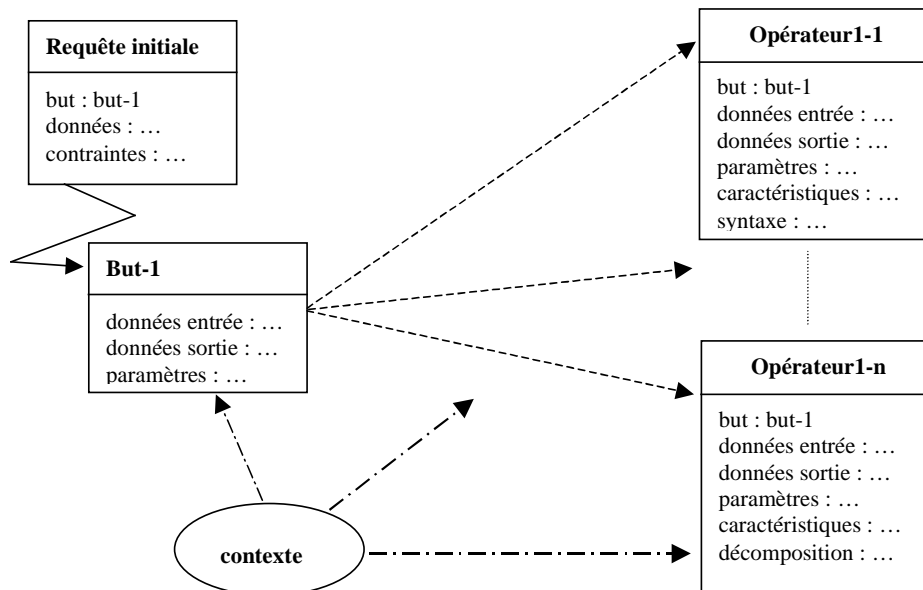


Fig.11 : relation entre les concepts d'OCAPI

BORG

Les informations se rapportant à l'image sont décrites dans le contexte de l'application sous forme d'un ensemble d'attributs / valeurs représenté à l'aide d'un langage de description de contexte (ensemble d'attributs symboliques prédéfinis). Le contexte est composé de trois niveaux de description :

- le niveau physique décrit la physique de formation de l'image et ses particularités techniques : en particulier, le type de capteur utilisé, les conditions d'acquisition, la nature du bruit, la qualité, ...
- le niveau perceptif décrit les indices visuels de l'image tels que le type d'information représentée (nuage de points, image de contours, ...), les caractéristiques des contours ou frontières entre objets, les caractéristiques des régions, ...
- le niveau sémantique traduit les informations attachées au concept d'objet pour le domaine d'application, il décrit les différents types d'objets possibles (taille, forme, couleur, ...) et leur organisation dans la scène (répartis, amassés, juxtaposés, ...).

Chaque action de TI est représentée indépendamment et est considérée comme une hypothèse affectée d'un coefficient d'importance et de vraisemblance qui la caractérise par rapport à l'application. Chaque action est décrite par :

- un but à atteindre : l'action qu'elle effectue,

- des contraintes sur le but : les exigences de qualité,
- des données d'entrée : les images d'entrée,
- des données de sortie : les images produites par l'exécution de l'action,
- une estimation de son importance par rapport à la solution globale,
- une estimation de sa fiabilité par rapport à la solution globale.

Les différentes actions composant le traitement d'une image sont reliées par deux types de liens : les relations de décomposition (d'une action en une combinaison d'actions) et les flux de données.

ExTI-SATI

L'approche utilisée dans ce système est entièrement basée sur la représentation fine de la donnée image. Un modèle permettant une description symbolique de la donnée image en terme d'indices visuels a été spécifié. Ce modèle définit une donnée image comme étant une structure composée des cinq éléments suivants (fig. 12) :

- la trame : c'est le support de la donnée, elle est de même dimension que l'image et délimite les coordonnées des points de celle-ci,
- le cache : c'est l'ensemble des points se trouvant dans les zones d'intérêt, il a la même dimension que la trame et est défini par une fonction cache à valeurs booléennes,
- l'index : il permet de regrouper les zones d'intérêt représentant un même concept par étiquetage des composantes du cache associé au concept, on peut ainsi considérer les concepts dans leur ensemble, indépendamment ou suivant certains critères,
- le signal : il représente la nature de l'intensité des points du cache reportés sur la donnée (niveau de gris, distances au bord, ...),
- le codage : c'est l'ensemble des valeurs que peuvent prendre les points du cache.

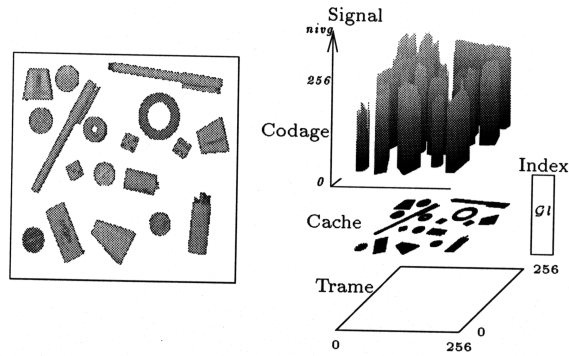


Fig.12 : représentation d'une donnée image dans ExTI-SATI

Pour manipuler ce modèle de donnée, un modèle d'opérateur a également été défini. Ce modèle décrit la transformation d'information réalisée sur chaque élément de la donnée et utilise les notions de structuration (regroupement d'informations), déstructuration (dissociation d'informations), spécialisation (ajout d'information), généralisation (retrait d'information) et nomination d'information.

BBI

BBI est un système multi-agents, les connaissances qui y sont représentées sont donc détenues par les agents. Chaque agent est défini de façon indépendante et est capable d'estimer ses capacités à résoudre un problème dans un contexte donné, de contrôler ses actions et d'évaluer ses propres résultats. Pour cela, chaque agent dispose de connaissances sur :

- le contexte de l'image défini sous forme d'une liste de couples (attribut / ensemble ou intervalle de valeurs) qui contient d'une part des attributs dits « précis » tels que la date ou les conditions météorologiques de la prise d'image, et des attributs dits « imprécis » ou « incertains » tels que le contraste ou le bruit,
- le problème décrivant le but à résoudre défini également sous la forme d'une liste de couples (attribut / valeur),
- les paramètres d'exécution qui sont les variables internes de l'agent et qui influencent son comportement.

Bilan sur l'étude des connaissances statiques

On peut distinguer deux principales approches dans les systèmes étudiés : la représentation par hiérarchies de concepts (modélisées sous forme de réseaux sémantiques) et la représentation par des modèles.

Les systèmes utilisant des hiérarchies de concepts ne différencient pas les concepts « action » des concepts « objet », ou ne représentent que l'un des deux. Ce type de représentation est bien adapté à des systèmes dédiés à un type d'image (ou à un type de traitement) dans lesquels l'ensemble des objets du domaine (ou des actions possibles) est connu et répertorié. Par contre, il permet difficilement la réutilisation et l'adaptation d'une solution à un nouveau problème.

Les systèmes utilisant des modèles comportent généralement plusieurs types de modèles et distinguent des modèles pour représenter les actions et des modèles pour représenter les entités manipulées. Ils permettent de s'abstraire plus facilement du domaine de l'image et donc de réutiliser les traitements pour des images de différentes origines. D'une part, la possibilité de représenter des traitements à plusieurs niveaux d'abstraction, et d'autre part la séparation des actions et des entités manipulées permettent la réutilisation et l'adaptation des applications en coopération avec l'utilisateur, ce dernier pouvant se focaliser sur la description des objets qu'il recherche ou sur la définition des traitements qu'il veut réaliser.

I.3.2. Connaissances stratégiques du TI

La plupart des systèmes que nous avons choisi d'étudier automatisent au moins une partie de la conception de l'application. Pour cela, ils utilisent des connaissances stratégiques qui représentent le savoir-faire du traiteur d'images. Ce dernier a acquis ces connaissances de façon entièrement expérimentale car il n'existe pas de théorie complète sur les stratégies de TI. Cela implique que les connaissances stratégiques intégrées dans tout système doivent être recueillies auprès des experts. Un système qui se veut évolutif doit donc posséder un modèle de la représentation des connaissances qui soit compréhensible à la fois par le système et par l'expert en TI ou disposer de moyens pour « traduire » ces connaissances d'un modèle humain à un modèle système.

Les connaissances stratégiques peuvent être examinées selon deux points de vue : d'une part, pour fournir des solutions opérationnalisables le système doit posséder des connaissances procédurales (les connaissances qui permettent de réaliser les traitements) ; et d'autre part, pour pouvoir faire appel à l'utilisateur ou pour pouvoir justifier ses résultats, le système doit posséder des connaissances sémantiques (les connaissances qui permettent d'explicitier les traitements).

VSDE

Les connaissances stratégiques de VSDE ont pour but d'instancier la hiérarchie de concepts pour en extraire une solution unique adaptée au contexte de l'application à résoudre. Ces connaissances

sont modélisées sous forme de règles de production qui utilisent des connaissances sur les algorithmes de TI pour choisir les meilleurs en fonction des données et des résultats attendus. Ces règles sont représentées sous la forme d'un ensemble de conditions et d'actions. Les conditions portent sur la présence ou l'absence de faits générés par la réponse d'une question à l'utilisateur ou par le système lui-même (caractéristiques mesurées sur l'image, faits déduits de l'application d'une règle précédente). Les actions correspondent à des opérations telles que :

- sélectionner une spécialisation ou une décomposition,
- éliminer une spécialisation ou une décomposition,
- proposer un réglage des paramètres,
- restreindre un réglage des paramètres,
- dialoguer avec l'utilisateur pour lever une ambiguïté, ...

MVP

MVP utilise deux planificateurs (un planificateur basé sur la décomposition et un planificateur basé sur les opérateurs) qui effectuent chacun une partie de la résolution et qui utilisent donc des connaissances stratégiques différentes.

Le planificateur basé sur la décomposition utilise des règles de décomposition pour réaliser deux types d'opération :

- la « planification schématique » qui consiste en la classification du problème en fonction des buts et de l'état initial,
- la « planification hiérarchique » qui effectue une décomposition des buts en sous-buts de plus en plus spécifiques en fonction de la classification du problème.

Le planificateur basé sur les opérateurs utilise des modèles d'action pour atteindre un but à partir d'un état initial. Les actions sont des traitements d'images de base (correspondant à l'exécution d'un opérateur), l'état initial est l'état de l'image initiale et le but est le traitement recherché. Une action est représentée en terme de ses pré-conditions et de ses effets.

OCAPI

Les connaissances stratégiques d'OCAPI sont modélisées sous forme de règles de production permettant l'automatisation de tout le processus de résolution de problème par une adaptation dynamique à la situation courante. Ces règles sont spécialisées selon quatre types :

- règles de choix : elles sont utilisées pour choisir l'opérateur le plus pertinent parmi tous ceux possibles, en fonction de la description des données, du contexte de l'application et des caractéristiques des opérateurs,
- règles d'évaluation : elles exposent comment accéder aux résultats produits par les opérateurs choisis et permettent de déterminer si ces résultats sont satisfaisants,
- règles d'initialisation : elles permettent de définir les valeurs d'entrées des paramètres d'un opérateur, en fonction du contexte et de la requête en cours de traitement,
- règles d'ajustement : elles expliquent comment modifier les valeurs des paramètres d'entrée lorsque les résultats d'un opérateur ne sont pas satisfaisants.

Les deux premiers types de règles sont attachés aux buts (un but sait choisir un opérateur et évaluer ses résultats) et les deux derniers sont attachés aux opérateurs (un opérateur sait initialiser et ajuster ses paramètres).

VISIPLAN

Le générateur automatique de plans de VISIPLAN cherche à résoudre une suite de sous-buts (définis chacun par un objet de référence recherché) qui correspond à une stratégie générale d'analyse des images. Pour réaliser la détection d'un objet, il dispose d'un ensemble d'images de la même scène (ex. : images IRM prises avec des réglages machines différents).

Le planificateur utilise une stratégie générale en cinq phases pour résoudre chacun des sous-buts :

- sélectionner, parmi l'ensemble d'images, la ou les images à utiliser, en fonction des caractéristiques principales de l'objet recherché (intensité, contours, ...),
- sélectionner le focus d'attention en utilisant la définition des objets de référence et les relations spatiales définies entre ces objets,
- sélectionner un opérateur de focalisation (accès direct, masquage direct ou projection interplans) en fonction du focus d'attention,

- sélectionner une méthode de segmentation grâce aux critères morphologiques de l'objet (régions, contours, ...), aux critères discriminants de l'objet (par rapport à une classe de méthodes) et aux critères de complexité des méthodes (lorsque plusieurs méthodes sont candidates),
- sélectionner un schéma de reconnaissance en fonction des caractéristiques morphologiques de l'objet.

A chacune de ces étapes, la stratégie de sélection est modélisée sous forme de règles de production.

BBI

Dans BBI, les connaissances stratégiques sont détenues par des agents. Il existe 30 agents répartis en 4 niveaux d'abstraction :

- le niveau des objets de la scène,
- le niveau des primitives images (contours, régions, ...),
- le niveau des objets caractérisés (cercle, rectangle, ...),
- le niveau pixel.

Pour permettre à chaque agent d'estimer ses capacités à résoudre un problème dans un contexte donné, BBI utilise un formalisme inspiré de la théorie de Dempster-Shafer. Cette théorie permet d'attribuer à un élément ou à un ensemble d'éléments deux valeurs qui représentent sa plausibilité (degré d'incertitude) et sa crédibilité (degré de certitude).

La compétence d'un agent peut ainsi être définie par l'attribution de ces deux grandeurs à des éléments ou à des ensembles d'éléments de son contexte.

Le modèle d'agent est inspiré du modèle du Black-Board (Tableau Noir) : chaque agent est formé d'une entité « action » (exécution d'un traitement, interrogation d'une base de données, ...), d'une entité « contrôle » (contrôle d'exécution et évaluation des résultats), d'une entité « communication » avec les autres agents (émettre ou satisfaire des requêtes) et d'une entité « archivage » (ses résultats, son contexte).

BORG

Le système BORG, basé sur l'architecture de Tableau Noir à contrôle par Tableau Noir BB1 [Hayes-Roth 85], s'appuie sur la représentation explicite des expertises de TI et des stratégies de

contrôle de la résolution sous la forme de Sources de Connaissances. Une Source de Connaissances est définie par un ensemble de conditions et de règles de production. Les conditions se décomposent en deux types : les déclencheurs qui ont pour rôle la détection des événements (création ou modification d'une action sur le graphe solution, ...) et les préconditions qui ont pour but la vérification de l'état courant de la résolution. Les règles de production forment la partie action de la Source de Connaissances, les actions consistant à créer ou à modifier des actions sur le graphe solution.

Ces Sources de Connaissances sont classifiées, selon leurs actions, en cinq types :

- construire le plan solution,
- instancier le plan solution avec des opérateurs de TI,
- exécuter les opérateurs de TI,
- remonter les résultats intermédiaires vers les niveaux supérieurs,
- évaluer les résultats intermédiaires.

Bilan sur l'étude des connaissances stratégiques

On notera tout d'abord que la classification des connaissances stratégiques en différents niveaux d'abstraction permet de réduire considérablement l'espace de recherche à chaque étape de la résolution.

La plupart des systèmes étudiés modélisent une partie de leurs connaissances stratégiques sous forme de règles de production, ces règles étant représentées soit directement dans une base de règles, soit à l'intérieur d'un modèle plus complexe tel celui des « sources de connaissances ». La partie « condition » utilise les connaissances statiques (connaissances générales en TI) et les connaissances sur l'application (description précise des objets recherchés et des images traitées). La partie « action » de ces règles correspond à une transformation du problème (décomposition en sous buts ou spécialisation) ou à la résolution d'un problème de base quand il existe un algorithme. Pour permettre à l'utilisateur de visualiser et de comprendre les connaissances stratégiques mises en œuvre dans une application, il est nécessaire de mettre en avant les connaissances sémantiques détenues dans ces règles par leur représentation au sein de l'application résultat.

Cependant une base de règles représentant les stratégies du TI est particulièrement difficile à construire. D'une part, un expert en TI travaille en se fiant plus à son intuition et son expérience qu'en mettant en œuvre un savoir préétabli. Il n'est donc pas toujours capable de détailler les

connaissances stratégiques qu'il utilise sous forme d'éléments suffisamment simples pour être modélisables sous forme de règles. D'autre part, la maintenance d'une base de règles est une tâche délicate. Il se pose en particulier le problème de savoir qui maintient cette base : l'expert en TI n'est pas toujours capable de gérer les problèmes d'implémentation et le concepteur du système ne peut pas toujours être à sa disposition. Un autre problème est celui de la taille et de la cohérence de la base : trop de règles peut entraîner un effet « usine à gaz » et la suppression de certains éléments peut conduire à des incohérences.

I.3.3. Résolution de problèmes de TI

La résolution d'un problème peut être décomposée en trois étapes principales : sa définition, sa résolution proprement dite et la présentation des résultats.

La définition du problème est généralement faite par l'utilisateur, mais elle peut également être réalisée par un module informatique dans le cas où le traitement de l'image n'est qu'une partie d'un système plus global. Le problème peut être défini entièrement a priori ou au fur et à mesure des besoins apparaissant pendant la résolution.

La résolution du problème proprement dite correspond à la mise en œuvre des connaissances stratégiques.

La présentation des résultats dépend de la représentation des connaissances statiques et stratégiques, des techniques de résolution utilisées et du niveau de coopération système/utilisateur.

Dans la suite de cette section, nous détaillons ces trois étapes pour les systèmes étudiés.

I.3.3.1 Définition d'un problème

La définition d'un problème peut être basée sur une description complète des actions à effectuer ou sur une description précise des objets à détecter. Définir précisément le problème à résoudre est un des points les plus délicats en TI. Pour pallier cette difficulté, certains systèmes proposent une aide à l'utilisateur.

Dans les systèmes dédiés à un type de problème particulier, tel que la détection de défauts pour VSDE ou la détection d'objets particuliers pour VISIPLAN, la description du problème est axée sur une description précise du contexte de l'image et des objets recherchés dans l'image en termes d'objets du domaine. Dans VSDE, un problème correspond à la description de l'environnement (capteur, lumière, ...), à un modèle de l'objet, à un ensemble d'images test (modèle des défauts) et à

la définition de la tâche d'inspection (mesures de tolérance géométrique, marquage d'un défaut, ...). Dans VISIPLAN, un problème comprend la définition précise des objets ou des composants d'objets à détecter (caractéristiques morphologiques, relations spatiales, ...), un ensemble d'images et d'objets de référence.

Au contraire, dans les systèmes non dédiés, la définition du problème doit comprendre une description précise de la tâche à accomplir en termes d'objectifs du TI, en plus du « contexte de l'image ». Si des précisions sur l'image ou les objets sont nécessaires pour l'établissement de la solution, ils seront calculés ou acquis interactivement pendant la résolution. C'est le cas dans BORG, dans OCAPI et dans BBI où la requête initiale est définie par un but et par des contraintes précisant ce but et par un contexte (conditions d'acquisition de l'image, domaine d'application, ...).

Certains systèmes, dont l'objectif est de faciliter l'expression des problèmes de TI, présentent des approches plus originales. Le système IMPRESS propose à l'utilisateur de représenter son but à l'aide de modèles binaires de surfaces, lignes et points qu'il peut définir en dessinant ce qu'il recherche sur l'image. Le système ExTI-SATI utilise un langage de description de concepts pour la formulation d'objectifs : un objectif est une donnée particulière définie à l'aide d'indices visuels.

I.3.3.2 Mise en œuvre des connaissances stratégiques

Les techniques de mise en œuvre des connaissances stratégiques dépendent essentiellement du niveau d'automatisation de la construction de l'application et de la façon dont sont modélisées ces connaissances. Dans ce paragraphe, nous présentons pour chacun des systèmes étudiés, l'architecture ou l'algorithme qui gère la construction et/ou l'exécution d'une application.

VSDE

La configuration d'une solution à un problème d'inspection est réalisée par un processus itératif de décomposition et de simplification du problème posé, alimenté par des connaissances a priori sur le domaine de l'inspection et sur le TI via un dialogue avec le spécialiste de l'application. Nous rappelons ici que les connaissances de VSDE sont représentées par une hiérarchie de concepts et qu'une solution correspond à une instanciation de cette hiérarchie. La résolution de problème se déroule en trois principales étapes :

- définition du problème par un dialogue avec l'utilisateur construit sur une série de questions ordonnées dynamiquement par un ensemble de règles et par la hiérarchie de concepts,
- configuration de la solution par itération des quatre étapes suivantes :

- ♦ instantiation en profondeur d'abord de la hiérarchie de concepts
- ♦ élagage par les types de données
- ♦ évaluation intermédiaire permettant de guider la suite de la résolution
- ♦ décomposition ou spécialisation du concept instancié
- évaluation de la solution sur un ensemble d'images : si la solution échoue sur certaines images, on relance le système sur ces images.

MVP

Pour générer un « script » de TI, MVP utilise deux paradigmes de programmation : la planification par décomposition et la planification basée sur les opérateurs. Le développement d'une application (fig. 13) est réalisé par l'enchaînement des étapes suivantes :

- spécification du problème par l'utilisateur (but et informations sur l'image),
- « planification schématique » : le planificateur basé sur la décomposition classe le type de problème en utilisant les connaissances sur le but et l'état initial,
- « planification hiérarchique » : les buts abstraits du schéma de plan sont affinés en buts de plus en plus spécifiques à l'aide des règles de décomposition,
- résolution des sous-problèmes par le planificateur basé sur les opérateurs en fonction du sous-but recherché et de la compatibilité avec les autres parties du plans,
- assemblage des parties de plans obtenues en respectant les contraintes générées par la décomposition,
- génération d'un script exécutable.

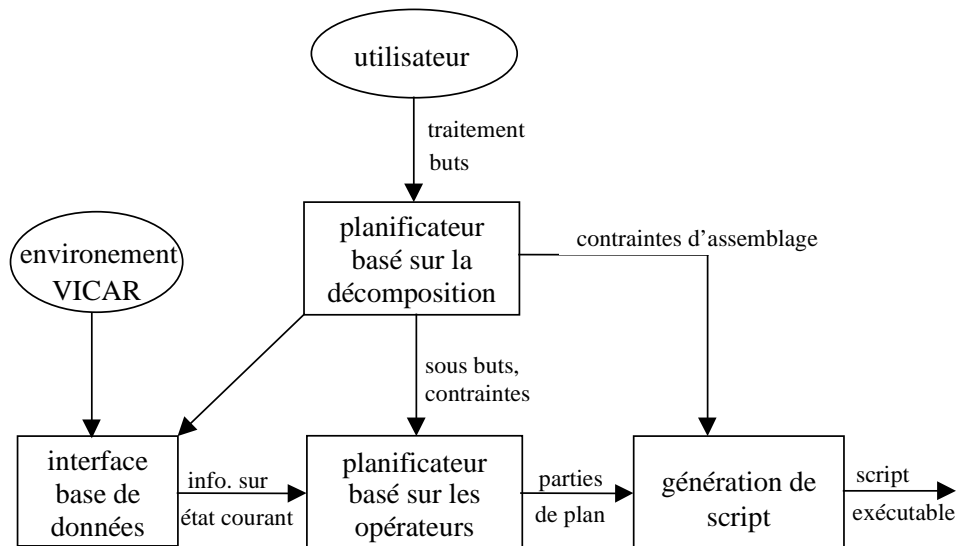


Fig.13 : architecture de MVP

ExTI-SATI

ExTI-SATI est un planificateur dans un espace d'états, basé sur un mécanisme de propagation qui rend compte de la modification d'information effectuée. Un état correspond à la description d'une image en termes d'indices visuels réalisée à l'aide d'un langage d'expression d'objectifs. Pour configurer un enchaînement d'opérateurs, le planificateur possède les quatre fonctions suivantes :

- acquisition de l'expression d'un but : ce but est traduit en une expression sous forme canonique à l'aide de règles transformationnelles,
- comparaison de la situation courante à ce but : si les deux expressions sont différentes, la progression vers le but s'effectue selon deux axes :
 - ♦ généralisation - spécialisation : si le concept courant est trop général par rapport à l'objectif,
 - ♦ composition structurelle : si le concept courant est un élément ou un sur-ensemble de l'objectif,
- choix de l'opérateur pertinent pour réaliser la progression souhaitée : il consiste en la sélection des opérateurs applicables, l'élimination de ceux qui ne vérifient pas la cohérence des domaines, la sélection d'un opérateur et la conservation de l'ensemble des opérateurs applicables,
- application de l'opérateur : le planificateur génère l'expression qui sera produite par l'exécution de cet opérateur. Si elle contribue à la résolution, l'opérateur est inclus dans le plan et fournit la nouvelle expression courante.

A partir du graphe d'opérateurs symboliques, le système génère un script exécutable qui utilise les opérateurs réels d'une bibliothèque de TI.

OCAPI

Pour planifier et exécuter les programmes d'une bibliothèque de TI, OCAPI utilise un moteur indépendant de toute application. Ce moteur permet de déterminer un plan (séquence hiérarchique d'opérateurs) approprié à un problème et de contrôler l'exécution de ce plan. Il dispose de plusieurs modules de base (fig. 14) : un planificateur (sélection et ordonnancement des opérateurs), un contrôleur d'exécution, une interface avec la bibliothèque d'opérateurs et un interpréteur de règles.

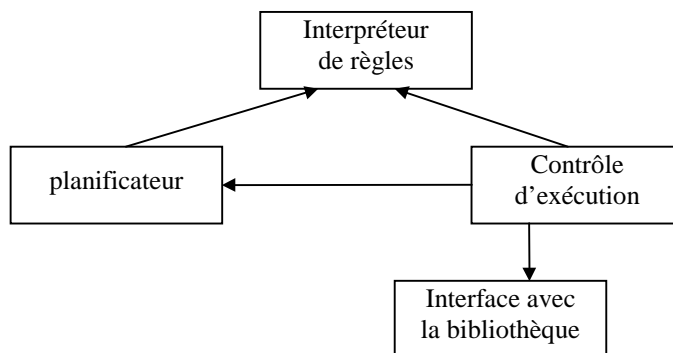


Fig.14 : graphe d'appel des modules

Le fonctionnement du moteur est basé sur la boucle d'exécution suivante (qui prend en entrée une requête ou un arbre « et/puis » de requêtes) :

1-s'il existe au moins un opérateur dont les arguments sont compatibles avec les requêtes

tant que toutes les requêtes ne sont pas traitées :

2 - sélectionner la requête à traiter (parcours de l'arbre en profondeur d'abord)

3 - classer les opérateurs (en utilisant les règles de choix)

tant que la requête n'est pas satisfaite :

4 - sélectionner le meilleur opérateur

5 - paramétrer l'opérateur (en utilisant les règles d'initialisation ou d'ajustement)

6 - exécuter l'opérateur (appel récursif dans le cas d'un opérateur complexe)

7 - évaluer les résultats (en utilisant les règles d'évaluation)

VISIPLAN

Pour générer des plans de détection d'objets, VISIPLAN (fig. 15) utilise un planificateur hiérarchique orienté objet, s'appuyant sur la définition des objets à détecter, de leurs caractéristiques morphologiques et de leurs relations. Il est composé de deux modules principaux :

- le pré-planificateur : c'est un outil interactif avec lequel l'expert sélectionne les opérateurs de pré-traitement et détermine leurs valeurs de paramètres. Il guide également l'utilisateur dans la définition des objets à détecter et de leurs relations.
- le générateur de plans proprement dit : c'est un interpréteur de règles qui exécute les actions des règles dont les pré-conditions sont satisfaites. Les règles de planification utilisent les sources de connaissances et les modèles acquis interactivement auprès de l'expert avant le début de la planification.

Lorsqu'un plan complet a été généré, chaque élément du plan est traduit en un appel à une procédure ou à une fonction par l'interpréteur de plan, puis est exécuté. Le plan est alors évalué : s'il est satisfaisant, il est sauvegardé, sinon il est révisé. La révision est guidée par une analyse des différents résultats intermédiaires réalisée par l'expert.

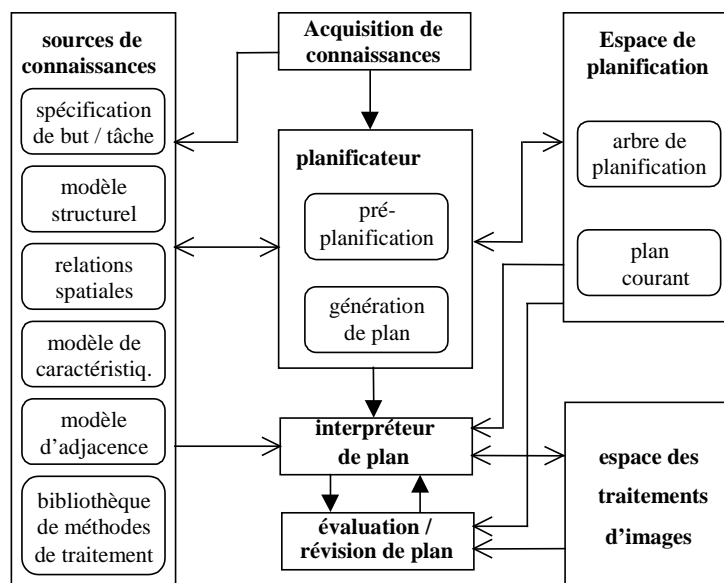


Fig.15 : architecture du système VISIPLAN

IMPRESS

Contrairement aux autres systèmes qui construisent la solution en partant d'un problème donné, le système IMPRESS procède en partant de l'image solution. La mise au point d'un traitement y est réalisée en quatre étapes (fig. 16) :

- prétraitement d'une image exemple (dessin fait par l'expert).
- sélection d'un « squelette de procédure » en fonction de l'image exemple : un squelette de procédure est une suite de modules de traitement d'image (lissage, différenciation, binarisation, ...). Ce squelette est déterminé en fonction du type de caractéristiques à extraire.
- construction d'une procédure détaillée à partir du squelette :
 - ♦ « estimation en sens inverse » d'une image d'entrée possible pour chaque module de la procédure par exécution de la fonctionnalité inverse du module.
 - ♦ « construction avant » : construction d'un enchaînement avec choix des opérateurs dont les résultats sont les plus proches des images intermédiaires calculées lors de « l'estimation en sens inverse ».
- évaluation des résultats de la procédure par mesure de similarité entre l'image obtenue et l'image exemple.

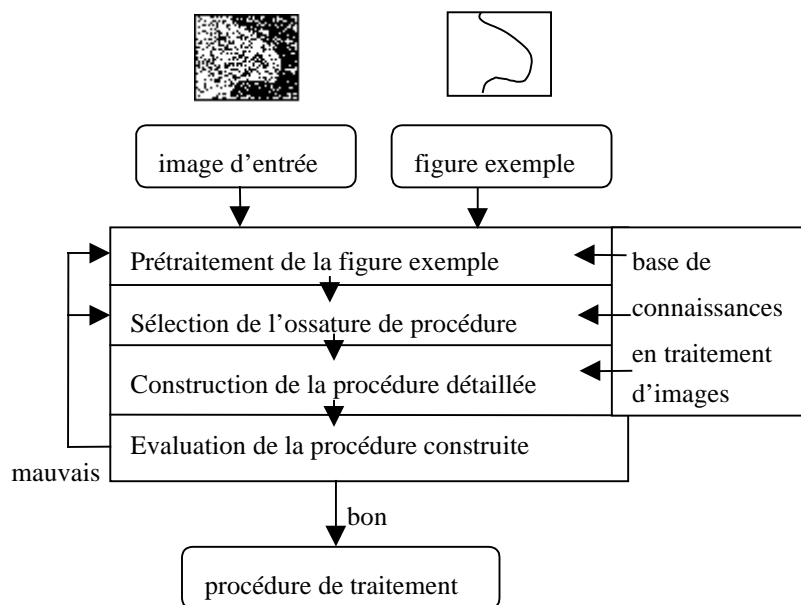


Fig.16 architecture du système IMPRESS

BBI

Dans le système BBI, la résolution d'un problème est effectuée par l'intermédiaire d'un protocole de communication entre agents, de type « appel d'offre », qui se décompose en six étapes :

- envoi de requête : un problème est posé par un agent qui le décompose en plusieurs sous-buts ; pour chaque sous-but, il envoie une requête (le problème et son contexte) aux autres agents.

- réponse des récepteurs : les agents traitent la requête, évaluent leur capacité à résoudre le problème et renvoient leur score d'aptitude estimé.
- supervision : l'émetteur de la requête sélectionne les agents les plus aptes et leur ordonne de résoudre le problème.
- exécution : chaque agent concerné résout le problème posé en paramétrant ses traitements, puis il évalue ses résultats (s'ils sont insuffisants, il relance la résolution en adaptant les paramètres), enfin il retourne un compte-rendu de ses résultats ainsi que leur évaluation.
- interprétation : l'émetteur demande les résultats qui l'intéressent ou une nouvelle exécution.
- livraison : les agents concernés envoient les données souhaitées à l'émetteur.

BORG

Face à un problème de TI, BORG va construire dynamiquement un plan solution par planification hiérarchique. Le plan est construit, selon cinq niveaux d'abstraction (requête, objectif, fonctionnalité, procédure, opérateur), par affinages successifs des buts d'un niveau en sous-buts du niveau inférieur. La planification procède en trois étapes :

- explicitation de la requête en termes de tâches de TI : traduction, reformulation et élimination des ambiguïtés,
- choix d'une démarche pour résoudre chacune de ces tâches, en fonction des spécifications de la tâche, du contexte de l'application et des caractéristiques de l'image d'entrée, chaque tâche est résolue en enchaînant plusieurs fonctionnalités,
- choix des moyens : sélection des procédures à enchaîner pour résoudre chaque fonctionnalité.

Le plan est ensuite instancié par décomposition de chaque procédure en un enchaînement d'opérateurs dont le système doit déterminer les valeurs des paramètres. Puis il est exécuté, c'est-à-dire que le système applique chacun des opérateurs dans l'ordre décrit par le plan en optimisant leurs valeurs de paramètres.

Le plan est alors évalué en remontant les résultats des opérateurs vers les niveaux supérieurs, chaque niveau ayant ses propres critères d'évaluation. Si un but est jugé non satisfait, le plan est corrigé en envisageant une autre décomposition.

Bilan sur la mise en œuvre des connaissances stratégiques

En dépit d'une grande diversité dans la représentation des connaissances stratégiques (règles de production, sources de connaissances, agents logiciels, ...) et dans les techniques de résolutions (dirigées par les buts, dirigées par les données), on notera tout de même quelques points communs à presque tous ces systèmes :

- la phase de définition du problème nécessite un module interactif de dialogue avec l'utilisateur et l'appel à ce module pendant la phase de résolution (pour demander des précisions) est souvent nécessaire,
- la construction d'une solution est généralement réalisée par décomposition d'un but en sous-buts ou par spécialisation de buts abstraits en buts de plus en plus spécifiques,
- même si la phase d'évaluation globale de la solution est toujours nécessaire, l'évaluation des résultats intermédiaires en cours d'exécution permet de réduire les erreurs et, par ce fait, de simplifier la phase de correction de la solution proposée.

I.3.3.3 Représentation des résultats

On distingue deux principaux types de représentation des résultats : les chaînes ou graphes d'opérateurs et les plans hiérarchiques de traitement.

La représentation sous forme de chaîne ou de graphe d'opérateurs est généralement utilisée dans les systèmes dédiés. D'une part, l'espace des problèmes est plus réduit, ce qui permet de les définir plus précisément et d'obtenir des résultats plus fiables. D'autre part, les solutions fournies par ces systèmes devront être utilisées en routine et donc avoir un temps d'exécution restreint. Ce type de représentation ne permet pas de mettre en évidence le raisonnement utilisé pour la mise au point de la chaîne ou du graphe et autorise donc difficilement les modifications locales du résultat. Cependant, dans MVP, la suite d'opérateurs conçue est conservée dans un script exécutable qui est annoté par les règles de décomposition de problèmes utilisées pendant la résolution. Ces annotations sont une aide précieuse pour la compréhension du résultat par l'utilisateur. Le système VSDE n'explique pas le raisonnement utilisé à l'utilisateur, mais il lui donne accès à un environnement interactif qui lui permet d'éditer, de modifier et de tester le réseau d'algorithmes créé.

Les plans hiérarchiques permettent de représenter l'ensemble des stratégies utilisées pour aboutir à la solution. Ils sont généralement représentés sous forme d'arbre « et-puis », où chaque niveau intermédiaire de l'arbre correspond à un niveau d'abstraction de l'analyse du problème et où les

feuilles correspondent à un appel à des opérateurs exécutables. Dans BORG, la solution fournie est un plan à cinq niveaux (fig. 17) qui correspondent à une hiérarchisation du TI :

- la requête représente le problème brut défini par l'utilisateur,
- les objectifs définissent les sous-problèmes à résoudre,
- les fonctionnalités sont des méthodes de TI très générales,
- les procédures se rapprochent de la notion d'opérateur sans qu'elles soient liées à une bibliothèque particulière,
- les opérateurs dépendent directement de la bibliothèque retenue.

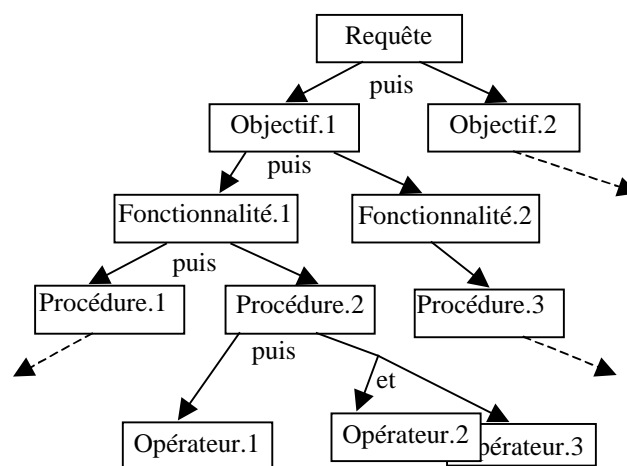


Fig.17 : plan hiérarchique à cinq niveaux de BORG

Dans OCAPI, la solution est également représentée par un plan hiérarchique utilisant les notions de « requête », « but » et « opérateur » définies pour la représentation des connaissances en TI (voir § I.2). Dans MVP, le plan solution est représenté par un graphe dans lequel les nœuds non terminaux correspondent aux buts et aux sous-buts (étiquetés par le nom de l'objet recherché) et où les nœuds terminaux correspondent aux opérateurs et aux données.

I.3.4. Rôle de l'utilisateur dans les systèmes de TI

Dans tous les systèmes, l'utilisateur intervient dans au moins une phase : la définition du problème à résoudre. Cependant les techniques et outils qui lui sont alors proposés peuvent être très divers. Certains nécessitent la connaissance d'un langage de description d'objectif. C'est le cas de BORG dans lequel l'utilisateur présente sa requête par l'intermédiaire d'une fenêtre de l'interface sous forme d'un texte vérifiant une grammaire prédéfinie, ainsi que de ExTI-SATI où l'utilisateur définit son objectif dans un langage approprié à l'aide du module SATI [Capdevielle 95]. D'autres

systèmes, tels que OCAPI, MVP ou BBI, proposent à l'utilisateur de définir son but et son contexte à l'aide d'une interface graphique à options dans laquelle il devra sélectionner l'opération à effectuer ainsi que les valeurs (numériques ou symboliques) caractérisant cette opération et le contexte de l'application. Enfin, quelques systèmes, cherchant à faciliter cette étape à l'utilisateur, lui propose de décrire son problème de manière graphique interactive directement dans l'image. C'est le cas du système IMPRESS où l'utilisateur décrit son but sous forme de surfaces, de lignes et de points qui peuvent être dessinés sur l'image, et du système VSDE dans lequel la phase de « dessin » de l'objectif est accompagnée d'un dialogue dans lequel le système pose des questions (ordonnées dynamiquement par des règles) à l'utilisateur sur les caractéristiques de l'image et des objets recherchés.

A l'exception des systèmes disposant de résultats de référence (dessins dans IMPRESS ou VSDE), qui peuvent évaluer les résultats par calcul de similarité entre des résultats produits et les résultats de référence, l'utilisateur a également à sa charge l'évaluation finale des résultats fournis par le système. En effet, en TI, il n'existe pas de fonction d'évaluation idéale et, en général, seul l'expert du domaine peut juger de la validité finale d'une application. Lorsqu'il juge les résultats insatisfaisants, le système cherche une nouvelle solution en révisant les stratégies employées ou en demandant à l'utilisateur de reconsidérer son but.

Pour la majorité de ces systèmes, l'intervention de l'utilisateur se limite à préciser la requête pour lever des ambiguïtés, à initialiser les paramètres des opérateurs et à faire l'évaluation finale. Cependant, quelques uns lui donnent un rôle plus important pendant la résolution.

Dans VISIPLAN, après avoir spécifié les tâches à accomplir, l'utilisateur réalise la première étape de la planification en sélectionnant les opérateurs de prétraitement (ex. : lissage) et en fixant les valeurs de leurs paramètres. Mais il est également responsable de la recherche des causes d'échec du planificateur.

Dans le système BORG, le rôle accordé à l'utilisateur est différent selon ses compétences :

- un utilisateur sans compétence ne peut utiliser le système qu'en mode automatique, son rôle est alors réduit à entrer sa requête via l'interface,
- un expert en TI peut construire lui même son enchaînement d'opérateurs et ne faire appel au système que pour résoudre certaines tâches,

- un utilisateur qui a la double compétence expert en TI / expert du système peut incrémenter et modifier la base de connaissances et peut intervenir dans la résolution en modifiant les choix faits par le système.

Les systèmes étudiés dans ce chapitre sont des systèmes de résolution de problèmes au moins partiellement automatiques. Il existe une autre famille de systèmes de TI : ce sont les environnements de programmation graphique, tels que KHOROS [Rasure 94] ou VISILOG [Iris 93]. Ces outils permettent à l'utilisateur de sélectionner et d'enchaîner des opérateurs. Dans ces systèmes, les opérateurs ou les regroupements d'opérateurs effectués par l'utilisateur sont vus comme des boîtes noires : il n'y a pas d'explication ni de modélisation du raisonnement de l'utilisateur et le travail doit être repris à zéro pour chaque nouvelle application. Il est donc difficile de réutiliser les éléments de solution construits lors d'une application précédente. Par ailleurs, l'utilisateur doit obligatoirement adopter une démarche ascendante (par regroupement d'opérateurs) lors de la construction d'une application, ce qui ne correspond pas toujours à la démarche du traiteur d'images qui adopte volontiers une démarche ascendante pour des parties de traitement qu'il connaît bien et une démarche descendante pour d'autres.

I.3.5. Conclusion sur les systèmes de TI

Quelque soient les représentations et les techniques employées, on remarquera que dans tous systèmes de TI à base de connaissances, le problème le plus difficile concerne l'évolution et la maintenance de cette base. Dans les systèmes dédiés, les représentations comportent des connaissances détaillées sur le domaine d'application mais permettent difficilement la réutilisation et l'adaptation des solutions. L'intégration de nouvelles connaissances correspond alors à l'insertion d'une nouvelle solution complète, ce qui conduit rapidement à une explosion de la taille de la base. Les systèmes non dédiés nécessitent davantage de connaissances stratégiques pour pouvoir s'abstraire des connaissances du domaine. Mais les connaissances stratégiques étant souvent représentées sous forme de règles, leur acquisition est plus délicate. L'intégration de nouvelles connaissances demande, dans ce cas, l'intervention d'un spécialiste du système pour écrire ces règles et pour maintenir la cohérence de la base. L'écriture de règles pose aussi le problème de l'analyse et de la compréhension du raisonnement de l'expert en TI.

Par ailleurs le manque général d'explicitation des stratégies utilisées permet difficilement à l'utilisateur d'intervenir dans la mise au point d'une application. En particulier, la présentation de la

solution sous forme d'un graphe ou d'une chaîne d'opérateurs est insuffisante pour un système interactif.

Cependant, cette étude nous a permis d'extraire un certain nombre de points importants et intéressants pour la réalisation d'un système interactif permettant l'acquisition et la réutilisation de connaissances en TI :

- la représentation à l'aide de modèles permet de s'abstraire plus facilement du domaine de l'image et de réutiliser les traitements pour des images de différentes origines,
- la séparation des connaissances sur les actions, sur les entités manipulées et sur le raisonnement dans différents modèles facilite l'acquisition interactive des connaissances,
- la représentation d'une application à plusieurs niveaux d'abstraction facilite sa réutilisation et son adaptation en coopération avec l'utilisateur,
- la représentation des stratégies sous forme de décomposition d'un but en sous-buts est facilement compréhensible par l'utilisateur et peut être visualisée sous forme de schémas,
- la représentation des connaissances procédurales et des connaissances sémantiques sur les stratégies dans un même modèle permet de « cacher » les connaissances procédurales derrière les connaissances sémantiques vis à vis de l'utilisateur,
- pour être la plus précise et complète possible, la définition d'un problème de TI doit pouvoir être faite soit par la description des actions à effectuer, soit par la description des objets recherchés, soit par les deux.

Pour la réalisation de notre système, nous allons donc opter pour une représentation des applications sous forme de plans hiérarchiques, intégrant une séparation entre les connaissances statiques et les connaissances stratégiques et autorisant l'incorporation de connaissances sémantiques.

II. Le Raisonnement à Partir de Cas pour l'aide à la réutilisation des connaissances

II.1. Introduction

L'un des buts principaux de l'Intelligence Artificielle (IA) est de concevoir des systèmes capables de reproduire le raisonnement humain. L'atteinte de ce but passe par l'étude des modèles et méthodes mis en œuvre par l'homme. Parmi les principaux modes de raisonnements humains, on peut distinguer le Raisonnement Déductif qui déduit de nouvelles connaissances à partir de celles déjà acquises (ex. : en logique du premier ordre), le Raisonnement Inductif qui généralise une idée à partir d'observations effectuées (ex. : création d'une taxonomie, classification), le Raisonnement Abductif qui recherche les causes d'un fait observé (ex. : diagnostic médical), le Raisonnement par Analogie qui interprète une nouvelle situation par comparaison avec une situation voisine déjà rencontrée (ex : apprentissage symbolique).

Le Raisonnement à Partir de Cas (RàPC) est une forme de raisonnement par analogie. Il consiste à raisonner à partir d'expériences ou de cas déjà rencontrés pour résoudre de nouveaux problèmes. En effet, on cherche souvent à résoudre un problème en se rappelant comment on a résolu un problème similaire : on diagnostique une panne ou une maladie parce que l'on est en présence de symptômes qui ont déjà révélé cette panne ou cette maladie, on établit un plan de travail parce qu'un plan similaire avait donné de bons résultats dans les mêmes circonstances. Le problème de l'homme lorsqu'il utilise ce style de raisonnement est qu'il possède une mémoire limitée et sélective qui ne lui permet pas toujours de se souvenir du cas le mieux approprié. Le RàPC apporte une solution à cette difficulté en ce sens qu'il fournit au système des moyens de mémoriser tous les cas et de retrouver le mieux adapté. Ses avantages pour les domaines d'applications qui possèdent une théorie faible ou mal structurée comme le TI sont multiples, il autorise :

- la représentation des exceptions,
- l'utilisation d'informations manquantes ou bruitées,
- la résolution d'un problème complexe par des interactions entre les solutions de problèmes plus simples,
- l'apprentissage dynamique,
- l'exploitation des cas à des fins pédagogiques.

En s'appuyant sur le même raisonnement que le RàPC, les systèmes d'aide à base de cas [Kolodner 91] assistent l'utilisateur en augmentant sa mémoire et en sélectionnant les cas les plus

proches du problème posé. Mais contrairement aux systèmes de RàPC, entièrement automatiques, ils laissent l'utilisateur prendre la décision finale : à savoir, sélectionner le cas le plus approprié et décider des besoins d'adaptation. Ce type d'aide permet de laisser la priorité à l'interactivité tout en assistant l'utilisateur. Dans la suite de ce chapitre, le terme « système de RàPC » sera employé aussi bien pour les systèmes automatiques de RàPC que pour les systèmes d'aide à base de cas.

Le fonctionnement général d'un système de RàPC peut se décomposer en cinq étapes principales (fig. 18) :

- « sélection » : cette étape consiste à sélectionner les cas de la base de cas qui sont les plus proches du problème courant. Pour que cette étape soit efficace, le système doit disposer de moyens de remémoration et d'indexation pour retrouver les meilleurs cas rapidement. Le système doit donc intégrer des mesures de similarité entre les cas qui autorisent une sélection fine.
- « adaptation » : il s'agit d'adapter les solutions proposées par l'étape de sélection pour obtenir une nouvelle solution. Pour cela, le système prend en compte toutes les informations fournies par la description du cas courant, évalue les différences entre celui-ci et les cas sélectionnés puis détermine une nouvelle solution en utilisant des connaissances d'adaptation spécifiques.
- « évaluation » : cette étape a pour but la validation de la solution produite. La confrontation de cette solution au monde réel est là pour détecter les erreurs des deux étapes précédentes. Elle est généralement réalisée dans une boucle évaluation / correction.
- « correction » : cette étape consiste généralement à relancer l'étape d'adaptation ou les étapes de sélection et d'adaptation. Elle évite de rester sur un échec et passe par l'acquisition de nouvelles connaissances pour essayer d'expliquer les erreurs.
- « apprentissage » : la nouvelle solution produite peut être prise en compte dans la base de cas. Mais cette phase requiert une restructuration de la base, car un simple ajout conduirait à terme à une base trop importante et donc ralentirait la phase de sélection.

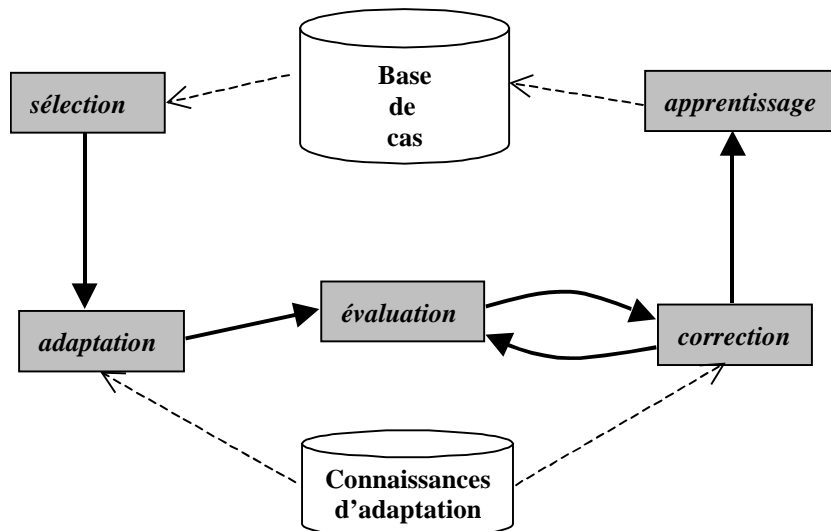


Fig.18 : fonctionnement général du RàPC

Ce découpage en cinq phases n'est pas exhaustif, les différentes phases ne sont pas toujours toutes présentes et distinctes : par exemple, certains systèmes ne réalisent pas d'adaptation et dans ceux qui la réalisent, elle est généralement associée aux phases d'évaluation et de correction.

Dans la suite de ce chapitre, nous présentons une étude de plusieurs systèmes de RàPC. Ces systèmes s'appliquent dans des domaines divers et utilisent différentes techniques pour représenter les cas mais aussi pour résoudre les problèmes. Parmi les types de problèmes résolus, on peut citer la conception avec le système de Chiu [Chiu 96], le système de Caulier [Caulier 95], DESIGNER [Chiron 97], EADOCS [Netten 96a] [Netten 96b] [Netten 97] et Déjà-Vu [Smyth 95] [Smyth 96], l'aide à la prise de décision avec ISAC [Bonzano 96] [Bonzano 97a] [Bonzano 97b], l'interprétation avec SICT-WICT [Grimnes 96], la planification avec le système de Prasad [Prasad 95], Resyn/RàPC [Lieber 97], PRODIGY/ANALOGY [Velo 96] [Muñoz-Avila 96], CAPLAN/CBC [Velo 96] [Muñoz-Avila 96], PARIS [Velo 96] et SEIDAM [Charlebois 91] [Charlebois 94] [Charlebois 96a], [Charlebois 96b].

Ces systèmes sont abordés suivant plusieurs points de vue : la représentation et l'organisation des cas, la phase de remémoration, la phase d'adaptation et la mémorisation d'un nouveau cas. Le tableau récapitulatif de la figure 19 sert de guide de lecture du chapitre : pour chacun des systèmes, il rappelle le type de problème résolu, le domaine d'application ou les objectifs et l'ensemble des points de vue présentés dans la suite de ce chapitre.

Système	Type de problème	Domaine d'application/objectif	Points de vue présentés
Système de Chiu	Conception	Pilotage de production industrielle	RC, RM, AD
Système de Caulier	Conception	Assistance à la conception pour la supervision d'un réseau de télécommunication	RM, MM
DESIGNER	Conception	Assistance à la conception en supervision industrielle (tableau de bord de surveillance)	RC, RM, AD
EADOCS	Conception	Construction de panneau de fibres composites	RM, AD
Déjà-Vu	Conception	Assistance ou automatisation de tâches de construction de logiciel	RC, RM, AD, MM
ISAC	Décision	Aide à la prise de décision dans le contrôle du trafic aérien	RC, RM
SICT-WICT	Interprétation	Interprétation d'images de tomographie	RC, AD, MM
Système de Prasad	Planification	Planification de recettes de cuisine	RC, RM, AD, MM
Resyn/RàPC	Planification	Construction de plans de synthèse pour des molécules en chimie organique	RC, RM, AD
PRODIGY/ ANALOGY	Planification	Divers domaines d'application dont le transport de marchandises	AD
CAPLAN/CBC	Planification	Planification de processus (ex : construction de pièces mécaniques)	AD
PARIS	Planification	Non dédié à un domaine d'application	RC, AD, MM
SEIDAM	Planification	Gestion de données provenant de capteurs satellites ou aériens et de GIS (Geography Information System)	RC, RM
Légende : RC = représentation et organisation des cas ; RM = phase de remémoration ; AD = phase d'adaptation ; MM = mémorisation des nouveaux cas			

Fig.19 : tableau récapitulatif des systèmes étudiés

II.2. Représentation et organisation des cas

Un système de RàPC manipule deux types de cas : les cas dits « cas sources » et les cas dits « cas cibles ». Un cas source est un cas de la base qui a déjà été résolu ; il contient la définition du problème ainsi que la solution proposée. Le cas cible est le cas de la situation courante, il correspond à la définition d'un problème pour lequel on cherche une solution.

Pour identifier les différents cas de la base et pour pouvoir les comparer à une nouvelle situation, un ensemble de critères caractérisant le cas est attaché, soit au problème, soit à la solution, soit au

deux. Une des principales difficultés de l'application du RàPC à un domaine particulier est de déterminer les bons critères de ce domaine, c'est-à-dire les critères qui autoriseront une sélection fine et sûre.

Nous présentons ci-après les représentations des cas choisies dans les systèmes étudiés. Les présentations sont classées en fonction du type de problème résolu.

II.2.1. Problèmes de conception

Les systèmes traitant des problèmes de conception s'appuient généralement sur une représentation objet et une organisation des classes de cas en hiérarchies, ces hiérarchies étant utilisées ensuite pour faciliter l'indexation des cas. Nous présentons plus en détail trois d'entre eux utilisant une représentation plus spécifique.

Le *système de Chiu* traite de pilotage de production industrielle, son but principal étant d'éviter tout échec dans la chaîne de production. Pour cela, il utilise la notion de donnée « FMEA » qui est définie ainsi : une donnée FMEA (Failure Mode and Effect Analysis) contient l'ensemble des possibilités d'échecs et leurs conséquences, elle est décrite par un défaut, une action corrective et le résultat de l'action. Le système de Chiu comporte trois bases de cas (fig. 20), contenant chacune des cas d'un niveau d'abstraction particulier : la base de haut niveau contenant les données sur les produits (composition, processus de fabrication, FMEA), la base de moyen niveau contenant les données sur les éléments composants les produits (processus de fabrication, FMEA) et la base de bas niveau contenant les données sur les processus de fabrication (FMEA). Ces trois bases sont donc reliées par les éléments qui les composent.

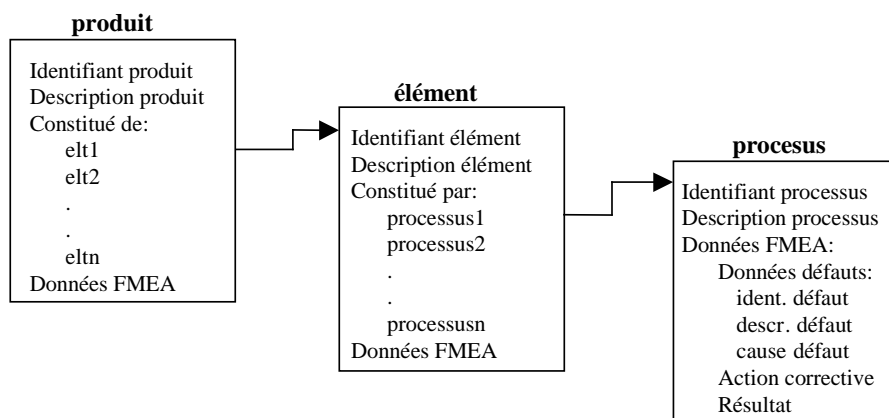


Fig.20 : organisation des trois bases de cas dans le système de Chiu

L'intérêt de cette représentation réside dans le fait qu'un cas peut être considéré en totalité ou en partie, ce qui facilite les phases de remémoration et d'adaptation.

DESIGNER est dédié à la conception de systèmes de supervision industrielle développés dans le cadre plus général du projet PADIM [Fuchs 97]. A partir de la description du système à superviser, donnée par l'utilisateur, il aide ce dernier à concevoir le nouveau système de tableau de bord de surveillance. Pour cela, il utilise des modèles génériques de connaissances pour la supervision : les « objets de supervision » qui décrivent les connaissances spécifiques à la supervision (fonctions, équipements, ...) et les « objets de représentation » qui décrivent le système sous forme d'objets d'interface (graphiques, courbes, photos, textes, ...). Ces modèles sont ensuite utilisés pour définir un cas : un cas est un couple problème / solution où le problème est défini à l'aide d'un ensemble d'objets de supervision et la solution à l'aide d'un ensemble d'objets de représentation. Un problème de conception peut ainsi être décomposé en sous-problèmes correspondant chacun à un objet de supervision pour lequel on recherchera l'objet de représentation solution.

Cette décomposition en sous-problèmes associés chacun à une partie de solution facilite la résolution d'un cas par composition de parties de solution. Cependant, elle n'est applicable que dans les domaines où la décomposition en sous-problèmes (inv. la composition de sous-problèmes) peut être faite en parallèle avec la décomposition en sous-solutions (inv. la composition de sous-solutions).

Déjà-vu est un système de conception de logiciels de pilotage d'installation industrielle. Un cas correspond à une tâche à réaliser dans l'usine, par exemple la gestion de véhicules sur rails qui chargent et déchargent des rouleaux d'acier. Un cas comporte une partie description et une partie solution : la partie description décrit les caractéristiques de la tâche (nature, machines requises, conditions initiales, but et contraintes de fonctionnement) et la partie solution correspond à l'organigramme du programme qui réalise la tâche. Les cas sont organisés en hiérarchies comportant deux types de cas : les cas de conception (cas concrets) dont la solution est directement un programme informatique et les cas de décomposition (cas abstraits) qui peuvent contenir du code mais également d'autres spécifications telles que la décomposition du problème en sous-problèmes plus simples. Ainsi la solution du cas racine d'une hiérarchie correspond à une vue abstraite du problème représenté dans les cas de la hiérarchie et chaque cas intermédiaire fournit une solution à un certain niveau d'abstraction de conception.

Cette représentation à différents niveaux d'abstraction est utilisée pour guider la recherche d'un cas similaire niveau par niveau et permet de mettre au point une nouvelle solution en combinant des portions de solutions complexes.

II.2.2. Problèmes de planification

La représentation des cas pour les problèmes de planification est plus délicate : une nouvelle solution est généralement construite en combinant des morceaux de plans issus de plusieurs cas et les informations permettant la sélection doivent donc se trouver dans la définition du problème mais également dans les différents éléments composant la solution. Un plan peut être linéaire ou hiérarchique (fig. 21). Un plan linéaire est défini comme un ensemble ordonné d'étapes, une étape correspondant à un état et à une transition. Un plan hiérarchique est composé de plusieurs niveaux d'abstraction ; chaque niveau décompose les problèmes du niveau supérieur en sous-problèmes, le niveau le plus bas décrit la suite d'étapes à réaliser. Un problème de planification correspond à un état initial e_0 et à la spécification d'un but b . Une solution est un plan dont le premier état est e_0 et dont le dernier état satisfait le but b .

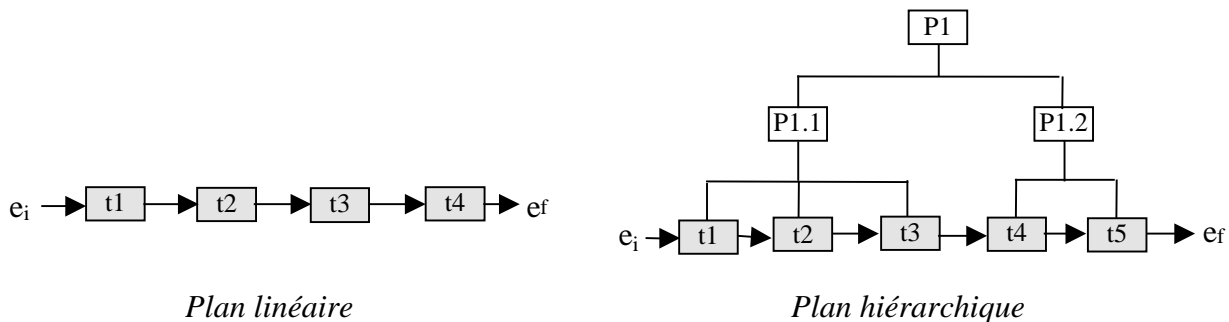


Fig.21 : plan linéaire et plan hiérarchique

Ces deux techniques ne sont pas toujours exclusives, ainsi pour réaliser un plan linéaire, le système **PARIS** utilise des techniques liées à la planification hiérarchique : les nouveaux cas sont abstraits à différents niveaux avant d'être stockés dans la base. Pour cela, on regroupe un ensemble d'opérateurs concrets en un seul opérateur abstrait, ce qui facilite par la suite la phase de remémoration.

Le **système de Prasad** a pour domaine d'application la planification de recettes de cuisson de légumes. Une recette est représentée par un plan hiérarchique d'opérateurs correspondant à des actions qui sont décomposées en sous-opérateurs jusqu'à obtention d'opérateurs de base. Les connaissances liées aux cas sont organisées dans deux bases : la base des propriétés et la base des plans. La base des propriétés comporte les propriétés des objets du domaine et est organisée en une hiérarchie. La base des plans se compose d'ensembles de structures de plans. Chaque structure correspond à une classe de plans organisés en hiérarchie (un type de recette est représenté par une classe de plan) et est identifié par l'opérateur de plus haut niveau de la classe (ex : « frire »). Cet opérateur possède des liens vers les opérateurs de niveaux inférieurs suivant lesquels il peut se

décomposer (ex : « préparer légume », « bouillir », préparer friture », « réaliser friture »), ces liens étant indexés par les propriétés des objets du domaine. Ce sont ces index, organisés hiérarchiquement dans la base des propriétés qui seront utilisés dans les phases de remémoration et d'adaptation.

La définition des structures de plans associées à des classes de problèmes va permettre de réduire rapidement l'espace de recherche pendant la phase de remémoration.

Le but du système **Resyn RàPC** est la construction de plans de synthèse pour des molécules en chimie organique. Un plan de synthèse est représenté par un plan linéaire où les états sont des structures moléculaires et les transitions des transformations moléculaires. Les structures moléculaires peuvent être comparées à l'aide d'une relation de subsomption (la molécule M1 est plus générale que la molécule M2, c'est-à-dire qu'elle correspond à la molécule M2 à laquelle on a ôté des atomes), cette relation est utilisée pour l'indexation des plans dans la base. Ainsi, un plan est indexé par la plus petite structure moléculaire qui permet l'application du plan et l'ensemble des index est organisé en une hiérarchie respectant la relation de subsomption.

L'objectif de **SEIDAM** (System of Experts for Intelligent Data Management) est de répondre à des requêtes concernant les forêts et l'environnement grâce à des capteurs distants (aériens ou satellites), des informations géographiques, des modèles et des mesures. Un cas correspond à la solution « généralisée » d'un problème déjà traité et est composé d'une requête, de données provenant des capteurs, d'informations géographiques et de méthodes d'analyse permettant de répondre à la requête. Le rôle du module de RàPC est de construire un plan pour répondre à la requête en utilisant le meilleur choix de données et de méthodes. Un plan correspond à un ensemble ordonné d'opérateurs, chacun de ces opérateurs faisant appel à un agent logiciel pour réaliser la tâche qui lui est assignée (ex. : copier un fichier dans un répertoire de travail, changer le format des données, segmenter une image, ...).

Les techniques de stockage des cas ne sont pas détaillées : les cas sont repérés par les buts et sous-buts qui y ont été résolus mais la base de cas ne semble pas posséder de technique d'indexation particulière.

II.2.3. Autres types de problèmes

Le système **ISAC** est un système d'aide à la prise de décision dans le domaine du trafic aérien. Son but est de résoudre des conflits de trajectoires entre deux avions en sélectionnant l'avion qui

doit manœuvrer et en décidant du type de manœuvre à effectuer. Ces décisions dépendent de la géométrie du conflit, des capacités des avions, de leur position, de leur destination, Le cas cible est construit automatiquement par le système HIPS qui gère les données des avions. Lorsque HIPS détecte un conflit, il trie et transforme les données le concernant avant de les transférer au système ISAC, ces données devenant les caractéristiques du nouveau cas cible. Un cas est ainsi décrit par un ensemble de couples attributs/valeurs et par la description des manœuvres choisies.

La détermination des critères importants est donc faite dynamiquement pour chaque cas, ce qui est particulièrement important pour un système devant travailler en temps réel. Mais le fait que cette détermination soit réalisée à l'extérieur du système empêche toute révision pendant la remémoration et risque donc de faire perdre de l'efficacité au système.

Le système *SICT-WICT* s'intéresse à l'interprétation d'image de tomographie. Sa particularité est d'être composé de deux raisonneurs à partir de cas : SICT (Segment Image Creek Task) propose une hypothèse d'interprétation pour chaque « segment » présent dans l'image (un segment est une zone correspondant à un objet ou à une partie anatomique) et WICT (Wholistic Image Creek Task) gère l'interprétation globale de l'image. Dans SICT, un cas est composé du segment définissant le problème, d'une hypothèse d'interprétation (la solution) et des hypothèses rejetées (justification). Dans WICT, un cas est composé, entre autres, d'un ensemble de segments avec leurs hypothèses et de la description du problème. Les cas résolus par le raisonneur SICT servent ensuite d'index pour le raisonneur WICT.

II.2.4. Bilan sur la représentation des cas

D'une part, déterminer les critères pertinents pour la caractérisation d'un cas demande une analyse approfondie du domaine. En particulier, dans certains domaines, ce ne sont pas toujours les mêmes critères qui sont utiles pour tous les cas. Pour donner le maximum d'efficacité au système, le choix des critères importants doit pouvoir être fait dynamiquement pour chaque cas et doit pouvoir être révisé en cours de résolution. On notera, par ailleurs, le besoin d'organiser la base et/ou les index dans des structures ou des hiérarchies à l'aide de ces différents critères de façon à accélérer la recherche d'un cas.

D'autre part, pour les systèmes de conception et de planification, on notera l'intérêt de construire une solution par composition de sous-solutions de plusieurs cas. Pour cela, on doit pouvoir considérer un cas en totalité ou en partie, trouver des moyens d'associer les sous-problèmes à des

sous-solutions et des moyens pour reconstruire une solution à l'aide de plusieurs éléments. On peut pour cela utiliser la représentation par plans hiérarchiques ou la représentation d'un même cas à différents niveaux d'abstraction et ainsi associer des critères à un cas abstrait ou à une partie de cas aussi bien qu'à un cas concret complet.

II.3. La phase de remémoration

Le but de cette phase est de sélectionner un ou plusieurs cas pour proposer, soit directement une solution, soit des moyens de progresser vers la solution (modèles, méthodes, solution partielle, ...). Pour la réaliser, il est nécessaire de définir des techniques de remémoration et d'indexation pour retrouver les cas les plus pertinents et ce, le plus rapidement possible. Cette recherche peut utiliser une indexation dynamique des cas ou une classification plus statique basée sur des niveaux d'abstraction ou sur la subsomption. Le système doit également posséder une (ou des) mesure(s) de similarité pour effectuer une sélection fine des cas. De telles mesures cherchent, soit à maximiser la similarité entre le problème posé dans le cas cible et celui posé dans le cas source, soit à minimiser l'effort d'adaptation entre les deux solutions.

Nous présentons ci-après les techniques de remémoration des systèmes étudiés en fonction du nombre d'étapes qu'ils nécessitent pour réaliser cette phase.

II.3.1. Remémoration en deux étapes

Certains systèmes séparent la phase de remémoration en deux étapes (fig.22), la première ayant pour but de réduire l'espace de recherche et la deuxième de sélectionner les cas les plus proches du cas courant dans cet espace réduit. C'est le principe utilisé par les systèmes EADOCS, ISAC et DESIGNER.

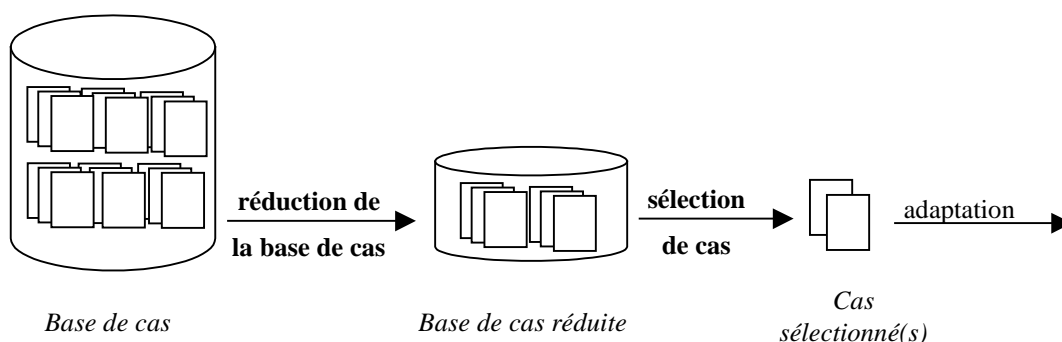


Fig.22 : remémoration de cas en deux étapes

Le système *EADOCS* s'intéresse à la conception de panneaux de fibres composites. Un problème y est défini par des besoins fonctionnels, des préférences sur les solutions et des critères d'optimalité. Les solutions sont organisées en classes. Chaque classe est caractérisée par les modes de comportement d'un prototype, un prototype définissant la structure de la solution. Une première étape permet de réduire l'espace de recherche en sélectionnant un/des prototype/s répondant aux besoins fonctionnels. La solution qualitative sélectionnée doit ensuite être transformée en une solution quantitative. La quantification de la solution prototype est réalisée en plusieurs étapes : recherche de cas similaires dans la classe, instantiation des solutions ou de parties des solutions des cas sélectionnés, assemblage des différents éléments instanciés pour construire une première solution qui sera ensuite adaptée au problème courant. La similarité entre deux cas d'une même classe dépend de la valeur de la somme pondérée des similarités sur les critères d'optimalité et de l'appariement des préférences sur les solutions.

Cette technique qui consiste en fait à sélectionner un type de problème pour travailler sur un espace de recherche réduit est intéressante mais ne peut s'appliquer que sur des domaines bien structurés.

Dans *ISAC*, la recherche d'un cas similaire est également réalisée en deux étapes (fig. 23). Lors du lancement de la remémoration, un réseau d'index est construit entre les cas de la base. La première étape recherche les cas sources « compatibles » avec le cas cible : deux cas sont dits compatibles s'ils respectent des contraintes prédéfinies sur certaines de leurs caractéristiques. La deuxième étape recherche les cas les plus proches du cas courant parmi les cas sélectionnés lors de la première étape. Les index relient les cas ayant la même valeur pour un paramètre, ce qui permet de réaliser la dernière étape de remémoration par propagation des critères le long du réseau d'index. L'activation de la propagation est proportionnelle au poids accordé à chaque critère.

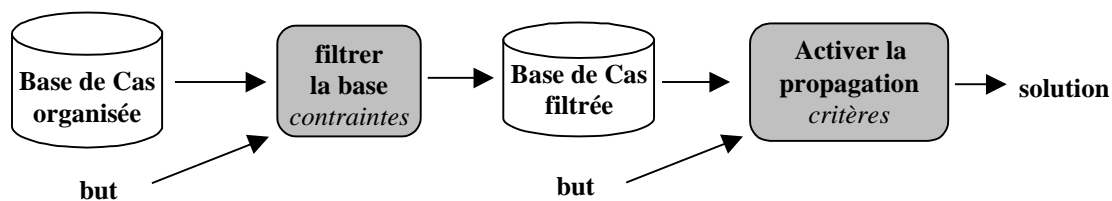


Fig.23 : les étapes de remémoration dans ISAC

Cette technique consiste en fait à définir deux types de contraintes sur les critères : des contraintes « fermes » qui vont fournir un espace de recherche réduit en dehors duquel il ne serait pas possible de trouver la solution, et des contraintes plus « souples » que l'on cherchera à respecter au mieux sans forcément les respecter intégralement.

DESIGNER est un système d'aide à la conception, il assiste l'expert en construisant une mémoire « artificielle » de son expérience. Pour faciliter la communication entre le système et l'expert, les auteurs de Designer ont cherché à construire une fonction de similarité qui s'approche le plus possible de la démarche suivie par l'homme. Ainsi, lorsqu'il définit son problème, l'utilisateur peut à tout moment consulter le cas le plus proche et être ainsi guidé par la fonction de similarité pour donner une définition plus précise du problème. Par ailleurs, le calcul de la similarité entre deux cas est réalisé en deux étapes : un calcul de la « similarité de surface » et un calcul de la « similarité de profondeur ». La similarité de surface détermine dans quelle mesure les cas comparés sont décrits par les mêmes types d'information alors que la similarité de profondeur calcule la proximité des valeurs des attributs (pour les attributs comparables). Ainsi, la similarité totale entre deux cas est définie comme étant la moyenne entre la similarité de surface et la similarité de profondeur.

La première mesure correspond bien à une sélection des cas compatibles, mais contrairement aux systèmes décrits précédemment, les deux étapes sont réalisées en parallèle. Les cas décrits par un ensemble différent de critères ne sont pas directement « disqualifiés », mais leur similarité avec le cas courant sera affaiblie.

II.3.2. Remémoration multi-étapes

Les systèmes modélisant les cas sur plusieurs niveaux d'abstraction, en particulier les planificateurs à base de cas, utilisent généralement un cycle remémoration/adaptation (fig. 24) qu'ils exécutent, soit un nombre prédéfini de fois, soit jusqu'à respect d'une certaine condition. Nous présentons ici la phase de remémoration multi-étapes du système de Chiu, de Déjà-Vu et du système de Prasad.

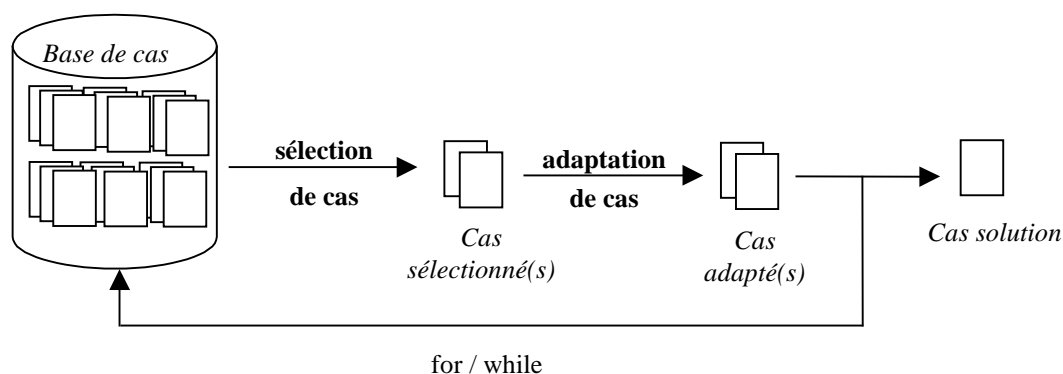


Fig.24 : remémoration de cas multi-étapes

Le *système de Chiu*, dont les cas sont organisés en trois niveaux d'abstraction (§ II.2), utilise une recherche multi-étapes pour permettre au raisonneur d'extraire le cas le plus similaire niveau par niveau. Il exploite d'abord des indices pour isoler le cas le plus similaire au cas courant dans la base de cas du plus haut niveau. Ensuite, il propose à l'utilisateur de sélectionner une donnée du cas source qui n'est pas satisfaisante. Cette donnée correspondant à un cas du deuxième niveau d'abstraction, il la modifie en relançant le processus de remémoration sur la deuxième base de cas. Le même principe est utilisé pour passer du deuxième au troisième niveau d'abstraction.

Le principe utilisé ici peut être étendu à n niveaux et est particulièrement intéressant pour les systèmes d'aide à l'utilisateur. De plus, il peut être appliqué aussi bien à des problèmes de planification qu'à des problèmes de conception, à condition de pouvoir modéliser le plan ou l'objet à concevoir sous la forme d'un arbre de décomposition.

La philosophie utilisée pour la phase de remémoration dans *Déjà-Vu* est d'essayer de minimiser les besoins d'adaptation au nouveau cas. La fonction de similarité définit le cas le plus similaire comme étant celui qui demandera le minimum d'effort d'adaptation. Pour cela, elle utilise les prémisses des règles d'adaptation, ces règles définissant chacune une tâche d'adaptation spécifique. Le deuxième principe de base de ce système est de représenter une solution complète par un ensemble de cas à différents niveaux d'abstraction de façon à construire une nouvelle solution en combinant plusieurs cas. Les phases de remémoration et d'adaptation ne sont pas complètement séparées. Pour résoudre un problème, le système exécute plusieurs fois le cycle remémoration/adaptation. A chaque cycle, il recherche un cas abstrait correspondant à une décomposition possible en sous-problèmes, puis il adapte cette solution abstraite à l'aide des connaissances d'adaptation. Ce cycle est exécuté jusqu'à l'obtention d'éléments de solution exécutable qui sont alors intégrés dans la solution finale.

Les auteurs de ce système partent du principe que le cas dont la définition du problème est la plus proche du problème courant n'est pas forcément le cas le plus facile à adapter au problème courant. Ce principe se vérifie sur de nombreux domaines d'application, mais il est relativement délicat à mettre en œuvre car il est souvent difficile d'évaluer les besoins d'adaptation d'une solution et même parfois d'évaluer la solution elle-même.

Le *système de Prasad* utilise également des techniques de planification hiérarchique. Il recherche une solution en appliquant plusieurs fois un cycle remémoration/adaptation jusqu'à l'obtention d'une suite d'opérateurs exécutables. Il part de la définition du problème et d'un plan où chaque niveau est défini comme un ensemble vide d'opérateurs. A chaque niveau (en partant du niveau le

plus élevé), le cycle remémoration/adaptation est appliqué comme suit : sélectionner une décomposition en sous-buts acceptable pour chaque opérateur de ce niveau, adapter cette décomposition en fonction des propriétés des objets du domaine, stocker les opérateurs résultants dans la liste des opérateurs du niveau directement inférieur.

La réalisation d'une phase d'adaptation à chaque fois que l'on descend d'un niveau dans le plan permet de redéfinir précisément les sous-problèmes restant et d'éviter les retour-arrière.

II.3.3. Remémoration à options

D'autres systèmes proposent plusieurs options pour réaliser la remémoration (fig. 25). Cela permet de choisir dynamiquement la technique à mettre en œuvre en fonction de ce que recherche l'utilisateur ou de ce que le système peut lui proposer comme solution. Dans la suite de ce paragraphe, nous décrivons le système de Caulier qui propose plusieurs index et plusieurs mesures de similarité, puis les systèmes Resyn/RàPC et SEIDAM qui offrent deux options différentes suivant qu'il existe ou non un cas s'appariant exactement au nouveau problème.

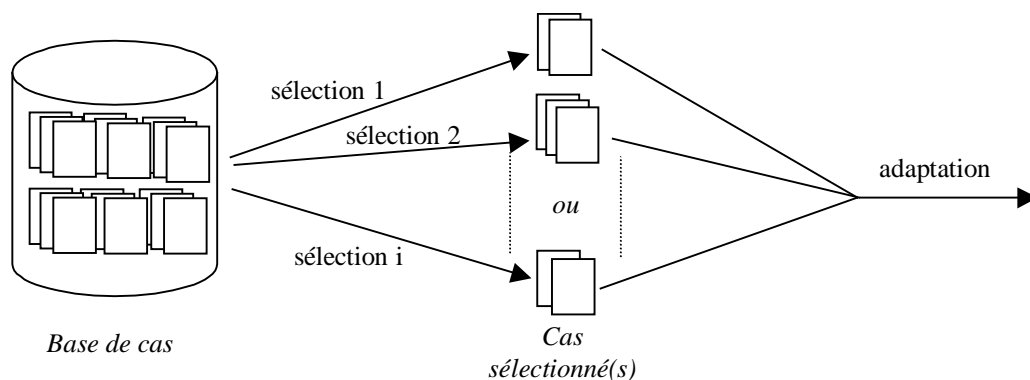


Fig.25 : remémoration de cas à options

Le *système de Caulier* est un système d'assistance à la construction et à la réutilisation de connaissances dans le cadre des activités de supervision d'un réseau de télécommunication. Ce système permet de rechercher un cas suivant plusieurs points de vue associés à des objectifs différents suivant l'utilisateur. A chaque point de vue correspond un index des cas et une mesure de similarité. Un index est représenté par un arbre de décision dont les nœuds sont les descripteurs les plus discriminants pour ce point de vue. Une première sélection de cas est réalisée à l'aide de l'index, puis la recherche du cas le plus pertinent est effectuée à l'aide de la mesure de similarité correspondant à cet index. Cette mesure est définie comme la somme pondérée des comparaisons

sur chaque descripteur, le choix des critères et des poids pour une mesure donnée dépendant de l'expertise du domaine.

Ce système utilise donc une option de remémoration différente pour chaque type d'utilisateur. Mais il pourrait être également classé dans les systèmes effectuant la remémoration en deux phases puisque la première sélection réalisée à l'aide d'un index correspond à une réduction de l'espace de recherche.

La phase de remémoration du système **Resyn/RàPC** est basée sur des techniques de classification. Il utilise pour cela la hiérarchie d'index décrite au § II.2.2 dans laquelle un index est une molécule subsumant le problème du cas (la molécule cible). Deux types de classification sont utilisées pour la recherche des cas similaires. La première correspond à une *classification dure* qui va rechercher un cas possédant un index plus général que le problème cible. Si aucun plan n'est associé à cet index, la recherche est poursuivie à l'aide d'une *classification élastique* qui modifie le problème et les index des cas pour réaliser un appariement approximatif. Pour cela, le système recherche parmi les fonctions de transformation définies en chimie organique, celles qui lui permettent de transformer le cas cible et un cas source pour les apparier. Ces fonctions de transformation définissent le *chemin de similarité* qui sera utilisé pour la phase d'adaptation.

L'originalité de cette approche est de réaliser la remémoration en commençant par définir les pas d'adaptation. L'avantage de cette technique est que si la remémoration réussit, l'adaptation est assurée. Mais pour chaque fonction de transformation d'un problème A en un problème B, on doit être capable de définir la fonction de transformation de la solution de B en une solution de A.

Dans **SEIDAM**, l'utilisateur pose sa requête en terme d'un ensemble de buts à atteindre. Le module de RàPC est utilisé pour déterminer des plans résolvant chacun de ces buts. Deux cas sont considérés similaires si les expressions représentant leur but possèdent le même opérateur et la même arité. Deux options sont utilisées pour la recherche d'un cas similaire : la première recherche s'il existe des cas dont le but s'apparie exactement, ils sont alors extraits en premier et l'on vérifie que le plan solution s'applique à l'état initial ; en l'absence d'appariement exact, une deuxième option consiste à sélectionner des cas et à tenter de les adapter au but courant en utilisant des techniques de régression de but [Fikes 81].

II.3.4. Bilan sur la phase de remémoration

L'intérêt d'une étape préliminaire visant à réduire l'espace de recherche paraît évident. Cela permet d'explorer l'espace restant plus en profondeur et d'utiliser des techniques d'essais-erreurs qui demanderaient trop de temps sur l'espace complet. Pour la réaliser, on peut, suivant le domaine d'application et sa représentation, soit considérer que certains critères sont plus importants que les autres et qu'ils définissent des contraintes « fermes » qu'il n'est pas possible de contourner, soit considérer que les cas ne peuvent être comparés que s'ils sont définis avec le même ensemble de critères.

On a vu que la pertinence d'un critère est souvent dépendante de la situation courante (cas cible, contexte d'utilisation, type d'utilisateur, ...), le calcul de la similarité entre deux cas doit donc être modulable en fonction des critères pertinents pour la situation donnée. En particulier, le choix des critères à prendre en compte et la définition du poids associé à chaque critère doit se faire dynamiquement.

Pour les systèmes de planification, il est particulièrement judicieux de mettre en œuvre un cycle de remémoration/adaptation applicable itérativement sur chaque niveau d'abstraction du plan. En particulier, dans les systèmes d'aide à l'utilisateur, où ce dernier peut ainsi recadrer son problème après chaque cycle.

II.4. La phase d'adaptation

Le but de la phase d'adaptation est de construire une solution qui résolve le problème du cas cible à partir du/des cas sélectionné/s lors de l'étape de remémoration. Cette phase n'est pas présente dans tous les systèmes : les systèmes d'interprétation, en particulier, se limitent généralement à la phase de remémoration. Elle est également souvent absente dans les systèmes d'assistance à l'utilisateur où le module de RàPC est plus utilisé pour pallier les problèmes de limitation et de sélectivité de la mémoire humaine et pour faire partager les expériences de plusieurs utilisateurs, que pour faire de la résolution automatique de problèmes. Par ailleurs, cette phase demande des connaissances d'adaptation spécifiques qui ne sont pas toujours aisées à modéliser et à mettre en œuvre.

Dans les systèmes où les connaissances permettant d'adapter une solution ne sont pas connues ou pas explicites, l'adaptation peut être réalisée au niveau de la définition du problème. Ainsi dans le système pour l'interprétation d'images *SICT-WICT*, les phases de remémoration/adaptation sont

dérivées de la méthode de résolution de problème définie dans CommonKADS [Breuker 94b] « proposer – critiquer – réviser ». La remémoration y correspond à la tâche « proposer », l'adaptation aux tâches « critiquer » et « réviser ». La tâche « critiquer » teste la qualité de la solution et la tâche « réviser » modifie le problème en fonction des résultats de la critique pour relancer le processus de segmentation/interprétation si besoin. La modélisation du RàPC à l'aide des modèles génériques de CommonKADS autorise son intégration dans des systèmes plus complexes et permet en particulier la coopération du RàPC avec d'autres techniques de résolution de problèmes.

Dans la suite de ce paragraphe, nous présentons les connaissances et techniques utilisées pour réaliser cette phase dans les systèmes de conception, puis dans les systèmes de planification.

II.4.1. Systèmes de conception

Dans les systèmes de conception, l'adaptation est le plus souvent réalisée à l'aide d'un ensemble de règles de transformation. Ces règles peuvent modifier directement des éléments de la solution, ou modifier des sous buts non satisfaits pour relancer localement le processus de remémoration. Elles sont généralement gérées par des connaissances de contrôle qui vérifient la cohérence de la solution adaptée. Quatre types d'adaptation dans des systèmes de conception sont décrites ci-après.

La phase d'adaptation du système **DESIGNER** consiste à créer un ensemble d'objets de représentation (la solution du cas cible décrite sous forme de schémas, courbes, ...) à partir d'un ensemble d'objets de supervision (objets des cas sélectionnés décrits sous forme de fonctions à exécuter, d'équipements nécessaires, ...).

Chaque objet de supervision pointe sur une liste d'objets de représentation. Les objets de représentation permettant de mettre en place la solution du cas cible sont obtenus par des fonctions de transformation. Ces fonctions font appel, soit à des règles simples (création d'un objet de représentation), soit à des règles de transformation visant à réduire l'écart entre les connaissances du cas courant et celles de la base de cas (ex. : duplication d'un élément sur l'exemple de la figure 26).

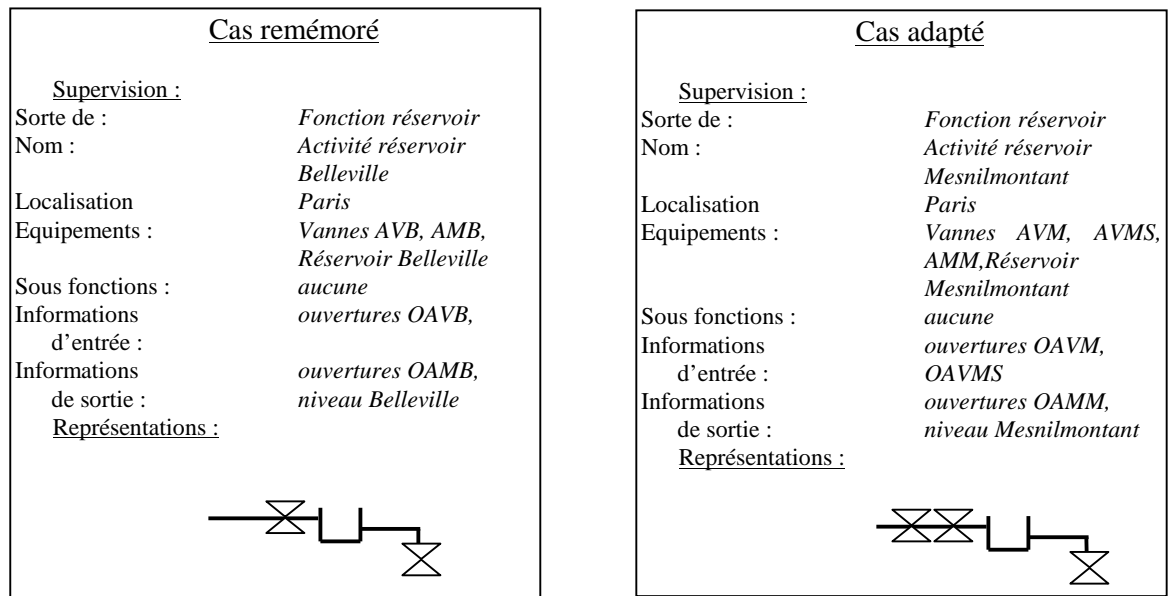


Fig.26 : exemple d'adaptation d'un cas dans DESIGNER

L'adaptation est ici entièrement basée sur des modèles génériques de connaissances du domaine dont la définition résulte d'une étude approfondie des problèmes de supervision industrielle.

EADOCS réalise une adaptation incrémentale du cas sélectionné par substitution ou instantiation de sous-solutions d'autres cas. Le cas cible est défini par un ensemble de sous-buts à atteindre qui ne sont généralement pas tous satisfaits par le cas sélectionné pendant l'étape de remémoration. EADOCS cherche alors à modifier la solution conceptuelle fournie par le cas sélectionné tout en conservant sa structure. La structure de la solution est appelée prototype et correspond au critère le plus important qui est le comportement général de l'objet à concevoir. Un nouveau cas cible est alors défini avec l'ensemble des sous-buts non satisfaits et le prototype du cas sélectionné. Cette nouvelle définition permet de relancer la remémoration d'un cas qui satisfasse les objectifs restants et qui soit intégrable à la solution conceptuelle initialement proposée. Ce processus peut être relancé jusqu'à l'obtention d'une solution satisfaisant tous les buts (fig. 27). Il est ensuite complété par une optimisation numérique.

La phase d'adaptation de ce système a demandé la mise en œuvre de processus complexes. En effet, le domaine d'application impose des contraintes lourdes sur la conception d'un élément. En particulier, un problème n'est pas décomposable en sous-problèmes résolubles séparément. Ces contraintes ont imposé de définir un modèle de cas cible et un processus de remémoration spécifiques à la phase d'adaptation et des modules de contrôle de cohérence et de gestion de conflits.

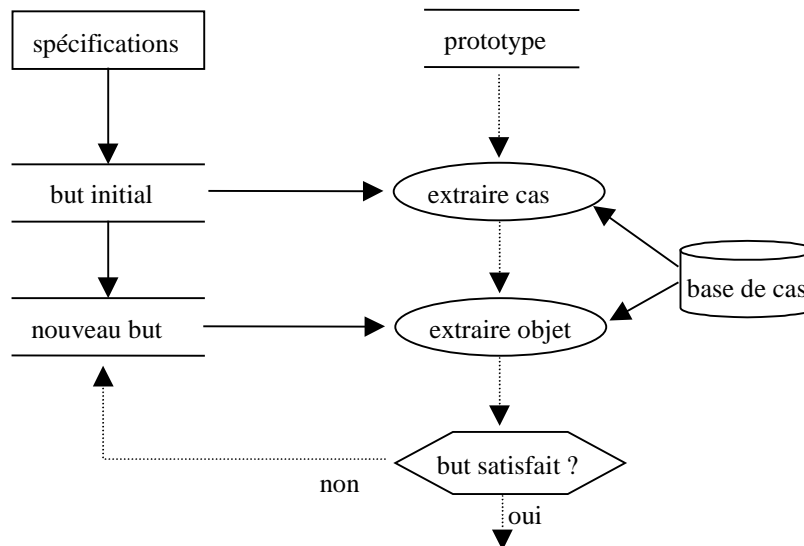


Fig.27 : enchaînement des différentes étapes dans EADOCS

Chiu définit une méthode « d'adaptation hybride ». Cette méthode combine des méthodes basées sur un modèle causal et des méthodes basées sur des règles d'ajustement de critères pour adapter la solution du cas sélectionné au nouveau problème. Pour cela, la méthode transforme la solution du cas source correspondant à ses données FMEA, la donnée FMEA étant définie par la description d'un défaut, la cause de ce défaut et une action corrective. Le modèle causal traite les relations entre les défauts possibles des composants du cas et leur processus de fabrication et les relations entre les défauts des processus et ceux de la machine. Les règles d'ajustement des critères s'occupent des données attachées aux relations entre les différents composants et des conditions de la machine.

On retrouve ici le même principe que dans le système précédent consistant à définir un sous-problème cible pour adapter le cas sélectionné en relançant le processus de remémoration. Mais il est ici simplifié par la représentation des problèmes : un problème est décomposable en sous-problèmes du niveau d'abstraction inférieur et chaque niveau utilise le même modèle.

Dans **Déjà-Vu**, une phase d'adaptation se déroule après chaque cycle de remémoration. Elle est réalisée de la même façon sur un cas d'un niveau abstrait que sur un cas d'un niveau concret. Elle utilise deux types de connaissances modélisées sous forme de règles qui sont appelées les « spécialistes d'adaptation » et les « stratégies d'adaptation ». Les spécialistes d'adaptation représentent l'ensemble des connaissances de transformation. Chaque spécialiste d'adaptation concerne une tâche d'adaptation spécifique correspondant à une modification locale et possède des connaissances déclaratives qui décrivent ses capacités particulières. Les stratégies d'adaptation sont là pour détecter et résoudre les conflits qui peuvent se présenter entre les différents spécialistes.

Elles possèdent des connaissances pour détecter les échecs dus aux interactions entre spécialistes et des méthodes pour résoudre les conflits.

II.4.2. Systèmes de planification

En planification à base de cas, on peut distinguer l'adaptation par génération et l'adaptation par transformation. L'adaptation par génération utilise un planificateur générateur, soit pour résoudre les buts non résolus par le(s) cas remémoré(s) comme dans **CAPLAN/CBC** et **PRODIGY/ANALOGY**, soit pour affiner une solution abstraite comme dans **PARIS**. Les connaissances et les techniques utilisées dans ce type d'adaptation relèvent plus de la planification que de l'adaptation d'un cas. Ci-après, nous présentons plus en détail l'adaptation par transformation de deux des systèmes de planification : Resyn/RàPC pour la planification linéaire de synthèse en chimie organique et le système de Prasad pour la planification hiérarchique de recettes de cuisine.

La phase de remémoration de **Resyn/RàPC** fournit un cas source associé à un ensemble de fonctions de transformation, si ce cas est trop éloigné du cas cible. Cet ensemble de fonctions permet de passer, par transformation, de la définition du problème du cas source (la molécule source) à la définition du problème du cas cible (la molécule cible) et il est appelé « chemin de similarité ». Dans l'exemple de la figure 28, le signe \approx représente la fonction de subsomption et le signe \Leftarrow représente une fonction de transformation.

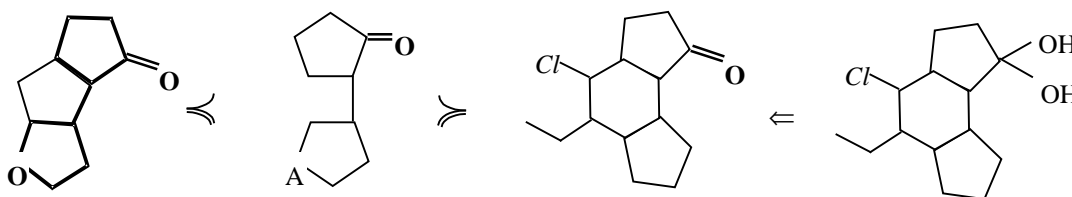


Fig.28 : exemple d'un chemin de similarité

Le processus d'adaptation du plan solution du cas source en un plan solution du cas cible utilise ce « chemin de similarité » et s'appuie sur trois opérations principales : la généralisation, l'instanciation et la transformation. La généralisation change le plan de synthèse d'une molécule donnée en un plan d'une molécule plus générale (suppression d'atome ou de liaison). L'instanciation transforme le plan d'une molécule donnée en un plan d'une molécule plus spécifique (ajout d'atome ou de liaison). La transformation modifie le plan d'une molécule donnée en un plan d'une molécule légèrement différente (remplacement d'un atome par un autre).

La phase d'adaptation est donc définie entièrement par l'ensemble de fonctions du chemin de similarité. Elle correspond en fait uniquement à la mise en œuvre du raisonnement choisi pour adapter le plan, ce raisonnement étant décidé pendant la phase de remémoration.

Dans le *système de Prasad*, une phase d'adaptation est réalisée après chaque cycle de remémoration pour modifier la liste d'opérateurs du niveau d'abstraction concerné. Les connaissances de modification sont représentées sous forme de règles « si-alors-sinon » et sont classées en fonction des propriétés des objets du domaine qu'elles concernent. Ces règles sont associées à un opérateur et réalisent des modifications pour adapter l'opérateur à l'objet du domaine. Lorsqu'il s'agit d'un opérateur abstrait, elles ajoutent ou retirent des opérateurs dans la décomposition de cet opérateur en sous-opérateurs. Lorsqu'il s'agit d'un opérateur de bas niveau, elles ajustent les valeurs de ses paramètres.

Cet exemple montre combien l'utilisation d'un même modèle pour représenter les connaissances de tous les niveaux d'abstraction est opportune. Ici les opérateurs de tous les niveaux du plan sont représentés avec le même modèle et toutes les connaissances d'adaptation sont modélisées par le même type de règle. Cela permet d'utiliser le même processus d'adaptation à chaque cycle, sans se soucier du niveau d'abstraction sur lequel on travaille.

II.4.3. Bilan sur la phase d'adaptation

L'intérêt de construire une nouvelle solution en combinant les solutions de plusieurs cas paraît évident, en particulier dans la planification hiérarchique où les décompositions en sous-problèmes sont associées à leurs solutions respectives. Pour cela, la phase d'adaptation doit être une phase de construction et doit disposer de connaissances permettant de vérifier la cohérence des combinaisons réalisées.

L'adaptation à l'aide de règles semble relativement facile à utiliser puisque l'on dispose des critères évalués de la définition du cas cible et du/des cas sélectionné/s pour remplir la partie prémisses. Mais d'une part, elle nécessite de pouvoir calculer l'écart entre la solution proposée et celle attendue. Or, si l'on sait comparer les problèmes pour sélectionner des cas proches, on ne sait pas toujours comparer les solutions, en particulier dans des domaines tels que le TI où il n'existe pas de fonctions d'évaluation fiables. D'autre part, pour être modélisées sous forme de règles, il faut que les connaissances d'adaptation puissent être décomposées en éléments suffisamment simples.

II.5. Mémorisation d'un nouveau cas

Peu de systèmes abordent réellement la phase de remémoration d'un nouveau cas qui fait appel à des techniques d'apprentissage et dont la mise en œuvre aborde deux principaux problèmes : quels cas stocker et comment les stocker. Un cas nouvellement résolu ne doit être stocké dans la base de cas que s'il apporte de nouvelles compétences au système car stocker tous les nouveaux cas conduirait la phase de remémoration à un goulot d'étranglement. Avant de déterminer comment stocker ce cas, il faut donc décider s'il est utile de le stocker. Dans la suite de ce paragraphe, nous montrons quelques exemples de mémorisation de cas s'appuyant sur les traces de résolution, sur l'abstraction de cas et sur des modèles.

II.5.1. Mémorisation à partir des traces de résolution

La décision de mémoriser un cas peut être prise en prenant en compte les traces de la résolution, en particulier, celles traitant les points résolus dans le nouveau cas qui ne l'étaient pas dans les cas de la base.

C'est ainsi que procède le système *SICT-WICT* où l'apprentissage est défini comme une tâche de haut niveau. L'intégration du nouveau cas est envisagée à la fin de chaque cycle du raisonnement par cas. Cette phase est réalisée par un apprentissage guidé par les échecs rencontrés lors de la résolution et peut faire appel à l'expert pour qu'il fournisse de nouvelles connaissances.

Le principe général utilisé est de mémoriser un cas uniquement s'il a nécessité un gros effort d'adaptation.

II.5.2. Mémorisation par abstraction

En planification, les techniques d'apprentissage de nouveaux cas sont souvent fondées sur une représentation du plan à plusieurs niveaux d'abstraction, car cette représentation permet de n'apprendre que la nouvelle partie du plan.

Dans le *système de Prasad*, des mécanismes de stockage sont associés à chaque opérateur du plan, quelque soit son niveau d'abstraction. Ce mécanisme stocke la nouvelle décomposition associée à l'opérateur à l'endroit approprié dans la hiérarchie correspondante. Les critères utilisés pour stocker un plan sont les mêmes que ceux utilisés pour la phase de remémoration.

Le système **PARIS** utilise également différents niveaux d'abstraction pour stocker les plans créés. Une abstraction de cas est réalisée automatiquement lors de la mémorisation. Un cas donné au niveau concret est abstrait à différents niveaux. Ce processus conduit à un ensemble de cas abstraits qui sont alors stockés dans la base de cas. Abstraire un cas consiste à réduire son niveau de détail dans la description du problème et dans la solution, ainsi un ensemble d'opérateurs concrets correspond à un seul opérateur abstrait. Cependant tous les cas ne sont pas intégrés : les cas trop particuliers sont laissés de côté.

La représentation sous forme de plans hiérarchiques utilisée dans ces deux systèmes, permet d'envisager la mémorisation sur chaque nœud du plan. On peut ainsi sélectionner les connaissances à intégrer à la base de cas à n'importe quel niveau de détail. En particulier, on peut ajouter uniquement un élément de stratégie (par exemple, une nouvelle décomposition) ou un élément de connaissance du domaine (par exemple, une valeur de paramètre).

II.5.3. Mémorisation à partir de modèles

Certains systèmes utilisent des modèles définis spécialement pour la mise en œuvre de la phase d'apprentissage. C'est le principe utilisé dans le système de Caulier et dans Déjà-vu.

Le *système de Caulier* utilise une approche basée conjointement sur l'acquisition des connaissances et le RàPC. Des modèles de connaissances du domaine ont été définis à l'aide de la méthodologie CommonKADS. Ces modèles sont utilisés par un module de capitalisation des connaissances (fig. 29) qui gère entièrement la base de cas et qui guide l'utilisateur lors la création d'un nouveau cas cible et lors du stockage d'un nouveau cas source.

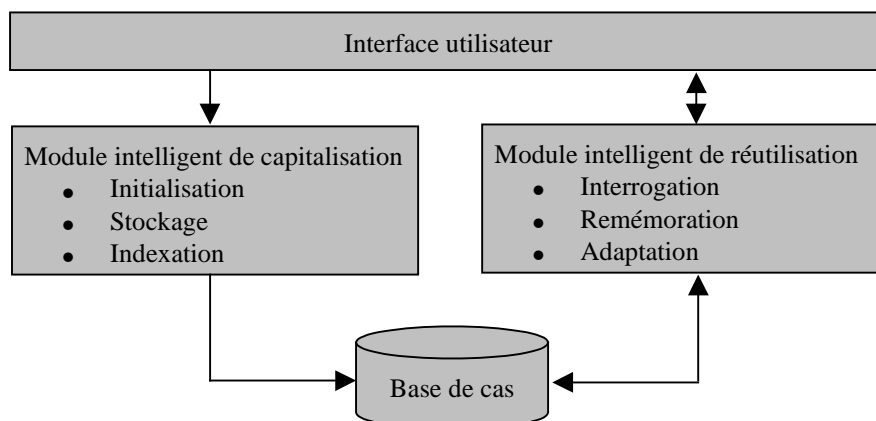


Fig.29 : représentation fonctionnelle du système de Caulier

Le système **Déjà-Vu** est un des seuls systèmes qui ait étudié entièrement le problème de la mesure de « l'utilité » d'un nouveau cas au sens des conséquences qu'entraîne son intégration ou sa

suppression sur les compétences du système. Pour cela, un « modèle de compétence » permettant de mesurer la compétence d'un cas en terme de recouvrement des autres cas a été défini. Les cas sont classés suivant des catégories de compétence correspondant à des configurations de recouvrement qui sont graduées en fonction de leur importance. Un cas est catégorisé en fonction de l'ensemble des cas dont il permet de résoudre le problème (recouvrement) et en fonction de l'ensemble des cas qui permettent de résoudre son problème (accessibilité). La catégorisation permet de supprimer des cas en fonction de leur importance. Les quatre catégories définies sont, par ordre croissant d'importance :

- les cas auxiliaires : cas dont l'ensemble de recouvrement est subsumé par l'ensemble de recouvrement d'un des cas de son ensemble d'accessibilité,
- les cas recouvrants : cas qui recouvrent un ensemble de régions de l'espace de problème couvertes indépendamment par d'autres cas,
- les cas supports : cas qui sont rassemblés en groupes à l'intérieur desquels chaque cas possède le même recouvrement que les autres,
- les cas pivots : cas qui ne sont accessibles par aucun autre cas.

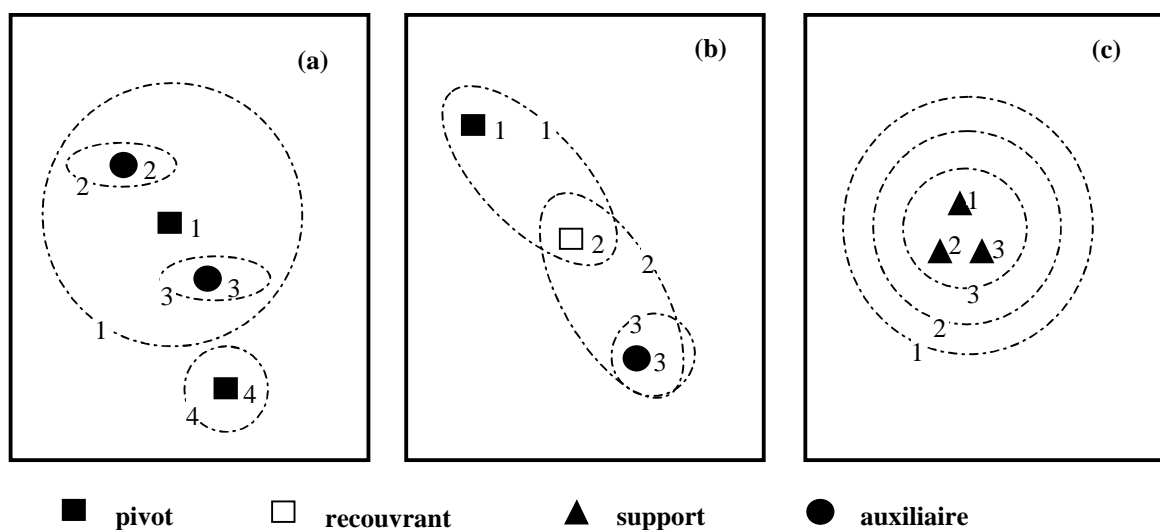


Fig.30 :exemples de catégories dans trois bases de cas

La figure 30 montre trois exemples de configuration d'une base de cas. Les lignes pointillées représentent l'ensemble de recouvrement d'un cas. Prenons quelques exemples explicatifs :

- exemple 1 : le cas 1 de la base (a) est un cas pivot car il n'est recouvert par aucun autre cas,
- exemple 2 : le cas 2 de la base (a) est un cas auxiliaire car tous les problèmes qu'il peut résoudre peuvent être résolus par le cas 1,

- exemple 3 : le cas 2 de la base (b) est un cas recouvrant car son ensemble de recouvrement peut être couvert en assemblant les ensembles de recouvrement du cas 1 et du cas 3,
- exemple 4 : les trois cas de la base (c) sont des cas support car ils possèdent tous les trois le même ensemble de recouvrement.

Dans ce système, on ne cherche pas seulement à savoir si le cas doit être intégré ou non à la base. Si le nouveau cas apporte de nouvelles compétences au système, on va rechercher si son intégration ne rend pas inutile certains anciens cas. La base de cas est donc toujours optimale : elle contient toutes les connaissances connues nécessaires mais uniquement ces connaissances.

II.5.4. Bilan sur l'apprentissage des cas

Une première technique consiste à comparer le nouveau cas à tous les cas de la base pour tester s'il doit être intégré. Cette technique permet de limiter le nombre d'ajouts mais elle s'avère vite insuffisante pour un système fonctionnant sur le long terme.

Une technique plus avancée consiste à déterminer un compromis entre le temps consacré à la recherche d'un cas et celui consacré à son adaptation en ne mémorisant que les cas ayant demandé un gros effort d'adaptation.

Enfin, les systèmes recherchant à toujours avoir une base de cas optimale, utilisent des techniques consistant à réviser entièrement la base de cas et son organisation à chaque fois que l'intégration d'un nouveau cas est envisagée.

On notera également que la représentation d'un cas par partie ou suivant plusieurs niveaux d'abstraction (en particulier en planification), permet de limiter considérablement l'intégration de nouvelles connaissances dans la base de cas.

II.6. Conclusion sur les systèmes de RàPC

L'étude présentée dans ce chapitre, nous a montré que le RàPC demandait en premier lieu une analyse précise du domaine d'application. Nous devons donc présenter une étude détaillée du TI pour déterminer l'ensemble des critères à prendre en compte et l'organisation de ces critères. Cette étude doit nous permettre de répondre, en particulier, aux questions suivantes :

- Quels sont les données qui caractérisent un problème de TI et un plan hiérarchique solution ?
- Quels sont les critères qui entrent en ligne de compte pour tous les cas ?

- Quels sont ceux qui sont optionnels (suivant le cas ou l'utilisateur) ?
- Quelle importance doit-on donner à un critère particulier par rapport aux autres ?

Il faut également déterminer les niveaux d'abstraction auxquels on veut représenter un cas : lorsque des couples sous-problème/sous-solution peuvent être déterminés à plusieurs niveaux d'abstraction, il peut être judicieux de définir plusieurs types de cas correspondant aux différents niveaux d'abstraction du domaine.

Il faut ensuite établir comment utiliser ces critères, c'est à dire spécifier une ou des fonctions de similarité et un algorithme de recherche des cas similaires.

Parmi les schémas proposés pour l'algorithme de recherche/adaptation, deux nous intéressent plus particulièrement : celui comportant une étape de réduction de l'espace de recherche et celui comportant une boucle de recherche/adaptation. Il serait donc opportun de définir un nouveau schéma respectant ces deux propriétés, d'une part en déterminant quels sont les critères qui permettent de réduire l'espace de recherche, et d'autre part, en définissant un cycle de recherche/adaptation s'appliquant sur les différents niveaux du plan.

Cet algorithme doit permettre de construire une solution en combinant plusieurs cas. On peut pour cela définir l'adaptation comme le remplacement d'une partie du cas sélectionné par des parties d'autres cas.

Enfin, on a vu qu'il était nécessaire de définir des mécanismes « intelligents » de mémorisation. En particulier, il faut éviter de stocker entièrement un nouveau cas, et plutôt tenter de sélectionner les connaissances nouvelles pour laisser celles qui sont déjà stockées (et peuvent être retrouvées) à partir d'autres cas.

III. Le système TMO

III.1. Motivations et objectifs

III.1.1. Nos motivations

En Traitement d'Image (TI), la mise au point d'une application est une activité complexe faisant intervenir deux catégories d'experts (fig. 31) : l'expert du domaine de provenance de l'image et l'expert en TI. Des difficultés surviennent pour définir formellement le problème (étape ❶), pour élaborer une solution (étape ❷) et pour évaluer les résultats (étape ❸).

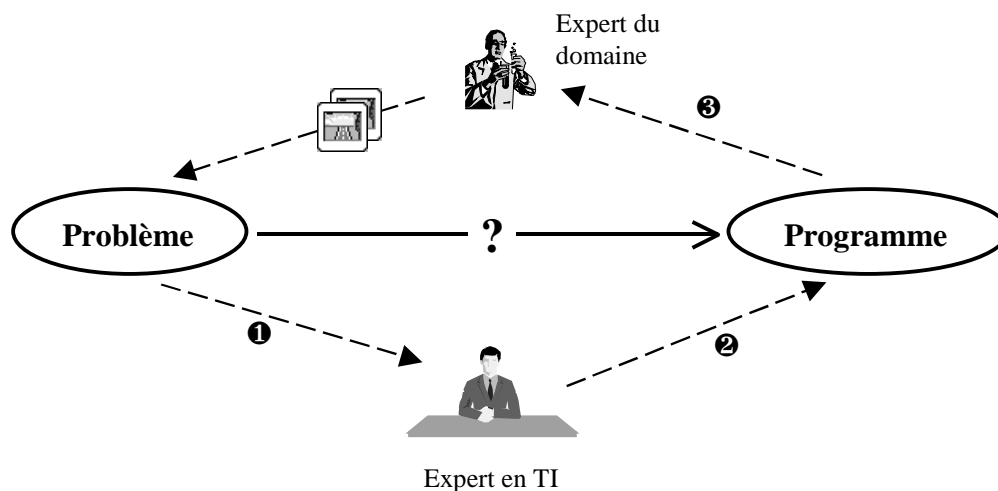


Fig.31 : mise au point d'une application de TI

Les connaissances mises en œuvre tout au long de cette activité sont difficiles à expliciter et à formaliser. Le savoir accumulé par l'expert en TI est non structuré et extrêmement diffus. Il n'existe pas de théorie sur laquelle s'appuyer pour modéliser le domaine, ce qui contraint l'expert à pratiquer une démarche relativement empirique. Celui-ci n'est souvent pas capable d'expliquer toutes les phases de son raisonnement, et ce, pour deux raisons essentielles. En premier lieu, les problèmes se posent généralement selon des termes du domaine d'application qu'il faut donc d'abord traduire en termes du TI (étape ❶), et ils ne sont pas spécifiés complètement au départ. En second lieu, la connaissance sur les algorithmes est imprécise et peu fiable. Il n'existe pas non plus de fonction quantitative d'évaluation des résultats. Tout ceci justifie le recours à beaucoup d'étapes d'essai-erreur et d'optimisation. Les objectifs initiaux doivent souvent être revus et complétés au cours de la résolution, ce qui nécessite l'intervention des deux types d'experts.

L'expert du domaine de provenance de l'image a un rôle important à jouer lors de la mise au point d'une application de TI. Tout d'abord, il définit le problème, pour lequel l'expert en TI doit

construire une solution correspondant à un programme (étape ②), en fournissant une ou plusieurs images (les objets à traiter) et une requête (les objectifs à atteindre). Puis lorsqu'une solution lui est proposée, il doit l'évaluer en visualisant les images résultats (étape ③). En effet, le TI est un domaine où il n'existe pas de fonction d'évaluation idéale et seul l'expert du domaine peut juger de la validité finale d'un résultat. Cette validation se fait, soit visuellement, soit en utilisant les résultats obtenus dans un protocole de test plus global (ex : statistiques sur les objets extraits de l'image). Les rôles de l'expert du domaine et de l'expert en TI lors de l'élaboration d'une application sont donc dépendants ; cela se traduit par des besoins de coopération et de communication entre ces experts.

Par ailleurs, la multiplicité et la fragilité des solutions accroissent les difficultés lors de la mise au point d'une application. Un problème donné possède généralement plusieurs solutions, plus ou moins adaptées à l'image à traiter et plus ou moins sensibles au type et à la quantité de bruit présent dans cette image. Pour pallier ces difficultés, on introduit souvent des informations a priori sur les résultats attendus et l'on teste plusieurs stratégies de traitement. Les informations a priori sont des connaissances que l'on a sur le domaine de l'image (« les noyaux de cellules en cytologie sont des objets convexes ») ou que l'on obtient par observation de la scène (« les noyaux de cellules sont des objets foncés posés sur un fond clair »).

Ce constat sur les difficultés du TI nous conduit à proposer un système d'aide aux utilisateurs dont nous allons maintenant détailler les objectifs.

III.1.2. Objectifs de notre système

La formalisation des connaissances de TI est une étape incontournable pour leur capitalisation mais elle doit aussi être un support de communication privilégié pour les différents acteurs. Il s'agit essentiellement de faciliter le dialogue entre l'expert du domaine et l'expert en TI et de favoriser la coopération et le partage des connaissances entre plusieurs experts en TI. Dans ce cadre, le premier objectif de notre système est la formalisation des connaissances mises en jeu. Ce système doit permettre d'acquérir et d'intégrer les connaissances au fur et à mesure de leur mise en évidence par la création d'applications. Pour cela, nous proposons un environnement interactif de construction et d'exécution d'applications de TI. Le second objectif de notre système est d'aider à la réutilisation des connaissances acquises dans le but de gagner du temps lors de la mise au point d'une application et de faire partager les connaissances de différents experts. Ce second objectif, tout aussi important que le premier, n'est pas détaillé ici, mais la totalité du chapitre IV lui est consacrée.

Le type d'approche que nous avons choisi pour la mise au point d'applications de TI s'appuie sur l'exploitation « intelligente » de bibliothèques de programmes, appelés opérateurs de TI. Suivant cette approche, construire une application de TI consiste à enchaîner et à paramétrer un ensemble d'opérateurs d'une bibliothèque donnée [Ficet 97]. Cette approche est de plus en plus utilisée en TI, d'une part parce qu'elle permet à un traiteur d'image de concevoir son application sans se soucier des détails de programmation (un expert en TI n'est pas toujours expert en programmation), d'autre part, parce que les algorithmes correspondant aux opérateurs sont les seules connaissances pour lesquelles il existe un consensus dans la communauté du TI.

Les bibliothèques de programmes ne sont pas utilisées uniquement dans le domaine du TI. Parmi les systèmes développés à l'aide de l'environnement SCARP [Willamowski 94b], beaucoup résolvent les problèmes en créant des enchaînements de programmes élémentaires répertoriés dans des bibliothèques (ex : le système Myosis développé pour le diagnostic électromyographique et le système Said mis au point pour le diagnostic vibratoire de plate-forme en haute mer).

De plus, on a vu au chapitre I.3, que les planificateurs automatiques de TI comme les environnements de programmation graphique utilisent de telles bibliothèques. On a également insisté sur le fait que les premiers permettaient peu à l'utilisateur d'intervenir dans la résolution (en particulier ces systèmes ne peuvent pas acquérir de connaissances auprès de l'utilisateur de façon interactive) et que les deuxièmes ne permettaient pas d'expliquer ni de modéliser le raisonnement. Dans le but de bénéficier des avantages dus à l'interactivité des environnements de programmation graphique, notre système doit permettre à l'utilisateur de sélectionner et d'enchaîner des opérateurs de TI. Mais il doit également lui donner la possibilité de modéliser et d'expliquer le raisonnement qui l'a amené à cet enchaînement, dans le but de réutiliser la stratégie mise en œuvre, à l'instar des systèmes automatiques.

Par ailleurs, une application réelle peut nécessiter l'enchaînement de plusieurs dizaines d'opérateurs. Pour expliciter le raisonnement utilisé lors de la mise au point de cet enchaînement, nous proposons de modéliser les applications sous forme de plans. Un plan de TI est obtenu par décomposition hiérarchique du problème posé en problèmes plus simples. Chaque problème ou sous-problème est associé à une tâche de TI, qui, suivant son niveau dans le plan, exprime le but recherché, la technique à employer ou l'algorithme à appliquer. Comme nous l'avons vu au chapitre I, cette modélisation sous forme de tâches génériques et décomposables est largement répandue en résolution de problème, mais peu appliquée en TI. D'une part la représentation des connaissances stratégiques à différents niveaux d'abstraction fournit une première forme

d'explicitation du raisonnement ; d'autre part, la notion de tâche se situe à un niveau compréhensible par l'utilisateur et regroupe des connaissances procédurales et des connaissances sémantiques, ce qui facilite la communication système/utilisateur.

Un plan de TI peut être schématisé sous forme d'arbre (fig. 32). Il représente non seulement l'enchaînement de l'exécution d'opérateurs de TI correspondant aux feuilles de l'arbre, mais également le raisonnement nécessaire à la création de cet enchaînement, correspondant à des tâches de TI schématisées par les rectangles gris.

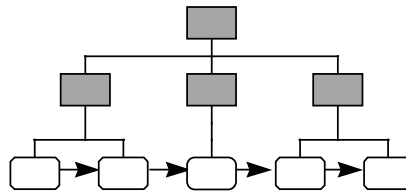


Fig. 32 : représentation d'un plan de TI

En s'appuyant sur cette représentation par plan hiérarchique, notre système interactif offre les trois fonctionnalités suivantes (fig. 33):

- le stockage d'applications dans une base de plans,
- la création interactive d'une application de TI,
- l'exécution interactive d'une application.

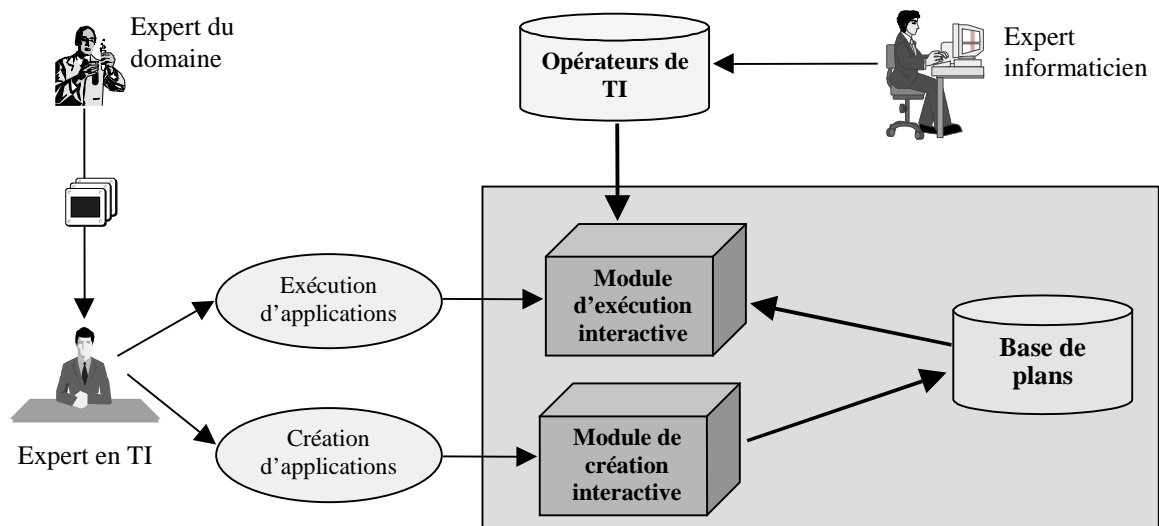


Fig.33 : système interactif de construction et d'exécution d'applications

A partir des images et de la requête fournies par l'expert du domaine d'application, l'expert en TI construit un plan de traitement. Avec notre système, il n'a plus à faire de programmation au sens

classique : il programme au « niveau connaissance » [Newell 82] en utilisant des blocs prédéfinis. Contrairement aux environnements de programmation graphique classiques, dans lesquels l'utilisateur doit obligatoirement adopter une démarche ascendante lors de la construction de son plan, dans notre système, il est possible de choisir entre une démarche ascendante (par regroupement d'opérations), descendante (par décomposition de problème en sous-problèmes) ou mixte. Cette dernière démarche correspond bien à celle du traiteur d'images qui procède volontiers de manière ascendante pour des parties de traitement qu'il connaît bien et de manière descendante pour d'autres. Il a également la possibilité de modéliser plusieurs techniques pour résoudre un même problème ou sous-problème, de façon à pouvoir tester et comparer différentes solutions.

Le traiteur d'images peut ensuite exécuter son plan en choisissant dynamiquement entre les techniques modélisées et ainsi évaluer les résultats en collaboration avec l'expert du domaine. Puis il a la possibilité de modifier son plan jusqu'à l'obtention de résultats satisfaisants. La modélisation sous forme de tâches décomposables facilite cette phase de mise au point, en particulier en fournissant un support de communication entre experts et en autorisant la visualisation des résultats intermédiaires.

Sur le schéma de la figure 33, nous avons également représenté l'expert informaticien. Un expert en TI n'ayant pas suffisamment de connaissances en programmation peut faire appel à lui pour implanter de nouveaux opérateurs de TI qui seront intégrés dans la bibliothèque. C'est également cet expert qui met en œuvre les différentes fonctionnalités du système. Chaque fonctionnalité correspond à une tâche de contrôle des connaissances du domaine. Pour conserver un caractère évolutif à notre système, les connaissances de contrôle sont également modélisées sous forme d'arbres de tâches (qui seront détaillés au chapitre III.3). Ainsi, l'utilisateur ayant les compétences requises peut définir de nouvelles fonctionnalités en utilisant le même principe que pour les applications de TI. Par exemple, un expert informaticien pourra définir les plans de contrôle nécessaires au module de réutilisation par assemblage de briques de base.

Dans la deuxième section de ce chapitre, nous présentons les différents niveaux de connaissances qui constituent l'architecture du système et leur articulation, puis dans la troisième section nous décrivons en détail le modèle TMO utilisé pour la représentation des connaissances de chaque niveau.

III.2. Architecture du système TMO

III.2.1. Une architecture à trois niveaux

Notre système possède trois niveaux de connaissances : le niveau des connaissances du domaine (le TI) et les niveaux des connaissances de contrôle et de métacontrôle pour gérer la modélisation et l'utilisation des connaissances du domaine.

D'après [Clouard 95], au niveau des **connaissances du domaine**, on distingue à nouveau trois types de connaissances :

- des connaissances sur l'application : elles permettent de donner un sens et un sujet à l'image. Ce sont les connaissances a priori fournies par l'expert du domaine qui servent à la définition du problème (par exemple, la description des objets recherchés),
- des connaissances en TI : elles permettent la détermination des stratégies à utiliser pour accomplir les tâches et évaluer les résultats,
- des connaissances sur les opérateurs : elles permettent leur sélection, la détermination des valeurs de paramètres et la réalisation d'enchaînements syntaxiquement corrects.

Les interactions entre ces trois types de connaissances sont symbolisées par la figure 34.

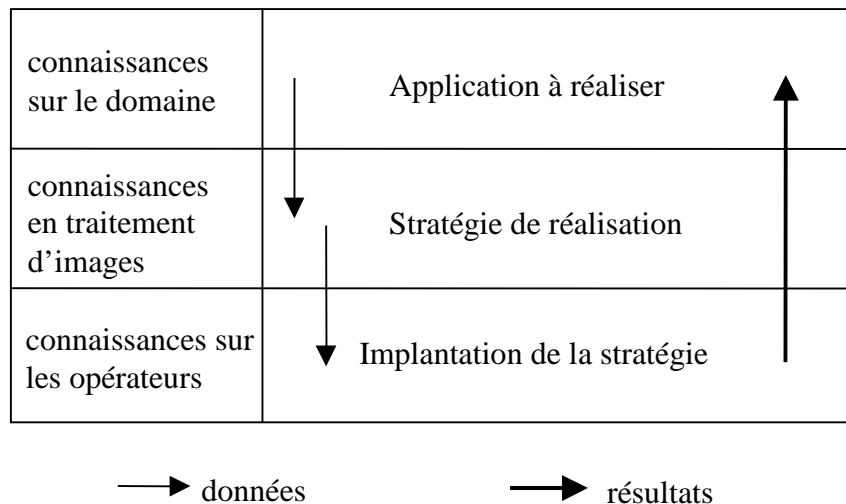


Fig.34 : interactions entre les trois types de connaissances du domaine

Les **connaissances de contrôle** sont les connaissances qui facilitent l'exploitation des connaissances du domaine. Ce sont elles qui gèrent la résolution des problèmes liés à l'application et leur explicitation. On peut séparer ces connaissances en deux catégories :

- le contrôle du domaine concerne la gestion des plans de TI (leur création, leur modification, leur exécution, ...), il correspond aux différentes fonctionnalités proposées à l'utilisateur,
- le contrôle du système gère les opérations concernant le bon fonctionnement d'une session, telles que l'initialisation, l'affichage des opérations à effectuer, la sélection des traitements accessibles à l'utilisateur, le contrôle des enchaînements, ...

Les **connaissances de métacontrôle** sont les connaissances sur le contrôle du contrôle. Elles décident quelles sont les tâches de contrôle qui doivent être réalisées et comment elles doivent l'être. Elles définissent en particulier les règles de comportement du système, vis à vis de l'utilisateur. En effet, le contrôle sera différent suivant le type de l'utilisateur : par exemple un expert informaticien peut intégrer des connaissances de contrôle pour définir une nouvelle fonctionnalité, alors qu'un expert en TI ne peut intégrer que des connaissances du domaine. Cette définition des connaissances de métacontrôle correspond à celle du niveau méta donnée par Maes et par Pitrat. Maes [Maes 87] définit un méta-système comme étant un système informatique dont le domaine d'application est un autre système informatique. De même, Pitrat [Pitrat 90] définit les métaconnaissances comme étant des connaissances sur les connaissances, qu'elles concernent les propriétés des connaissances ou leur manipulation.

III.2.2. Articulation des trois niveaux

Les connaissances de chacun des trois niveaux peuvent être vues comme des traitements opérés sur des objets du niveau inférieur (fig. 35). Le métacontrôle effectue des traitements sur le contrôle (réaliser une tâche de contrôle, sélectionner une tâche de contrôle, ...) et le niveau contrôle effectue des traitements sur le domaine (choisir une tâche du domaine, réaliser une tâche du domaine, ...).

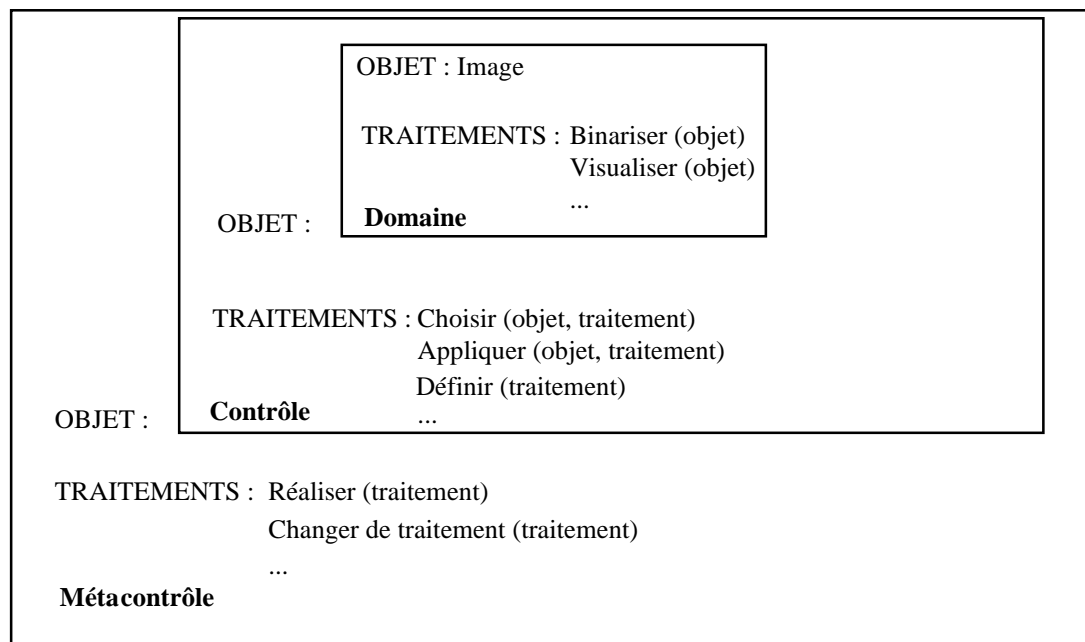


Fig.35 : exemple de représentation des trois niveaux

Cependant, pour le niveau contrôle, cette représentation d'un niveau agissant sur les connaissances du niveau inférieur présente des exceptions. En effet, si les connaissances du « contrôle domaine » traitent bien des connaissances du niveau « domaine », ce n'est pas toujours le cas pour les connaissances du « contrôle système » qui peuvent opérer sur des connaissances du niveau domaine comme sur des connaissances de leur propre niveau. En fait, si l'on se place du point de vue des traitements, ces connaissances se trouvent au niveau contrôle, et si l'on se place du point de vue des données traitées, ces connaissances se classent au niveau métacontrôle. Ceci illustre la difficulté de distinguer les connaissances des différents niveaux et renforce l'intérêt d'avoir une représentation uniforme pour les trois niveaux afin d'éviter une différenciation inefficace.

III.3. Modélisation des connaissances TMO

Les connaissances des trois niveaux sont représentées uniformément à l'aide du modèle TMO [Ficet 96] qui définit un traitement sous forme de plan hiérarchique à l'aide de Tâches, de Méthodes et d'Outils. A un but à atteindre, nous associons une tâche, à un savoir-faire nous associons une méthode qui spécifie comment une tâche est accomplie et à la description d'un code informatique nous associons un outil. Par la suite, nous nous appuierons sur la convention proposée figure 36 pour représenter graphiquement ces trois notions : une tâche est modélisée par un rectangle, une méthode par une ellipse et un outil par un rectangle à coins arrondis. Le modèle TMO offre des

caractéristiques intéressantes pour nos objectifs, en particulier celles d'être compréhensible par l'utilisateur tout en étant opérationnalisable. Il a déjà été validé dans d'autres domaines tels que la conduite de projet [Moire 94] et la sélection de variétés de céréales [Deslandes 94]

BUT OU SOUS-BUT	Tâche
SAVOIR-FAIRE	Méthode
RESSOURCE	Outil

Fig.36 : convention de représentation graphique des connaissances

III.3.1. Notions de Tâche, Méthode et Outil

Tâche

Une tâche est la représentation d'un but ou d'un sous-but dans le système. Elle décrit un but à atteindre et rassemble les éléments nécessaires à l'atteinte de ce but : les données à traiter, les types de résultat à produire et les méthodes de résolution connues. Quand une tâche décrit un problème général, elle se décompose en sous-tâches qui décrivent des problèmes plus élémentaires; ces sous-tâches pouvant être décomposées à leur tour. Une tâche peut être résolue de plusieurs manières, on lui associe donc une ou plusieurs méthodes.

Les tâches servent à représenter tous les buts du système, quelque soit leur niveau :

- niveau « domaine » : les tâches représentent un objectif de TI ou une sous-partie de celui-ci. Par exemple, *isoler les objets du fond* est une tâche principale du domaine et *éliminer le fond* en est une sous-tâche.
- niveau « contrôle domaine » : les tâches représentent les opérations que l'on veut effectuer sur les plans de TI. *exécuter un plan* et *sauvegarder un plan* sont des tâches de ce niveau, *charger un plan* est une sous-tâche de *exécuter un plan*. Ces tâches du niveau contrôle peuvent être réalisées en interaction avec l'utilisateur.
- niveau « contrôle système » : les tâches représentent les opérations de contrôle du bon déroulement des opérations. *initialiser le système* et *gérer l'interface du système* sont des tâches du contrôle système. Ces tâches du niveau contrôle ne peuvent être exécutées que par le système.

- niveau « métacontrôle » : les tâches représentent les opérations de contrôle sur le contrôle. *exécuter une tâche* et *choisir une méthode* sont des tâches du métacontrôle. On parlera alors de métatâches.

Méthode

Une méthode spécifie comment une tâche peut être accomplie. Chaque méthode est associée à une seule tâche, mais une tâche peut être associée à plusieurs méthodes (fig. 37).

Pour cela, on cherchera à expliciter, pour chaque méthode, quand il est possible et réellement souhaitable de l'utiliser. Le choix de la méthode à utiliser peut être fait, soit par l'utilisateur (tâche du domaine), soit par le système (tâche de contrôle).

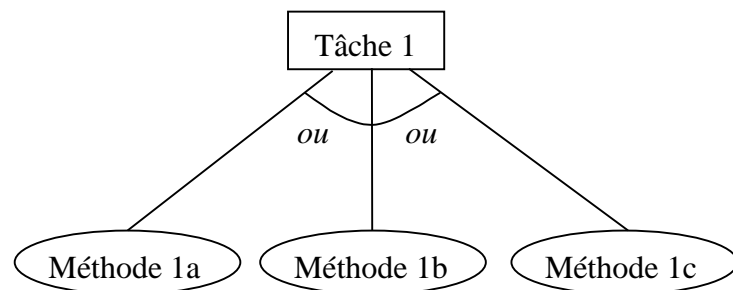


Fig.37 : plusieurs méthodes associées à une même tâche

Le corps de la méthode peut prendre deux formes :

- une décomposition en sous-tâches qui prend alors l'aspect d'un arbre "PUIS" (fig. 38), on parle alors de « méthode complexe ». Si T1 se décompose en (PUIS T11 T12 T13) par la méthode M1, on exécutera d'abord T11, puis T12, et enfin T13.

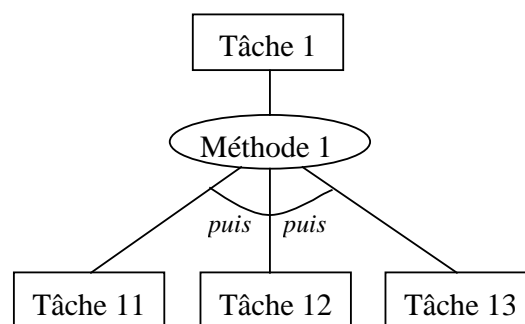


Fig.38 : méthode complexe

- l'appel à un code informatique par l'intermédiaire d'un outil (fig. 39), on parle alors de « méthode terminale ».

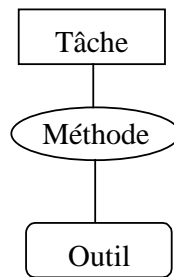


Fig.39 : méthode terminale

L'ensemble des sous-tâches ou des outils à exécuter pour réaliser une tâche est dépendant de la méthode choisie. Ce sont donc les méthodes qui vont gérer les flux de données entre tâche et sous-tâche et entre tâche et outil.

Par la suite, tous les nœuds représentant plusieurs méthodes pour une même tâche étant des nœuds « ou » et tous les nœuds représentant une décomposition en sous-tâches par une méthode étant des nœuds « puis », nous ne noterons plus ces relations entre les différents fils d'un nœud de façon à simplifier les schémas.

Outil

Un outil est la réification d'un code informatique, c'est-à-dire sa représentation pour l'utilisateur en termes conceptuels (but, entrées, paramètres, ressources utilisées, ...) avec un lien sur le code permettant sa mise en œuvre. Le code informatique qui lui est associé est vu par l'utilisateur comme une boîte noire dont il connaît seulement la transformation qu'elle effectue sur les entrées pour produire les sorties et dont il peut régler les paramètres. L'outil est là pour expliciter les opérations effectuées par le code qui lui est associé, il doit connaître les types des entrées/sorties et la signification exacte de l'opération que le code effectue sur les entrées.

Au niveau contrôle, nos outils font appel à des fonctions LISP ou C agissant sur le système ou sur les tâches du domaines.

Au niveau domaine, ils sont reliés à des opérateurs de TI de la bibliothèque PANDORE [Clouard 97] développée au GREYC. Un opérateur de notre bibliothèque est un programme de TI qui correspond à une opération non décomposable : une opération complexe doit être réalisée par l'enchaînement de différents opérateurs. Un opérateur se présente sous la forme d'un programme exécutable, qui prend en entrées des images, qui fournit en sortie des images et/ou un résultat numérique et dont le comportement peut être modifié par des paramètres à valeurs numériques. Un opérateur est polymorphe, c'est à dire qu'il est capable d'exécuter le traitement souhaité sur différents types d'images (image de pixels, image d'étiquettes, carte de régions, ...) sans que l'on ait

à spécifier la structure de donnée manipulée. Le développement de la bibliothèque ayant été fait à l'aide d'un langage orienté objet, la fonction à appliquer est sélectionnée automatiquement en fonction du type des images d'entrées. Chaque opérateur est masquable, c'est à dire qu'il peut travailler, soit sur les images entières, soit uniquement sur certaines parties spécifiées par un masque. Cette technique du masquage permet de focaliser les traitements sur certaines parties des images. Par ailleurs, la bibliothèque reste ouverte vis-à-vis du développement de nouveaux opérateurs, en fonction du domaine d'application ou des stratégies adoptées pour traiter les images.

Opérationnalisation du modèle

Le modèle TMO est un modèle opérationnel. L'exécution d'un plan "Tache - Méthode - Outil" entraîne l'exécution d'une suite d'outils, donc d'une suite de codes informatiques (fig. 40). On peut donc interpréter l'exécution du graphe comme étant l'exécution de cette suite de codes informatiques. Cette suite étant obtenue par une succession de choix dans le graphe, il s'agit bien ici d'exécution de programme avec « choix dynamiques ».

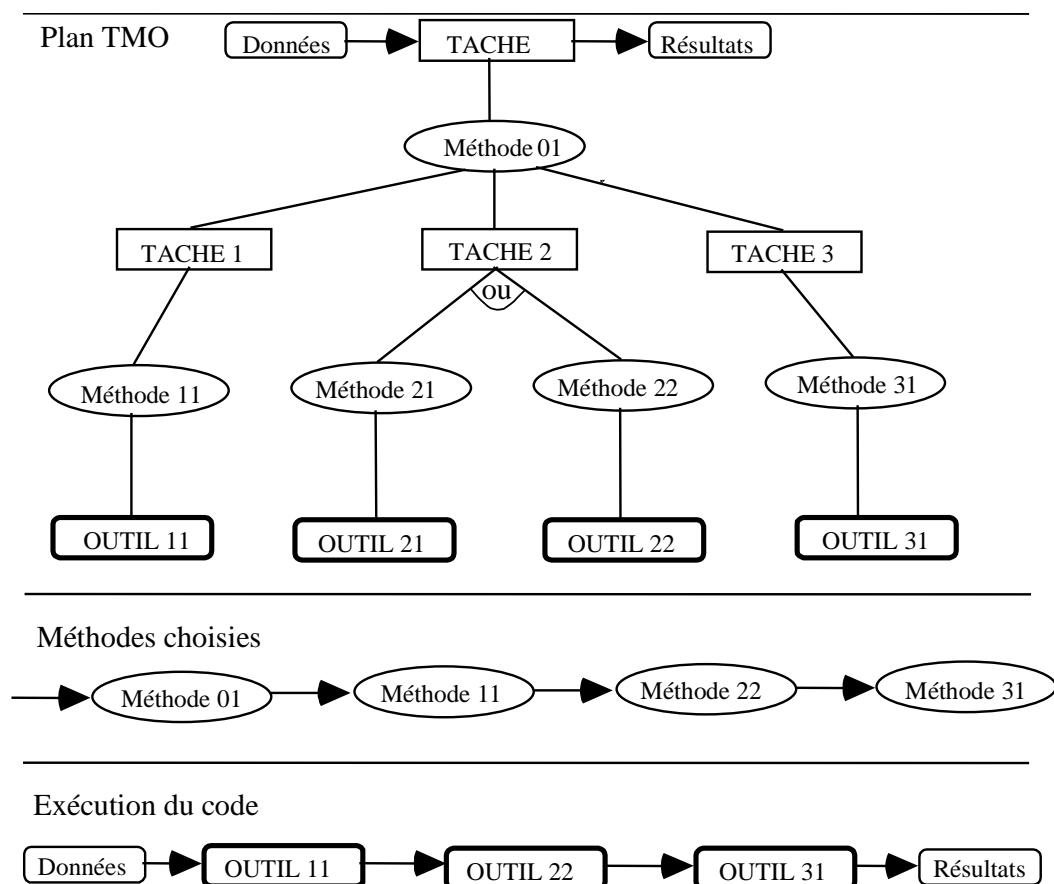


Fig.40 : exécution de programme avec choix dynamiques

III.3.2. Représentation de l'architecture à l'aide du modèle

Pour un niveau donné, l'application d'un traitement sur une donnée peut être modélisée par une tâche réalisée sur une/des entrée/s, produisant une/des sortie/s, suivant une des méthodes possibles (fig. 41).

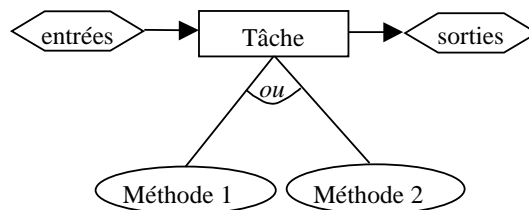


Fig.41 : modélisation TMO d'un traitement

Suivant le même principe, l'application du modèle TMO à notre architecture à trois niveaux est schématisée par la figure 42.

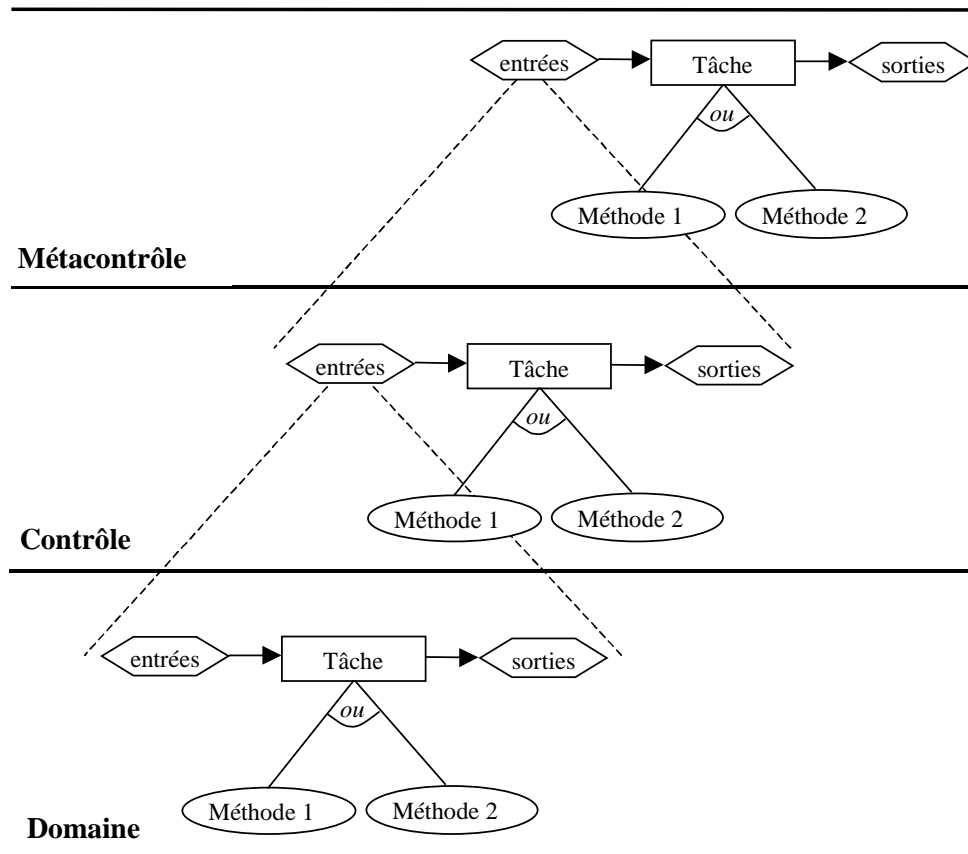


Fig.42 : application du modèle TMO à l'architecture trois niveaux

Les données traitées (entrées et sorties) par les tâches de métacontrôle sont des tâches de contrôle et les données traitées par les tâches de contrôle sont des tâches du domaine.

L'ensemble des tâches (quelque soit leur niveau) est géré par un agenda qui contient la liste des tâches à exécuter. Une boucle de métacontrôle retire une tâche de la liste et l'exécute tant que

l'agenda n'est pas vide. L'exécution d'une tâche est réalisée par la métatâche *exécuter tâche* détaillée ci-après. Nous présentons maintenant des exemples d'arbres de tâches pour chacun des trois niveaux, ce qui nous permet ensuite d'illustrer l'articulation entre les niveaux sur un exemple concret.

III.3.2.1 Exemple de tâche de métacontrôle

La figure 43 présente la tâche de métacontrôle *exécuter tâche* qui permet d'exécuter une tâche T de n'importe quel niveau. Comme nous venons de le voir, c'est la métatâche la plus importante. Deux méthodes sont disponibles : *exécution interactive* pour une tâche du domaine et *exécution automatique* pour une tâche de contrôle. Chacune de ces méthodes décompose la tâche *exécuter tâche* en sous-tâches qui vont d'abord sélectionner (interactivement ou automatiquement) parmi les méthodes de la tâche T celle à appliquer, puis exécuter l'outil correspondant ou mettre les sous-tâche dans un agenda qui gère l'ordre d'exécution des tâches.

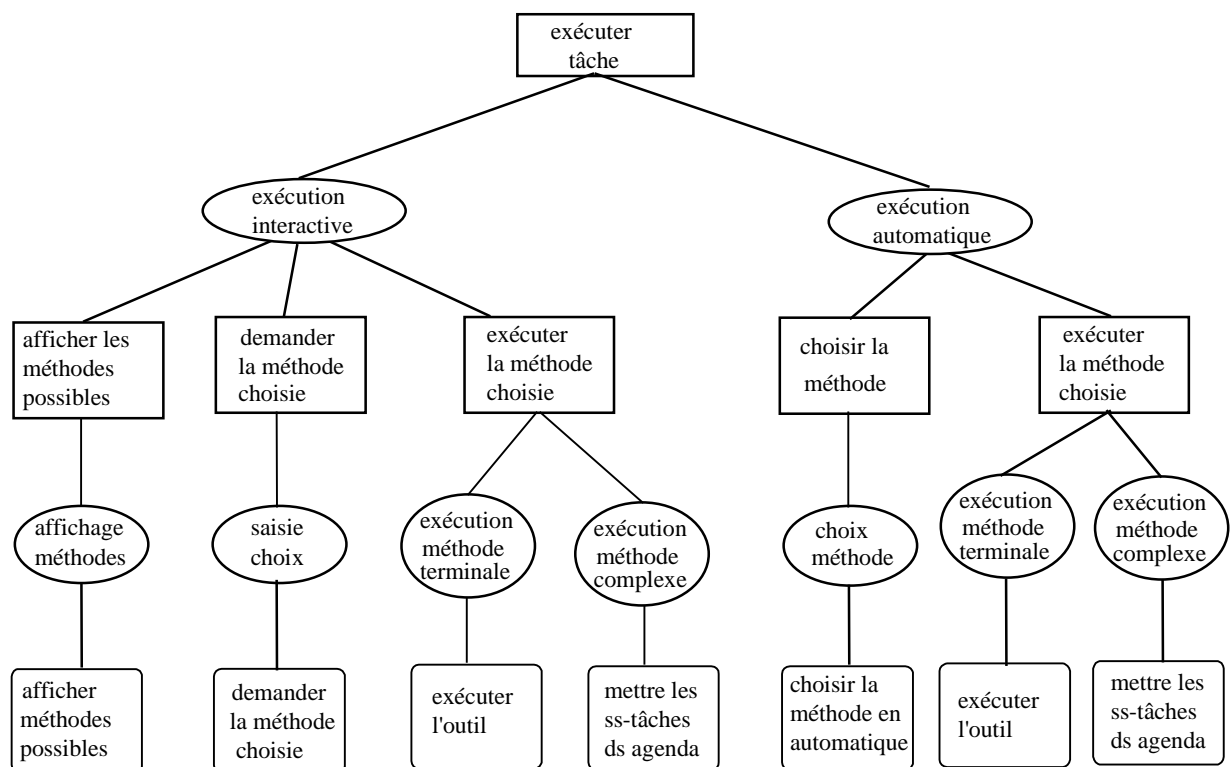


Fig.43 : la tâche de métacontrôle « exécuter tâche »

III.3.2.2 Exemple de tâche de contrôle

La figure 44 présente la tâche de contrôle *exécuter plan* qui permet d'exécuter un plan de TI prédéfini. Cette tâche est décomposée en trois sous-tâches qui vont charger les objets LISP

composant le plan, afficher l'arbre de tâches correspondant au plan puis mettre la tâche principale dans l'agenda pour qu'elle soit exécutée par la métatâche *exécuter tâche*.

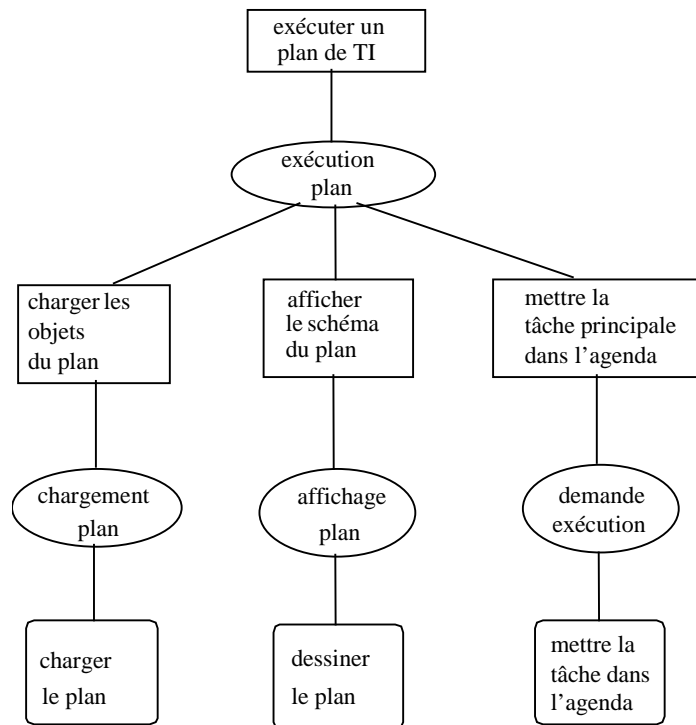


Fig.44 : la tâche de contrôle « exécuter un plan de TI »

III.3.2.3 Exemple de tâche du domaine

La figure 45 présente la tâche du domaine *isoler les objets du fond*. La décomposition en sous-tâches n'est donnée que partiellement : le plan comporte en réalité une quarantaine de tâches et une trentaine d'outils. La tâche racine *isoler les objets du fond* est décomposée en deux sous-tâches *éliminer le fond* (de l'image) et *former les objets à partir des régions*, qui sont décomposées à leur tour jusqu'à l'obtention de tâches réalisables par des outils. La sous-tâche *séparer les régions* est un exemple de tâche qui comporte deux méthodes : *séparation totale* si l'on veut obtenir des régions correspondant chacune à une cellule et *séparation partielle* si l'on veut obtenir des régions correspondant à des amas de cellules. Le choix entre ces deux méthodes sera fait par l'utilisateur au moment de l'exécution de cette tâche.

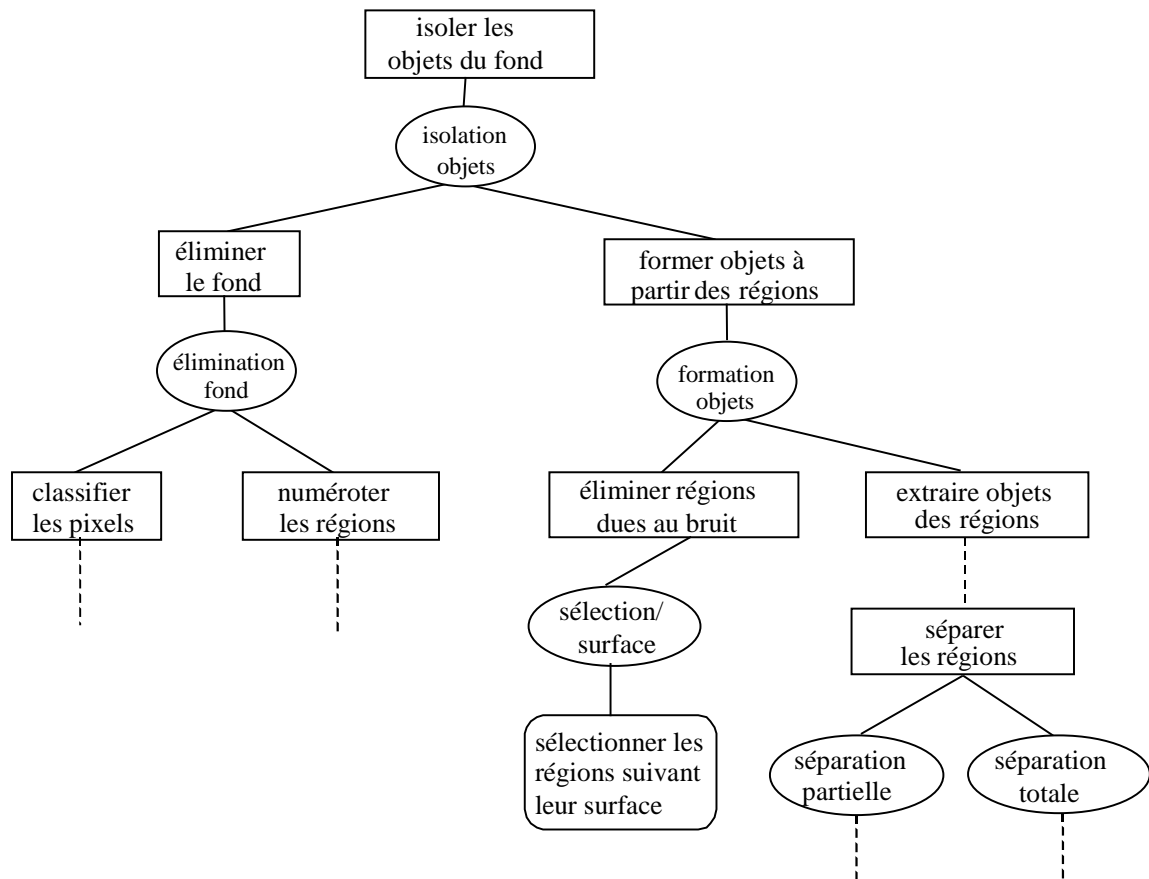


Fig.45 : la tâche du domaine « isoler les objets du fond »

III.3.2.4 Exemple d'articulation entre les trois niveaux

La figure 46 montre l'articulation entre les trois exemples de tâches précédents. La métatâche *exécuter tâche* a pour entrée la tâche de contrôle *exécuter plan*. La métatâche exécute la tâche *exécuter plan* qui a pour entrée la tâche du domaine *isoler les objets du fond*, c'est à dire qu'elle exécute ses sous-tâches (*charger les objets du plan*, *afficher le schéma du plan*, *mettre la tâche dans l'agenda*). Puis elle exécute la tâche du domaine *isoler les objets* du fond qui se trouve à ce moment dans l'agenda.

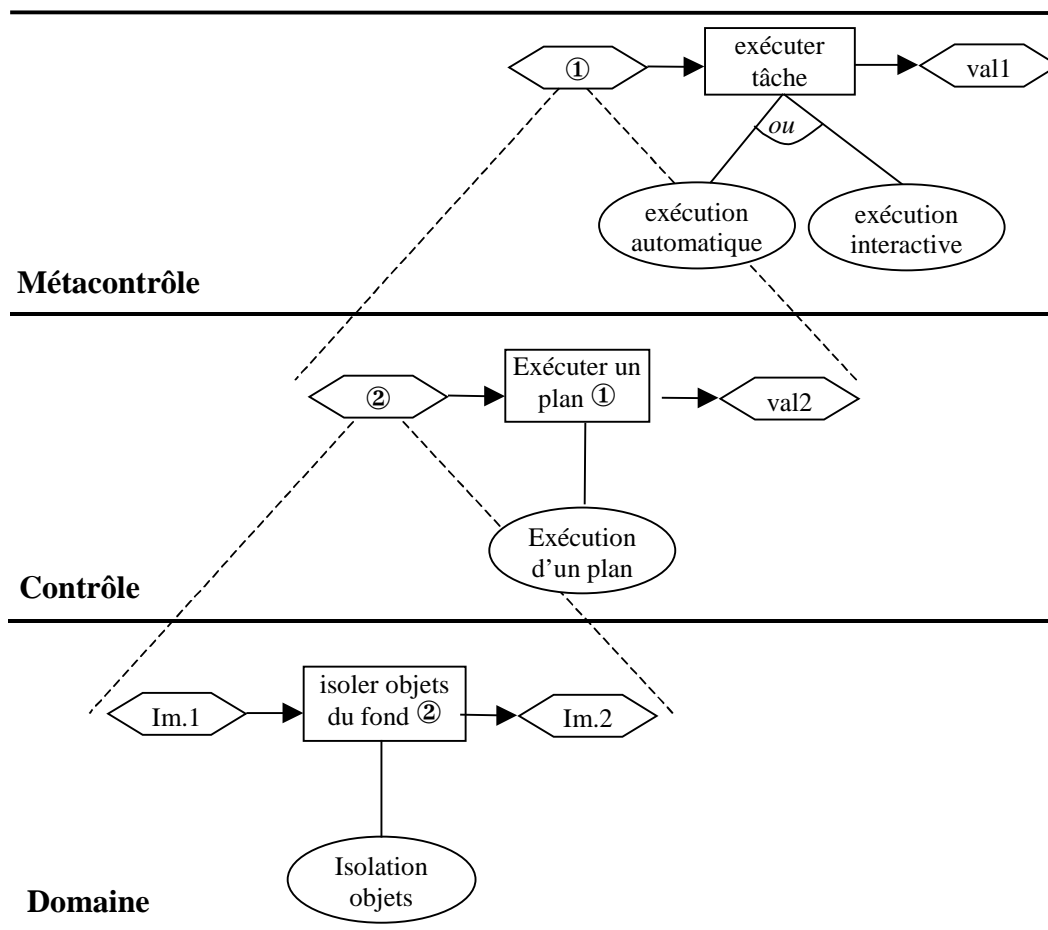


Fig.46 : exemple d'articulation entre les trois niveaux

III.3.3. Intérêt du modèle

Le modèle TMO répond parfaitement à nos besoins. En particulier, il possède un certain nombre des caractéristiques qui sont apparues comme indispensables pour la définition de notre système lors de l'étude présentée dans le chapitre I :

- c'est un modèle compréhensible par l'utilisateur et directement opérationnalisable : la notion de tâche décomposable se situe à un niveau de description accessible à l'utilisateur et les liens définis entre les tâches et des outils faisant appel à des codes informatiques en font une représentation opérationnelle,
- il sépare les connaissances portant sur les objets manipulés, les stratégies et les actions : les objets manipulés sont les entrées/sorties des tâches et outils, les stratégies sont détenues par les méthodes (une méthodes explique comment réaliser une tâche) et les actions sont représentées par les tâches et les outils (les tâches pour les actions de haut niveau, les outils pour les actions de bas niveau),

- il représente le raisonnement à différents niveaux d'abstraction : chaque méthode complexe décompose une tâche en une suite de sous-tâches du niveau d'abstraction inférieur, fournissant ainsi une explicitation du problème à plusieurs niveaux d'abstraction,
- il représente les connaissances procédurales et les connaissances sémantiques dans le même modèle : les tâches et les outils sont des objets exécutables mais ils peuvent également détenir des informations sémantiques telles que la description des objets recherchés ou des techniques utilisées,
- il permet l'accès aux résultats intermédiaires : chaque sous-tâche et chaque outil possède ses propres résultats auxquels il est possible d'accéder pour rechercher, par exemple, la cause d'échec d'un traitement.

L'expérimentation de ce système (détaillée au chapitre V) a montré que la formalisation de la démarche du traiteur d'images à l'aide d'un graphe TMO fournit un bon support de communication entre l'expert en TI et l'expert du domaine et permet une vision globale du processus. Cependant la lourdeur du processus d'acquisition interactive de connaissances et la multiplicité des applications indique la nécessité de réutilisabilité des traitements pour accroître l'aide fournie par le système. La réutilisation des applications, qui est notre deuxième objectif, est l'objet du chapitre suivant.

IV. Aide à la réutilisation des connaissances

IV.1. Motivations et objectifs

IV.1.1. Nos motivations

La modélisation sous forme de plans d'applications de TI dans divers domaines fait apparaître rapidement le besoin de pouvoir réutiliser des parties de plans construits auparavant. En effet, on peut se retrouver en face d'un problème qui a déjà été résolu dans un domaine d'application différent. Par exemple, la recherche d'un type particulier de cellules dans une image de cytologie (fig. 47-a et 47-d) correspond au problème « extraire des objets de taille relativement grande, de convexité assez élevée et d'un niveau de gris éloigné de celui du fond de l'image ». Ce problème peut avoir été rencontré dans le cas du tri de pièces industrielles en vrac sur un tapis roulant (fig. 47-b et 47-e). Dans un tel cas, on voudra réutiliser la stratégie mise en œuvre. On peut également souhaiter résoudre un nouveau problème sur un type d'images déjà traitées ou proche d'un type d'images déjà traitées. Par exemple, les images de cytologie et d'histologie présentées ci-dessous sont assez proches car toutes issues du milieu biomédical et acquises par le même type de capteur. Les problèmes posés sur ces images sont cependant très différents : extraction d'un type de cellules dans le premier cas et extraction de regroupements de noyaux dans le deuxième (fig. 47-c et 47-f). Toutefois, les deux premières phases du traitement (éliminer le bruit et éliminer le fond) sont identiques. Lors de la construction du deuxième plan, on cherchera à réutiliser une partie du premier plan construit.

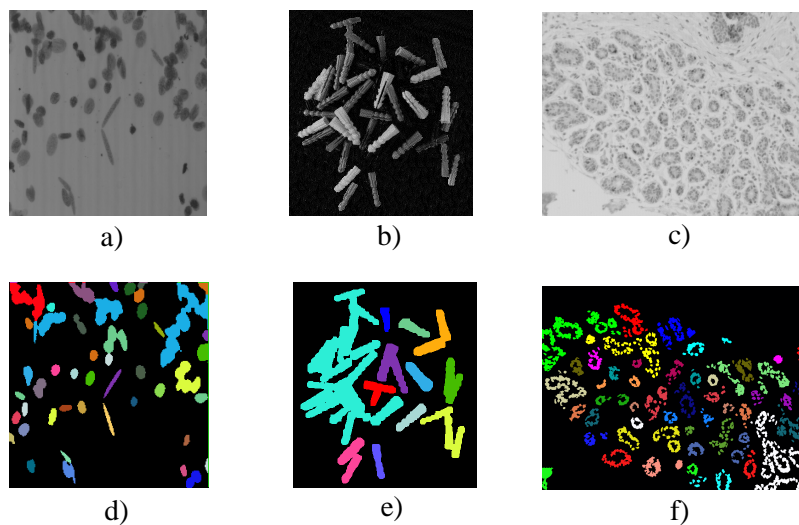


Fig.47 : images d'origine et images de résultat (ou résultat intermédiaire) pour les applications :

a, d : recherche de cellules dans une image de cytologie

b, e : tri de pièces industrielles

c, f : extraction de groupements de noyaux dans une image d'histologie

La réutilisation de plans ou de parties de plans TMO pour construire une application a un double intérêt pour le traiteur d'images :

- le gain de temps : la mise au point d'une application est un travail long et complexe pour lequel la réutilisation permet de réduire certaines étapes (que ce soit pour le choix de la stratégie générale ou de la technique appliquée pour un point particulier).
- le partage des connaissances et des expériences : un utilisateur peut réutiliser une application créée par un autre utilisateur, ou utiliser les connaissances et l'expérience modélisées pour un domaine d'application dans un autre domaine.

Pour mettre en œuvre ce type de réutilisation, nous avons choisi d'employer des techniques de RàPC. Le RàPC résout un nouveau problème en retrouvant et en adaptant des solutions ou des éléments de solutions d'un problème précédemment résolu. Il utilise un raisonnement assez proche du raisonnement humain pour pouvoir coopérer avec l'utilisateur et permet de se focaliser sur plusieurs catégories d'informations concernant le plan, que ce soient des informations sur le problème posé, sur les images traitées ou sur la stratégie à adopter [Ficet 98a].

IV.1.2. Objectifs du module de RàPC

Les avantages apportés par le RàPC à notre système se rapprochent de ceux exposés par Kolodner [Kolodner 91] sur « l'aide à la décision basée sur les cas ». Elle rappelle que le RàPC est un raisonnement fréquemment utilisé par l'humain pour résoudre des problèmes mais que ce dernier peut rarement l'exploiter pleinement, et ce pour plusieurs raisons :

- il lui est impossible de conserver tous les cas en mémoire,
- parmi ceux qu'il connaît, il ne se souvient pas toujours du cas le mieux approprié au moment voulu,
- s'il est un novice du domaine, il lui est difficile de réaliser la phase d'adaptation.

Kolodner propose donc un système qui permette à l'humain (le novice comme l'expert) de mieux raisonner par analogie, en laissant au système la charge de la mémoire et de la sélection des meilleurs cas, et en laissant à l'utilisateur la charge de la décision finale.

Suivant le même principe, l'intégration d'une base de cas et du module de RàPC dans notre système (figure 48) se veut être une aide supplémentaire pour l'utilisateur lors la construction de ses applications.

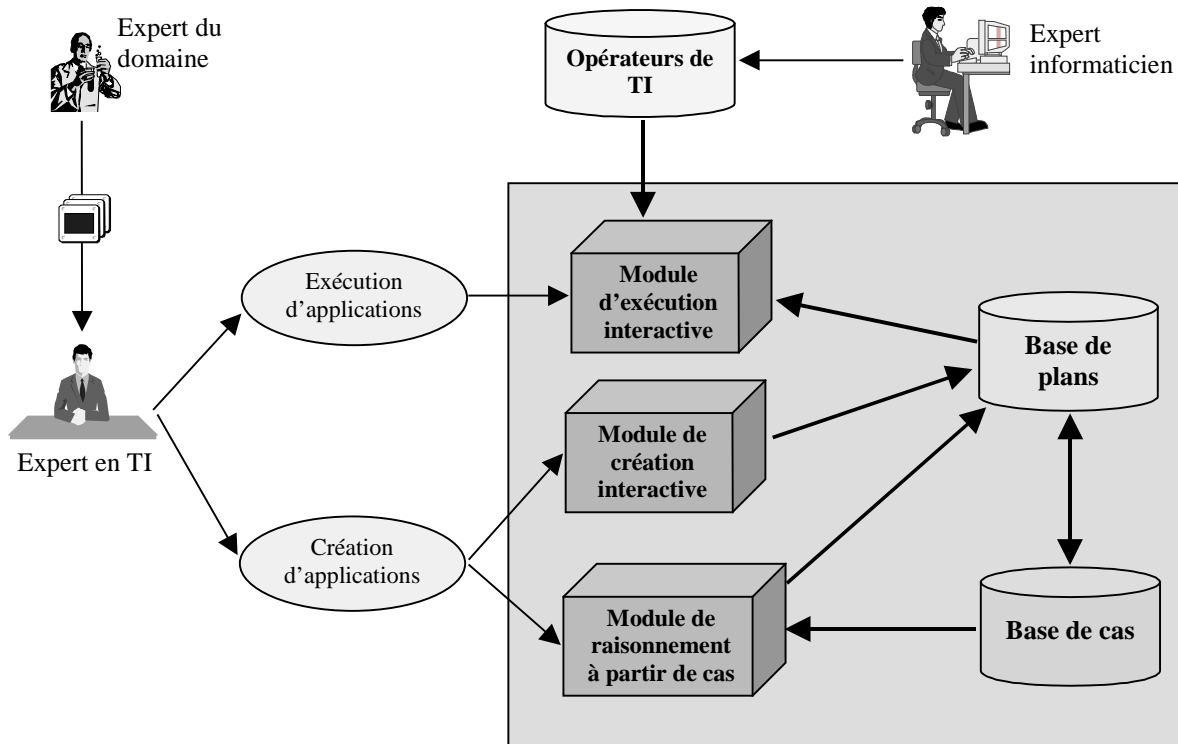


Fig.48 : système interactif muni d'un module de raisonnement à partir de cas

Le choix d'une représentation pour les éléments de la base de cas a été effectué à partir d'une étude des entités manipulées en TI. La représentation choisie pour les cas et les techniques de comparaison sont présentés dans la section IV.2. L'utilisateur de notre système peut maintenant créer ses applications soit à l'aide du module de création interactive, soit à l'aide du module de RàPC, soit en alternant l'utilisation des deux modules. Le fonctionnement du module de RàPC pour la création d'applications est décrit section IV.3. Enfin les techniques de mémorisation permettant la gestion de la base de cas sont spécifiées dans la section IV.4.

IV.2. Représentation des cas

Un cas est globalement composé de deux parties : la description de la solution et la description du problème.

Dans notre système une solution est modélisée sous forme d'un arbre TMO, qui peut être repéré par sa tâche racine. La base de cas peut donc être construite en associant la partie solution d'un cas à la tâche racine de chaque plan de la base de plans [Ficet 98b]. Cependant dans la section II.2, nous avons insisté sur l'intérêt de construire une solution par composition des sous-solutions de plusieurs

cas. Nous pouvons pour cela associer plusieurs cas à un même plan : le premier cas sera associé à la tâche racine du plan, les autres seront associés à des sous-tâches de ce même plan. Pour éviter un encombrement trop rapide de la base de cas, seules les sous-tâches représentatives d'une technique particulière devront être associées à un cas. Dans la figure 49, nous avons représenté un plan TMO comportant huit tâches et cinq outils (les méthodes n'ont pas été représentées dans le but de simplifier le schéma). Des cas seront associés aux tâches T1, T2 et T4 qui définissent une certaine stratégie. Par contre aucun cas ne sera associé aux tâches T3, T5, T6, T7 et T8 qui font directement appel à un outil.

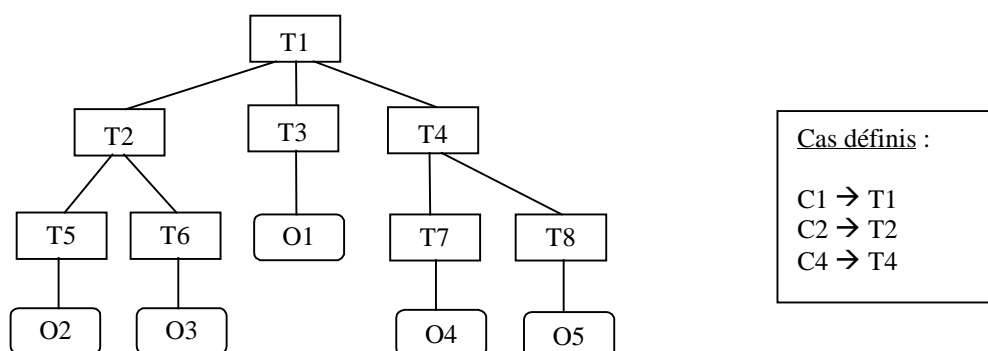


Fig.49 : exemple d'association d'un ensemble de cas à un plan TMO

La description du problème est généralement réalisée à l'aide d'un ensemble de critères discriminants. Nous avons vu dans la section II.2 que la détermination de ces critères demandait une étude approfondie du domaine, celle-ci est présentée dans la section IV.2.1. Puis dans la section IV.2.2, nous décrivons les fonctions de similarité qui effectuent la comparaison entre deux cas à l'aide de ces critères.

IV.2.1. Choix des critères de sélection

Outre les systèmes présentés dans la section I.3, notre étude pour la détermination des critères de similarité s'appuie sur des ouvrages et des thèses dédiés aux techniques du TI [Marion 87] [Elmoataz 90] [Chassery 91] [Kunt 93] [Cocquerez 95]. Le but est de déterminer des critères permettant de caractériser un problème de TI en fonction des actions à effectuer et des données à traiter.

Le premier problème concerne le choix d'un vocabulaire commun aux différents traiteurs d'images. A l'exception des actions de bas niveau (correspondant aux opérateurs de notre bibliothèque), il n'existe pas réellement de consensus sur les termes utilisés en TI. Ceci est dû, en

particulier, aux difficultés de s'abstraire du domaine de l'image (beaucoup de traiteurs d'images travaillent sur un seul domaine d'application et utilisent donc les termes de ce domaine).

La plupart des critères que nous proposons sont issus d'une classification des termes retenus pour décrire les actions et les données du TI. Des listes complètes de ces termes sont fournies au chapitre V.

Nous avons distingué deux grandes catégories de critères : ceux liés à la description de la tâche et ceux liés à la description de l'image et de son contexte.

Critères liés à la description de la tâche

Ces critères sont des informations se rapportant à l'opération réalisée par la tâche et à sa position dans le plan par rapport aux autres tâches. Ils comprennent le type ou la phase de traitement, la définition même du problème et le niveau d'abstraction correspondant.

Le **type ou la phase de traitement** correspond globalement au type de problème résolu par la tâche. Suivant le niveau d'abstraction de la tâche concernée, on prendra en compte :

- soit le type de traitement : la tâche racine d'un plan complet définit un traitement de haut niveau qui appartient à un type de traitement [Marion 87] tel que l'*amélioration* ou la *restauration* (correction des imperfections dues à l'appareillage et aux conditions d'acquisition), la *détection* (recherche de la présence d'un indice visuel, d'un contour ou d'une texture prédéfini), la *segmentation* (partitionnement de l'image en contours ou en régions) ou l'*extraction d'attributs* (mesures des caractéristiques sur les objets présents dans l'image),
- soit la phase de traitement : chaque sous-tâche d'un plan définit une partie du traitement qui correspond à une phase particulière de ce traitement. Par exemple, le *pré-traitement*, la *détermination de germes*, la *localisation de contours*, le *regroupement/décomposition* sont des phases possibles pour un traitement de type segmentation.

Les différentes phases de traitement représentent un découpage vertical du plan (fig. 50) pour un type de problème, certaines phases pouvant être optionnelles (par ex. : la phase de prétraitement dans un problème de segmentation).

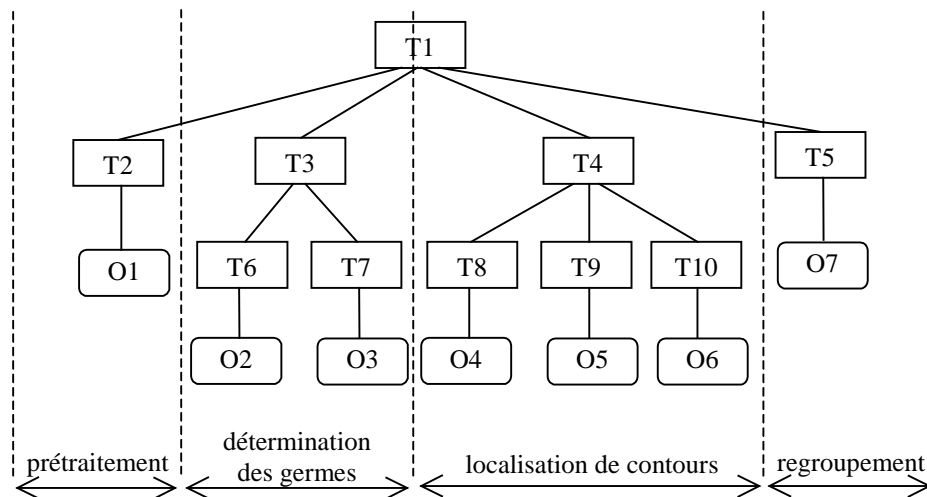


Fig.50 : exemple de découpage vertical d'un plan pour un problème de segmentation

La **définition du problème** est composée d'un ensemble de mots clefs sélectionnés parmi trois listes prédéfinies :

- une liste de verbes : ce sont les opérations effectuées par la tâche. Suivant le niveau d'abstraction de la tâche, elles définissent des actions de haut niveau telles que *détecer* ou *classifier*, ou des actions de bas niveau telles *binariser* ou *lisser*,
- une liste de noms : ce sont soit les objets sur lesquels la tâche effectue une opération tels que les *contours*, les *régions*, le *fond* de l'image ou le *bruit*, soit des noms définissant la technique à appliquer tels que *variance* (« binariser suivant la variance ») ou *croissance* (« former les régions par croissance »),
- une liste d'adjectifs : ce sont les qualificatifs soit des objets sur lesquels l'action est réalisée tels que *petit* (« supprimer les petits arcs ») ou *local* (« détecter les minima locaux »), soit de l'action elle-même tels que *partiel* (« séparer partiellement les régions ») ou *fort* (« faire un lissage fort »).

Comme on peut le voir sur les exemples proposés, le vocabulaire appartenant à ces trois listes de mots clefs est indépendant de tout domaine d'application.

Enfin les **niveaux d'abstraction**, dont l'ensemble correspond à un découpage horizontal du plan (fig. 51), sont basés sur ceux définis dans le planificateur automatique BORG [Clouard 94], ils sont également très proches des trois niveaux d'abstraction que distingue David Marr [Marr 82] dans son étude sur la perception visuelle.

- Le premier niveau d'abstraction est le *niveau intentionnel*. Les tâches de ce niveau répondent à la question « quoi faire ? » et portent sur les objectifs du TI. « isoler les objets du fond », « éliminer le fond de l'image » et « extraire les groupes d'objets » sont des tâches de ce niveau.

- Le deuxième niveau est le *niveau fonctionnel*. Les tâches de ce niveau répondent à la question « comment faire ? » et représentent une technique du TI dans laquelle on fait abstraction des contraintes techniques liées à la réalisation. Elles décrivent des méthodes de TI pour atteindre un but donné. « classifier les pixels », « éliminer les régions dues au bruit » et « former les groupes d'objets par la distance » sont des tâches de ce niveau.
- Le troisième et dernier niveau est le *niveau opérationnel*. Les tâches de ce niveau répondent à la question « avec quoi ? » et représentent un savoir-faire technique sur le TI qui peut s'implanter sous forme d'algorithme. Ce niveau se rapproche de la notion d'opérateur mais sans être lié à une bibliothèque particulière. « binariser », « étiqueter les régions » et « former les groupes d'objets en fonction de leur distance » sont des tâches de ce niveau.

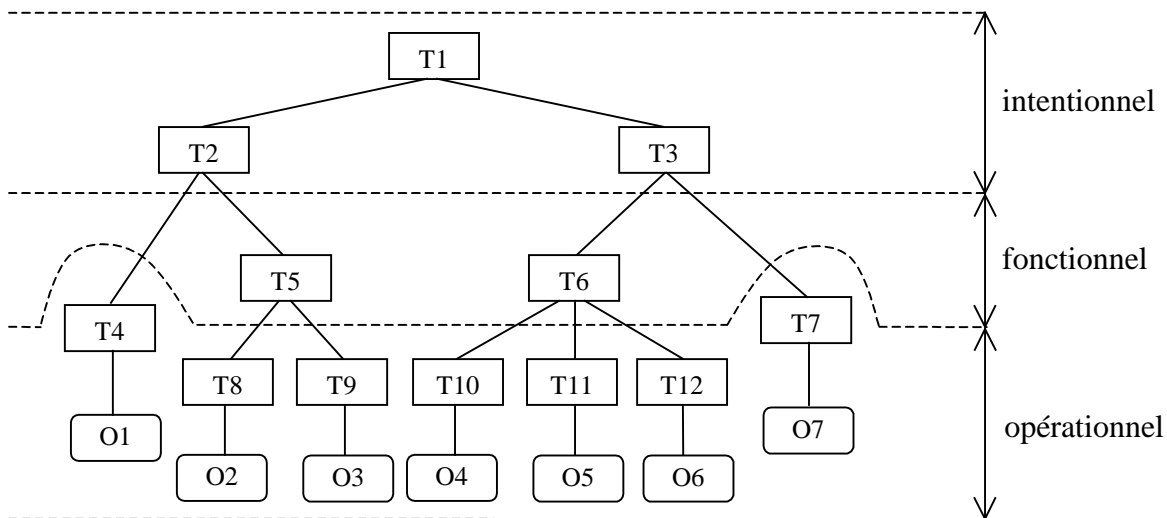


Fig.51 : exemple de découpage horizontal d'un plan

Critères liés au contexte des images

Parmi les connaissances se rapportant à la description de l'image et de son contexte, Elmoataz [Elmoataz 90] distingue trois grands types :

- les connaissances physiques : ce sont les connaissances concernant la physique de la formation de l'image, elles décrivent en autres les conditions d'acquisition de l'image, la résolution et la nature des capteurs,
- les connaissances perceptuelles : elles correspondent à la description symbolique de l'image en termes d'indices visuels ou de structures définies par des relations entre ces indices,
- les connaissances sémantiques : elles se rapportent à la « scène » analysée et aux éléments qui la composent et sont généralement associées à des modèles d'objets.

Toutes ces connaissances ne peuvent pas directement constituer des critères discriminants pour notre système. En effet, certaines fournissent trop peu d'informations exploitables (par exemple, les connaissances sémantiques sont souvent exprimées dans les termes du domaine de l'image), d'autres sont redondantes (par exemple la résolution et la taille relative des objets). Cependant, une bonne partie de ces connaissances peut être modélisée sous forme d'un ensemble de critères (symboliques ou numériques) qui sont discriminants pour le choix des tâches à réaliser et des stratégies à appliquer.

Parmi les critères que nous avons définis, certains découlent des connaissances physiques et décrivent la qualité de l'image. Il s'agit en particulier du **type** et de la **quantité de bruit** que l'on trouve dans l'image et de la **qualité du contraste**. Ce sont des critères particulièrement importants pour le choix d'un prétraitement.

D'autres critères correspondent plutôt à la description perceptuelle de l'image. Ils comprennent la **présence** ou l'**absence d'un fond** et l'**aspect des objets** présents ou recherchés (*niveau de gris homogène, couleur claire, textures, contours épais, ...*).

Enfin, le dernier groupe de critères est issu des connaissances sémantiques abstraites du domaine de l'image et portent sur ce que l'on cherche à mettre en évidence. Il s'agit de la **forme** des objets recherchés (*convexe, concave, allongé, compact, carré, rond, ...*), de la **taille relative** des objets par rapport à l'ensemble des objets présents dans l'image, la **position** des objets recherchés (*droite, milieu, gauche, bas, centre, haut*), et les **relations** entre les objets recherchés (*proches, connexes, inclus, ...*).

IV.2.2. Calcul de la similarité entre deux cas

On a vu à la section II.3 que l'on pouvait distinguer deux principes pour le calcul de cas proches : le premier cherche à maximiser la similarité et le second à minimiser l'effort d'adaptation. L'absence de méthode automatique d'évaluation des résultats en TI nous conduit à choisir la première approche. Nous décrivons les fonctions utilisées pour le calcul de la similarité dans le premier paragraphe de cette section. Puis dans le deuxième paragraphe, nous détaillons les modes de comparaison définis pour les différents types de critères. Par ailleurs, les critères énoncés dans la section précédente n'entrent pas tous en compte pour une application donnée. La gestion de l'absence de valeur pour un critère est expliquée dans le troisième paragraphe.

Les fonctions de similarité

Parmi les critères définis dans la section précédente, le premier groupe (critères liés à la définition de la tâche) caractérise l'action effectuée et est dépendant de notre modèle de représentation. Quelque soit la tâche concernée, une valeur doit être donnée à chacun de ces critères :

- une tâche n'a de sens que si le problème qu'elle résout est défini,
- la définition du problème et la position de la tâche dans l'arbre par rapport à un axe temporel horizontal permet de déduire le type ou la phase du traitement (découpage vertical),
- la définition du problème et la position de la tâche dans l'arbre par rapport à un axe vertical définit son niveau d'abstraction (découpage horizontal).

Ces critères définissent un ensemble de tâches résolvant un « type de problème ». Ce sont des critères « obligatoires », c'est à dire qu'un cas cible possède obligatoirement une valeur pour chacun d'eux. La recherche des cas possédant un ensemble de valeurs proches de celles du cas cible pour cet ensemble de critères permet de réduire considérablement l'espace de recherche. Une première fonction de similarité Φ_t utilisant ces trois critères va donc servir à restreindre l'ensemble de cas sources candidats. Elle est définie par la formule (1) comme la moyenne pondérée des calculs de similarité sur chacun des critères : S est le cas source, C est le cas cible, α_{Cr} est le coefficient d'importance du critère Cr et $\phi_{Cr}(S,C)$ est la similarité entre S et C suivant le critère Cr. La valeur retournée par une fonction ϕ_{Cr} varie de 0 si les valeurs de Cr sont très différentes entre les deux cas à 1 si elles sont identiques. Les coefficients α_{Cr} sont également compris entre 0 et 1, ce qui permet de normaliser la fonction Φ_t entre 0 et 1.

$$\Phi_t(S, C) = \frac{\sum \alpha_{Cr} \times \phi_{Cr}(S, C)}{\sum \alpha_{Cr}} / Cr \in \{ \text{critères liés à la tâche} \} \quad (1)$$

Le deuxième groupe de critères (critères liés au contexte des images) caractérise les objets manipulés et est dépendant de l'images traitée. Ces critères n'ont pas de sens pour toutes les images : par exemple la qualité du contraste n'a pas de sens lorsque l'on traite une carte de régions et la position des objets n'a pas de sens lorsque l'on cherche à extraire tous les objets présents. Ce sont des critères « optionnels », c'est-à-dire qu'un cas cible n'aura pas forcément de valeur pour tous ces critères. La recherche de cas sources possédant des valeurs proches de celles du cas cible pour ces critères s'effectue sur l'ensemble de cas retenus à l'aide de la fonction Φ_t . Une seconde

fonction de similarité Φ_i permet donc de réduire cet ensemble de cas pour en faire une liste présentable à l'utilisateur. Cette fonction est définie par la formule (2) comme la moyenne pondérée des calculs de similarité sur chacun des critères : les notations utilisées ont la même signification que pour la formule (1). Les coefficients α_{Cr} , les fonctions ϕ_{Cr} ainsi que la fonction Φ_i possèdent les mêmes propriétés que dans la fonction Φ , c'est-à-dire qu'ils sont normalisés entre 0 et 1.

$$\Phi_i(S, C) = \frac{\sum \alpha_{Cr} \times \phi_{Cr}(S, C)}{\sum \alpha_{Cr}} / Cr \in \{ \text{critères liés au contexte des images} \} \quad (2)$$

Les définitions des fonctions ϕ_{Cr} calculant la similarité suivant les différents critères sont données dans le paragraphe suivant. L'utilisation des fonctions de similarités Φ_t et Φ_i par l'algorithme de sélection/adaptation ainsi que le réglage des coefficients d'importance font l'objet de la section IV.3.

Les différents modes de comparaison sur un critère

Les critères proposés sont de type et de nature différents et ont chacun un domaine de valeurs particulier : certains sont monovalués, d'autres multivalués ; certains ont des valeurs numériques, d'autres des valeurs symboliques. Il est clair que la liste des critères relatifs au contexte des images proposée ne peut pas être exhaustive : les critères que nous proposons ont émergé de notre étude de la littérature du TI et de la mise en œuvre de nos applications. Un grand nombre de traitements et de domaines devront être testés pour la compléter. Nous avons donc défini des types de critères associés chacun à un mode de comparaison de façon à pouvoir intégrer facilement un nouveau critère. La comparaison entre la valeur d'un cas cible et celle d'un cas source suivant un type de critère est définie par une fonction générique. Les types de critères définis sont :

- critère numérique strict : la valeur est un nombre entier ou réel et la comparaison entre deux valeurs est 1 si elles sont égales, 0 sinon, (pas d'exemple actuellement)
- critère symbolique strict : la valeur est un symbole et la comparaison entre deux valeurs est 1 si elles sont égales, 0 sinon, (ex. : présence d'un fond \rightarrow oui/non)
- critère numérique gradué : la valeur appartient à un intervalle d'entiers ou de réels et la comparaison entre deux valeurs est l'écart des deux valeurs rapporté à la plage de valeurs définie par l'intervalle, (ex. : taille relative des objets $\rightarrow [1,5]$),

- critère symbolique gradué : la valeur appartient à un ensemble ordonné de symboles et la comparaison entre deux valeurs est le rapport entre l'écart entre les deux valeurs suivant l'ordre défini rapporté à la largeur de la plage de valeur définie par la liste, (ex. quantité de bruit → *{très faible, faible, moyen, fort, très fort}*),
- critère multivalué : la valeur est une liste non ordonnée de symboles et/ou de nombres et la comparaison entre deux valeurs est le rapport entre le nombre d'éléments communs aux deux valeurs et le nombre d'éléments de la première, (ex. : verbes de définition du problème → *{amincir, binariser, calculer, classifier, ...}*).

Gestion de l'absence de valeur pour un critère

La gestion des valeurs manquantes est un problème important en apprentissage. L'absence d'une valeur pour un exemple peut avoir plusieurs causes :

- la valeur n'a pas de sens pour l'exemple (contraste sur une carte de régions),
- la valeur est apparue inutile pour l'exemple,
- la raison est indéterminée (erreur, panne, ...).

Dans le domaine de l'apprentissage symbolique, Thomas propose quatre façons de gérer une valeur manquante pour un attribut [Thomas 96] :

- la considérer comme non pertinente pour l'exemple (ne pas en tenir compte),
- la considérer comme une valeur particulière et l'utiliser en tant que tel,
- considérer l'absence comme une source d'information (contre exemple),
- déterminer la probabilité que l'attribut ait une valeur v en examinant les valeurs des autres exemples de la classe.

Notre point de vue sur l'absence d'une valeur est différent suivant qu'il s'agit du cas source ou du cas cible :

- l'absence d'une valeur pour un cas cible signifie que la valeur est non pertinente pour le cas (soit elle n'avait pas de sens, soit l'utilisateur l'a jugée inutile pour sa recherche), cette absence n'aura donc aucun impact sur le calcul de la similarité, ce qui correspondra à donner un calcul de similarité sur le critère et un coefficient d'importance nuls,

- l'absence d'une valeur pour un cas source (si cette valeur est présente pour le cas cible) signifie qu'une des conditions de similarité n'est pas respectée, cette absence doit donc faire baisser le résultat du calcul de la similarité, ce qui correspondra à calcul de similarité sur le critère nul et un coefficient d'importance non nul.

Ces deux conditions sont respectées par l'ensemble des fonctions génériques calculant la similarité pour un type de critère.

IV.3. L'algorithme de sélection/adaptation

IV.3.1. Principe général de l'algorithme

Lors de la description des différents processus de sélection/adaptation du chapitre II, nous avons insisté sur deux points particuliers :

- l'intérêt d'une étape préliminaire dans le processus de sélection visant à réduire l'espace de recherche,
- l'adéquation d'un cycle sélection/adaptation applicable itérativement pour les problèmes de planification.

Nous proposons donc un processus (fig. 52) qui met en œuvre un cycle sélection/adaptation applicable itérativement à différents niveaux du plan solution et dont chaque exécution du cycle comporte une phase de réduction de l'espace de recherche.

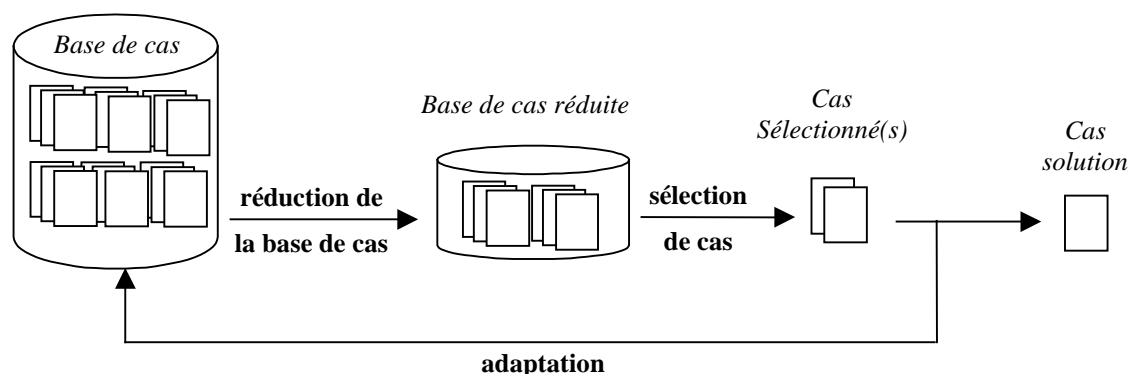


Fig.52 : schématisation du processus de sélection/adaptation mis en œuvre

On a vu que l'on pouvait réaliser la phase de réduction de l'espace de recherche, soit en utilisant des critères définissant des contraintes strictes, soit en considérant que deux cas ne peuvent être comparés que s'ils sont définis à l'aide du même ensemble de critères. La deuxième technique n'est pas adaptée à notre domaine. En effet, parmi les critères liés au contexte des images, certains

peuvent être non informants pour le cas cible sans être disqualifiants pour le cas source. La phase de réduction de la base de cas est donc réalisée à l'aide de la fonction Φ_t utilisant les critères « obligatoires » liés à la description de la tâche et la phase de sélection de cas à l'aide de la fonction Φ_i utilisant les critères « optionnels » liés au contexte de l'image.

L'objectif de notre module de RàPC est de fournir une aide au traiteur d'images lors de la construction d'applications en l'aidant à sélectionner les solutions des problèmes déjà résolus proches de son problème courant. La mise en œuvre du processus de sélection/adaptation est donc réalisée en coopération avec l'utilisateur selon l'algorithme suivant :

- 1 - Demander à l'utilisateur les valeurs des critères liés à la description du problème
- 2 - Rechercher l'ensemble • des cas correspondant aux valeurs désirées en utilisant Φ_t
- 3 - Demander à l'utilisateur les valeurs des critères liés au contexte des images
- 4 - Tant que l'ensemble • n'est pas de taille convenable
 - Régler les poids des critères
 - Réduire l'ensemble • en utilisant la fonction Φ_i
- 5 - Demander à l'utilisateur de choisir un cas source parmi l'ensemble •
- 6 - Présenter le plan solution à l'utilisateur et lui proposer de modifier les sous tâches qui ne lui conviennent pas, soit à l'aide du module de RàPC, soit à l'aide du module de création interactive

Les étapes 1 et 3 correspondent à la saisie de la description du cas cible. L'étape 2 réalise la phase de réduction de l'espace de recherche. La sélection de cas sources candidats est mise en œuvre par l'étape 4 et est finalisée par le choix de l'utilisateur dans l'étape 5. Enfin l'étape 6 a pour but l'adaptation du cas source sélectionné au problème courant.

Les principes de sélection et d'adaptation utilisés dans cet algorithme sont détaillés dans les deux paragraphes suivants (IV.3.2 et IV.3.3).

IV.3.2. Sélection d'un cas source

Au cours de l'étape 2 de l'algorithme, la réduction de l'espace de recherche consiste à sélectionner les cas sources qui traitent du même type de problème que le cas cible C . Cela correspond à sélectionner les cas S tels que $\Phi_t(S, C) > \alpha_t$ où α_t est un seuil fixé à l'avance (la fonction Φ_t retournant une valeur entre 0 et 1, α_t sera fixé à 0,5 par défaut). Les valeurs des poids de chaque critère pour la fonction Φ_t sont également fixes : la même importance est accordée à chacun des trois critères (1 / 1 / 1), pour la définition du problème nous accordons une importance un peu

plus grande aux verbes qu'aux noms et adjectifs (0,4 / 0,3 / 0,3) car l'action réalisée par la tâche est d'abord décrite par le verbe. Cette étape fournit un premier ensemble de cas Σ .

Pour que l'utilisateur puisse faire son choix à l'étape 5, il faut que l'ensemble de cas résultant de l'étape 4 soit de taille raisonnable. S'il possède trop peu de cas, l'avis de l'utilisateur perd de son importance, et s'il en possède trop, il sera difficile à l'utilisateur de faire son choix. Le caractère itératif de l'étape 4 permet de déterminer un ensemble de cas présentable à l'utilisateur sous forme de liste : ce dernier peut alors examiner chaque cas en détail avant de faire son choix. La modification de l'ensemble Σ à chaque itération est effectuée par relaxation en modifiant les poids des critères et/ou le seuil de sélection. Pour cela, lorsque l'utilisateur saisit les valeurs des critères de son cas cible, il indique pour chaque critère s'il le considère comme important ou pas. Tous les coefficients des critères sont initialisés à 0,5. A chaque itération, le système sélectionne les cas S de l'ensemble Σ tels que $\Phi_i(S, C) > \alpha_i$ où α_i est le seuil de sélection. Si l'ensemble résultant n'est pas de taille convenable (par défaut entre 2 et 5 cas), on augmente les coefficients des critères les plus importants de 0,1 et on baisse ceux des critères les moins importants de 0,1 pour l'itération suivante. Lorsque l'on ne peut plus modifier les coefficients (coefficients des critères les moins importants à 0), si l'ensemble de cas source n'est toujours pas de taille convenable, un deuxième mode de relaxation consistant à diminuer le seuil α_i est utilisé.

La sélection du cas source le plus intéressant peut alors être faite par l'utilisateur. La réalisation de cette étape de façon interactive permet de laisser suffisamment d'importance à l'aspect intuitif qui caractérise le mode de travail d'un expert en TI.

IV.3.3. Adaptation interactive d'un plan

On a vu au chapitre II.4 que l'adaptation d'un cas à l'aide de parties d'autres cas était particulièrement intéressante dans le domaine de la planification. Dans notre système, un cas peut être adapté à plusieurs niveaux et de plusieurs façons : localement ou globalement, à l'aide du module de RàPC ou du module de création interactive.

Le plan solution d'un cas peut demander uniquement des modifications locales. Il s'agit par exemple de régler les paramètres d'un opérateur ou de remplacer un opérateur par un autre mieux adapté. Ce type de modification peut être fait à l'aide de la fonctionnalité « modifier élément » du module de création interactive.

Mais ce plan peut également demander des modifications plus globales, c'est à dire nécessiter le remplacement d'un sous-plan par un autre. Pour cela, l'étape 5 de l'algorithme propose à l'utilisateur d'adapter la solution de son cas en remplaçant la tâche racine d'un sous-plan par une autre tâche. La tâche remplaçante peut être obtenue, soit en relançant l'algorithme pour trouver un cas correspondant, soit par construction via le module de création interactive.

Dans l'exemple de la figure 53, le plan solution du cas source sélectionné est adapté en trois étapes successives. Lors de la première étape d'adaptation, l'utilisateur demande au système de relancer l'algorithme pour remplacer le sous-plan A : un nouveau cas cible correspondant au sous-problème est alors spécifié, ce cas cible permet de sélectionner le plan A' et de remplacer A par celui-ci. Lors de la deuxième étape, il remplace le sous-plan B par le sous-plan B' qui est construit à l'aide du module de création interactive (le sous-plan B' comporte une tâche et un outil, il est donc plus rapide de le construire manuellement). Enfin, pour la troisième phase d'adaptation, l'utilisateur va transformer l'outil C en un outil C' en changeant l'opérateur relié à l'outil.

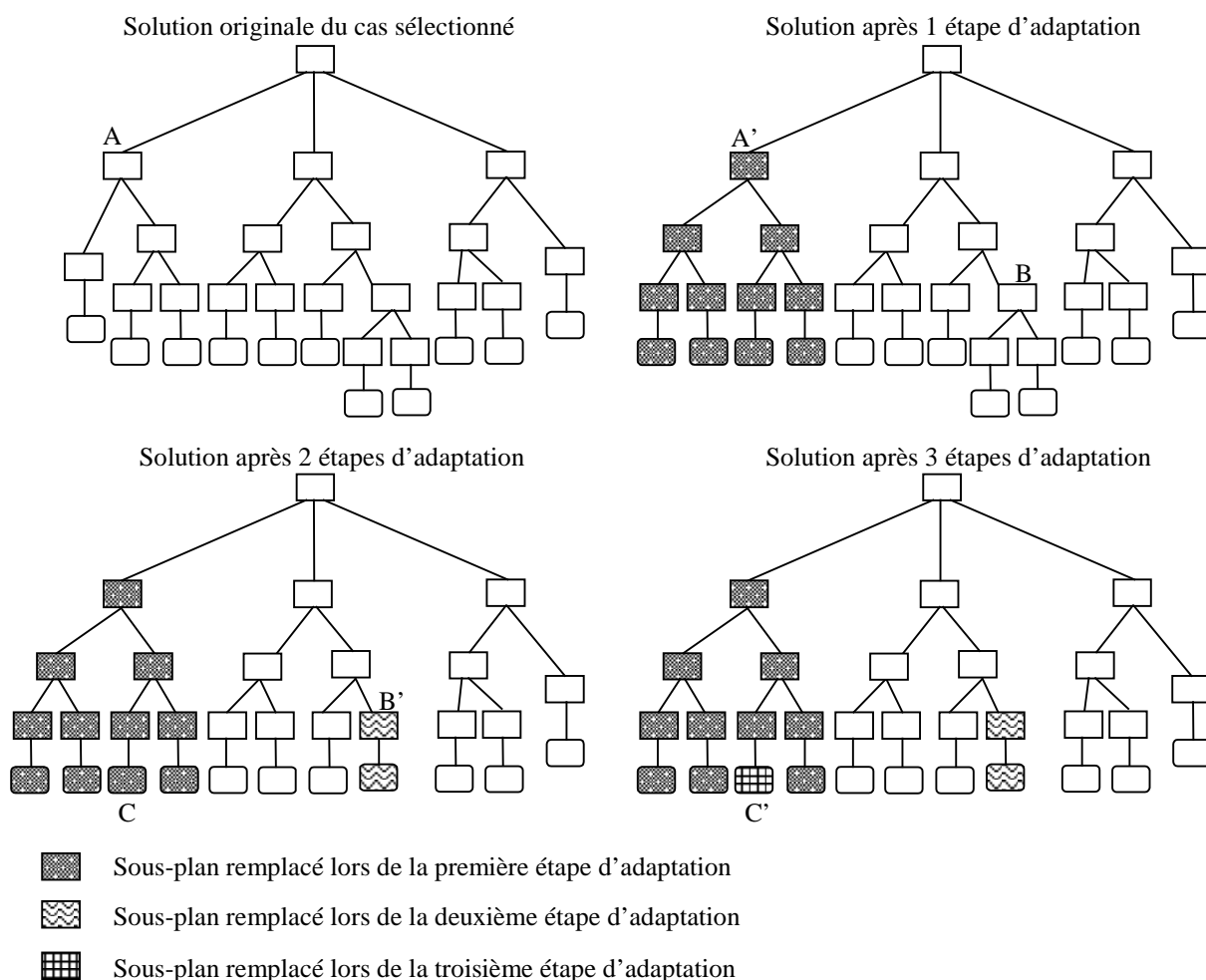


Fig.53 : exemple d'adaptation d'un plan solution en trois étapes

Cet exemple montre l'intérêt de l'aspect récursif de l'algorithme : un plan solution peut être adapté à n'importe quel niveau dans l'arbre de tâches (A est une tâche de haut niveau, B une tâche de bas niveau et C est un opérateur) et autant que nécessaire (on remplace A par A', puis A' est adapté en remplaçant C par C'). Une fois la nouvelle solution mise au point, se pose le problème de savoir si cette solution doit donner lieu à l'ajout de nouveaux cas dans la base. Ce problème est discuté dans la section suivante.

IV.4. La mémorisation d'un nouveau cas

On a vu au chapitre II.5 que la mémorisation d'un nouveau cas n'a d'intérêt que s'il apporte de nouvelles connaissances à la base. Cela implique qu'un cas doit respecter deux conditions pour être intégré : les connaissances qu'il comprend doivent être correctes et elles doivent être suffisamment différentes de celles des cas déjà présents dans la base.

Vérifier le respect de la première condition consiste à tester la cohérence et l'efficacité du plan solution. Un plan est cohérent s'il s'exécute normalement et il est efficace s'il fournit des résultats satisfaisants. La cohérence peut être vérifiée par le bon déroulement d'une exécution mais l'efficacité ne peut être approuvée que par l'utilisateur, puisqu'il est seul juge de la justesse des résultats. L'intégration d'un nouveau cas sera donc réalisée à la demande de l'utilisateur après que celui-ci ait validé sa solution par un ensemble de tests.

Un plan solution complet peut donner lieu à l'intégration de plusieurs cas dans la base : en effet si un plan complet représente la solution d'un problème de haut niveau, les différents sous-plans inclus représentent les solutions de problèmes de niveaux inférieurs. Lorsque l'intégration d'un cas est demandée, la première étape consiste donc à déterminer la liste des plans et sous-plans constituant les solutions des cas candidats à une intégration. Cette liste correspond à l'ensemble des plans ayant subi une adaptation, c'est à dire aux ascendants des sous-plans remplacés qui ne figurent pas encore dans cette liste et qui sont de taille suffisamment importante (au minimum sur trois niveaux de tâches). Si le plan remplaçant a été construit à l'aide du module de création interactive, on l'intègre également dans la liste. La figure 54 reprend le plan de l'exemple d'adaptation présenté figure 53, la détermination des plans candidats est réalisée en examinant les trois sous-plans remplacés :

- sous-plan de racine A' : D est inséré dans la liste, A' n'est pas inséré car il est issu d'un cas de la base,

- sous-plan de racine B' : les plans de racines E et F sont insérés dans la liste, B' a été construit manuellement mais il n'est pas inséré car il ne possède que deux niveaux,
- sous-plan de racine C' : les plans de racine A' et G sont insérés dans la liste, H et C' ne sont pas insérés car ils possèdent moins de trois niveaux.

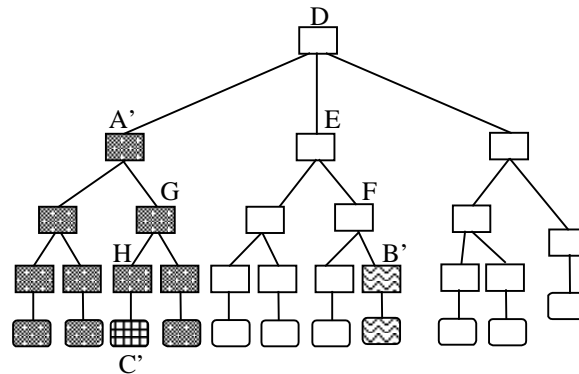


Fig.54 : exemple de détermination des cas candidats à la mémorisation

Puis pour chacun des plans de cette liste, l'utilisateur doit préciser les valeurs de critères du cas correspondant qui ont été modifiées. Le système recherche alors le cas de la base ayant la plus grande similarité avec le nouveau cas et intègre ce dernier si cette similarité est inférieure à un seuil donné (i.e. si le cas est suffisamment différent des autres cas de la base). La similarité calculée ici correspond au minimum entre la similarité suivant les critères liés à la tâche et la similarité suivant les critères liés au contexte des images.

L'intégration de nouveaux cas peut également être demandée pour un plan entièrement construit avec le module de création interactive. Dans ce cas tous les sous-plans de plus de trois niveaux de tâches seront des sous-plans candidats et les cas devront être entièrement définis pour chaque sous-plan. Ce type de mémorisation est particulièrement contraignante pour l'utilisateur qui devra saisir beaucoup de valeurs. Elle peut cependant être demandée par un utilisateur qui estime avoir modélisé des connaissances nouvelles et réutilisables.

IV.5. Conclusion

L'expérimentation du module de RàPC sur des applications de segmentation d'images a montré la pertinence de notre approche aussi bien au niveau du choix des critères de sélection que de l'algorithme de recherche/adaptation :

- les cas les plus pertinents sont proposés à l'utilisateur,
- l'adaptation récursive permet la conception d'un plan par assemblage de parties d'autres plans,

- la possibilité d'utiliser le module de construction interactive évite de rester sur un échec.

Cependant, pour pouvoir s'appliquer dans n'importe quelle situation et pour fournir une aide plus importante ce module pourra être complété sur plusieurs points :

- la liste des critères liés au contexte des images devra être perfectionnée pour s'adapter à différents types de traitement et à différents domaines d'image, ce travail est dépendant de la disponibilité et des applications en cours de nos traiteurs d'images et ne peut donc être fait que sur une longue période,
- certaines valeurs de critères peuvent être mesurées (qualité du contraste) ou déduite du domaine d'application (type de bruit) : il serait intéressant de fournir des moyens de valuer automatiquement ces critères pour alléger la tâche de l'utilisateur,
- une fois l'ensemble des critères figé, la base de cas devra être hiérarchisée pour accélérer la recherche d'un cas,
- la valuation automatique de certains critères et la hiérarchisation de la base de cas permettrait aussi d'améliorer la phase de mémorisation.

V. Implémentation et expérimentation

V.1. Implémentation du système interactif et de la base de cas

L'implémentation de l'architecture TMO et de l'algorithme de RàPC a été réalisée en CLOS (CommonLisp Object System). L'interface graphique permettant la coopération entre le système et l'utilisateur est réalisée en C++ et fait appel à la bibliothèque Motif. Nous décrivons tout d'abord les éléments de l'architecture (les principales classes, le contrôle), puis nous exposons les différentes fonctionnalités proposées par l'interface graphique.

V.1.1. Les éléments de l'architecture

Nous présentons ici la description des classes TACHES, METHODES, OUTILS, CAS, ainsi que les classes représentant les données manipulées. Les classes TACHES et OUTILS représentent des opérations à effectuer sur les données, et possèdent donc chacune une liste d'entrées, une liste de paramètres, une liste de sorties et une liste de résultats. Pour les tâches et les outils du niveau domaine, les entrées et les sorties ont des valeurs de type image et les paramètres et les résultats des valeurs de type numérique.

Les classes TACHES et METHODES possèdent un champ *niveau_requis* qui définit le niveau requis par l'utilisateur pour les modifier ou les exécuter. Ce champ peut prendre trois valeurs : « expert du système », « expert en TI » et « novice ». Cette différenciation entre différents types d'utilisateurs permet de ne donner accès à certaines tâches de contrôle et de métacontrôle qu'aux utilisateurs compétents : par exemple, seul l'expert du système peut créer ou modifier une tâche de contrôle et seuls l'expert du système et l'expert en TI peuvent créer de nouveaux cas.

Pour parcourir facilement l'arbre TMO, plusieurs liens sont établis entre les différents éléments : ainsi une tâche dispose de la liste de ses méthodes, une méthode « connaît » la tâche qu'elle résout et la décomposition en sous-tâches qu'elle implique ou l'outil auquel elle fait appel.

D'autre part, lors d'une exécution, pour récupérer les données fournies en sortie par les tâches et les outils, on a besoin de savoir quelles décompositions en sous-tâches et quels outils ont été choisis : pour cela, il est nécessaire d'instaurer des liens dynamiques père/fils entre une tâche et les sous-tâches qui correspondent à la méthode choisie ou entre une tâche et l'outil auquel elle fait appel. Pour éviter des recherches inutiles de données nous notons également si la tâche ou l'outil qui les calcule a déjà été exécuté ou non.

Dans la définition de nos classes, certains champs sont « fixes » et d'autres sont « variables ». On entend par « champ fixe », un champ dont la valeur est définie une fois pour toute à la création de l'objet. Ce sont les champs qui définissent et caractérisent l'objet. Les « champs variables » ne sont pas là pour définir l'objet, mais pour aider à son utilisation lors d'une exécution. Ils sont initialisés avant chaque exécution et leur valeur est modifiée en fonction de l'utilisation que l'on fait de l'objet au cours de l'exécution.

Dans la suite de ce paragraphe nous détaillons les différents champs des principales classes, et nous expliquons leur signification.

La classe TACHES

Les objets de la classe TACHES représentent tous les buts ou sous-buts que le système peut réaliser, quelque soit leur niveau.

CHAMP	DESCRIPTION	DOMAINE	TYPE
<i>titre</i>	Titre explicitant la requête de la tâche à l'utilisateur	Chaîne de caractères	fixe
<i>dénomination</i>	Titre court permettant le repérage dans une liste	Chaîne de caractères	fixe
<i>niveau</i>	Niveau de connaissance auquel la tâche appartient	[métacontrôle, contrôle, domaine]	fixe
<i>nb_entrées</i>	Arité d'entrée, pour la création des entrées et leur parcours	Entier	fixe
<i>nb_sorties</i>	Arité de sortie, pour la création des sorties et leur parcours	Entier	fixe
<i>nb_paramètres</i>	Arité de paramètre, pour la création des paramètres et leur parcours	Entier	fixe
<i>nb_résultats</i>	Arité de résultat, pour la création des résultats et leur parcours	Entier	fixe
<i>entrées</i>	Liste des entrées de la tâche	Liste d'instances des différentes classes de données	variable
<i>sorties</i>	Liste des sorties de la tâche	Liste d'instances des différentes classes de données	variable
<i>paramètres</i>	Liste des paramètres de la tâche	Liste d'instances des classes de données numériques	variable
<i>résultats</i>	Liste des résultats de la tâche	Liste d'instances des classes de données numériques	variable
<i>niveau_requis</i>	Niveau minimum de l'utilisateur pour modifier ou exécuter cette tâche	Entier entre 1 et 3	fixe
<i>méthodes</i>	Liste des méthodes associées à cette tâche	Liste d'instances de la classe METHODES	fixe

<i>père</i>	Tâche représentée par le nœud père dans le graphe de tâches	Instance de la classe TACHES	variable
<i>fil</i>	Tâche/s ou outil représenté/e par le/s nœuds fils dans le graphe de tâches	Ensemble d'instance de la classe TACHES ou instance de la classe OUTILS	variable
<i>successeur</i>	Tâche(s) successeur(s) direct(s) dans le graphe de tâches	Liste d'instances de la classe TACHES	fixe
<i>prédécesseur</i>	Tâche(s) prédécesseur(s) direct(s) dans le graphe de tâches	Liste d'instances de la classe TACHES	fixe
<i>statut</i>	Etat de la tâche lors de l'exécution du graphe de tâches	[<i>exécuté, nil</i>]	variable

Les champs *entrées*, *sorties*, *paramètres*, *résultats*, *père*, *fil* et *statut* sont des champs « variables » qui servent lors de l'exécution de la tâche. En particulier, les champs *père* et *fil* permettent de repérer le chemin qui a été choisi dans l'arbre de tâches et le champ *statut* sert à vérifier si la tâche a été exécutée pour savoir si l'on peut se servir de ses résultats et de ses sorties. Les liens *père/fils* sont gérés automatiquement lors de l'exécution d'une méthode.

La classe METHODES

Chaque instance de la classe METHODES est associée à une instance de la classe TACHES et représente une des techniques grâce à laquelle cette tâche peut être effectuée.

CHAMP	DESCRIPTION	DOMAINE	TYPE
<i>titre</i>	Titre explicitant la stratégie de la méthode à l'utilisateur	Chaîne de caractères	fixe
<i>dénomination</i>	Titre cours permettant le repérage dans une liste	Chaîne de caractères	fixe
<i>ma_tâche</i>	Tâche à laquelle la méthode est rattachée	Instance de la classe TACHES	fixe
<i>niveau_requis</i>	Niveau minimum de l'utilisateur pour modifier ou exécuter cette tâche	Entier entre 1 et 3	fixe
<i>descendant</i>	Type de la méthode (complexe ou terminale)	[<i>décomposition, outil</i>]	fixe
<i>corps</i>	Enchaînements de tâches à réaliser ou nom de l'outil appelé	Liste d'instances de TACHES sous la forme (puis t1 t2 t3) ou instance d'OUTILS	fixe

La classe OUTILS

Les instances de la classe OUTILS font référence à un code informatique et possèdent tous les renseignements nécessaire à son exécution. Le code informatique peut être une fonction Lisp ou C, mais pour les outils du domaine, il s'agit généralement d'un opérateur de la bibliothèque Pandore.

Ces outils peuvent s'exécuter selon différents modes, chaque mode correspondant à un type de boucle classique de programmation :

- mode *normal* : exécution simple et unique de la fonction,
- mode *optimise* : on exécute la fonction avec différents jeux de paramètres et on retourne, soit le premier résultat qui répond à des critères pré-établis (mode d'optimisation *premier*), soit le résultat qui répond le mieux à des critères pré-établis (mode d'optimisation *meilleur*),
- mode *pour* : on exécute la fonction un nombre prédéfini de fois et on retourne le dernier résultat,
- mode *tant_que* : on exécute la fonction tant que son résultat répond à certains critères et on retourne le résultat de l'avant-dernière exécution.
- mode *jusqu_a* : on exécute la fonction tant que son résultat ne répond pas à certains critères et on retourne le résultat de la dernière exécution.

Pour les modes *optimise*, *tant_que* et *jusqu_a*, l'adéquation du résultat aux critères demandés est testée par une fonction d'évaluation.

CHAMP	DESCRIPTION	DOMAINE	TYPE
<i>titre</i>	Titre explicitant le but de l'outil à l'utilisateur	Chaîne de caractères	fixe
<i>dénomination</i>	Titre cours permettant le repérage dans une liste	Chaîne de caractères	fixe
<i>nb_entrées</i>	Arité d'entrée, pour la création des entrées et leur parcours	Entier	fixe
<i>nb_sorties</i>	Arité de sortie, pour la création des sorties et leur parcours	Entier	fixe
<i>nb_paramètres</i>	Arité de paramètre, pour la création des paramètres et leur parcours	Entier	fixe
<i>entrées</i>	Liste des entrées de l'outil	Liste d'instances des différentes classes de données	variable
<i>sorties</i>	Liste des sorties de l'outil	Liste d'instances des différentes classes de données	variable
<i>paramètres</i>	Liste des paramètres de l'outil	Liste d'instances des classes de données numériques	variable
<i>résultats</i>	Résultat de l'exécution de l'outil	Instance d'une classe de données numériques	variable
<i>type_o</i>	Type du code à exécuter	[<i>Lisp</i> , <i>C</i> , <i>Pandore</i>]	fixe
<i>père</i>	Tâche représentée par le nœud père dans le graphe de tâches	Instance de la classe TACHES	variable

<i>appel</i>	Nom de la fonction ou de la commande à appeler	Chaîne de caractères	fixe
<i>statut</i>	Etat de l'outil lors de l'exécution du graphe de tâches	[<i>exécuté, nil</i>]	variable
<i>mode_exe</i>	Mode d'exécution de l'opérateur	[<i>normal, optimise, pour, tant_que, jusqu_a</i>]	fixe
<i>mode_opti</i>	Mode d'optimisation si c'est <i>mode_exe = optimise</i>	[<i>premier, meilleur</i>]	fixe
<i>nb_iterations</i>	Nombre d'itérations à effectuer si <i>mode_exe = pour</i>	entier	fixe
<i>évaluation</i>	Fonction d'évaluation associée si <i>mode_exe = optimise, jusqu_a</i> ou <i>tant_que</i>	Instance de la classe des fonctions d'évaluation	fixe

Une fonction d'évaluation est définie par le nom de la fonction à exécuter, ses entrées, ses sorties, ses paramètres et son résultat.

Les classes de données

La modélisation des traitements de tous niveaux sous forme d'arbre de tâches entraîne une gestion des flux de données complexe. En particulier, dans les plans de TI où des choix sont possibles, les données doivent être capables de retrouver elles-mêmes le chemin à parcourir dans l'arbre de tâches. Pour éviter les confusions, nous devons réinitialiser les données des entrées, sorties, paramètres et résultats à chaque exécution. Ceci est vrai à l'exception des constantes de calcul nécessaires aux opérateurs (par exemple, la connexité utilisée), des données à demander systématiquement à l'utilisateur (par exemple, l'image d'entrée de la tâche racine d'un plan) et de quelques données que l'on peut considérer comme des variables globales du système (par exemple, l'agenda des tâches, l'utilisateur courant, le contexte d'une session, ...).

Pour gérer ces différents types de données, nous avons défini plusieurs classes :

- **PERMANENTS** : la classe des données constantes. Une instance est définie par les champs *valeur* (valeur définie à la création de l'objet), *domaine-valeur* (ensemble des valeurs possibles) et *dénomination* (ce que représente la donnée), les deux derniers champs n'étant utilisés que pour la communication avec l'utilisateur.
- **VALUTI** : la classe des données dont la valeur doit être demandée à l'utilisateur. Une instance est définie par les champs *domaine-valeur*, *dénomination*, *par-défaut* (valeur à utiliser par défaut) et *valeur* (la valeur saisie par l'utilisateur). La demande d'une valeur à l'utilisateur est gérée par un réflexe de type si-besoin. Le système fournit alors à l'utilisateur les informations nécessaires à la saisie (domaine de valeur, dénomination et valeur par défaut).

- **MODELE** : la classe des données calculées ou déduites. Une instance est définie par les champs *domaine-valeur*, *valeur* (la dernière valeur affectée à l'élément) et *ou-trouver*. Le champ *ou-trouver* indique où l'élément peut retrouver sa valeur en cas de besoin ; il est de la forme (ou (I, c, n) (I, c, n) ...) où I est une instance de TACHES ou d'OUTILS, c est un champ de I (entrées, sorties, paramètres ou résultats) et n est le numéro d'ordre de l'élément du champ où se trouve la valeur (par exemple : aller chercher la valeur dans la deuxième entrée de la tâche T1 sera noté (ou (T1 entrées 2))). La valeur de la donnée est gérée par réflexe : parmi les choix proposés dans le champ ou-trouver, on recherche la tâche ou l'outil qui possède un père (relié à la méthode choisie) et qui a été exécuté (ses données possèdent des valeurs).

La classe CAS

Chaque cas est défini par un objet. Un cas possède un champ *nom_tache* qui est le nom de la tâche racine du plan solution du cas, un champ *fichier* qui est le nom du fichier dans lequel est sauvegardé la tâche racine, un champ *domaine* qui contient le domaine d'application du cas et qui est utilisé uniquement pour informer l'utilisateur, et un champ *critères* qui est une liste d'association (critère, valeur). Comme nous l'avons précisé au chapitre précédent, notre ensemble de critères n'étant pas exhaustif, sa modélisation sous forme d'une liste d'association nous permet de le modifier facilement. Chaque critère de cette liste est une instance de la classe CRITERE définie par les champs *importance* (coefficient d'importance compris entre 0 et 1), *obligatoire?* (vrai si c'est un critère lié à la tâche et faux si c'est un critère lié à l'image) et *domaine* (le domaine de valeur). Le type du critère permettant en particulier de savoir comment comparer deux valeurs dépend de la sous-classe à laquelle il appartient. Les définitions des critères actuellement répertoriés sont présentées dans le tableau ci-dessous. Pour les critères liés au contexte des images, le coefficient dépend du cas et des décisions de l'utilisateur.

critère	obligatoire?	importance	type	domaine
verbes-but	vrai	0,4	multivalué	Ensemble des verbes
noms-but	vrai	0,3	multivalué	Ensemble des noms
adjectifs-but	vrai	0,3	multivalué	Ensemble des adjectifs
phase-traitement	vrai	0 à 1	symbolique strict	Ensemble des phases
niveau_abstraction	vrai	0 à 1	symbolique gradué	{opérationnel, fonctionnel, intentionnel}
type_bruit	faux	0 à 1	symbolique strict	Ensemble des types de bruit
quantité_bruit	faux	0 à 1	symbolique gradué	{nul, très-faible, faible, moyen, fort, très-fort}
contraste	faux	0 à 1	symbolique gradué	{très-faible, faible, moyen, fort, très-fort}
présence_fond	faux	0 à 1	symbolique strict	{oui, non}

aspect_objets	faux	0 à 1	multivalué	Ensemble des symboles
formes_objets	faux	0 à 1	multivalué	Ensemble des symboles
taille_objets	faux	0 à 1	numérique gradué	[1, 5]
position_objets	faux	0 à 1	multivalué	Ensemble des positions
relation_objets	faux	0 à 1	multivalué	Ensemble des symboles

Pour éviter que l'emploi de deux symboles différents pour exprimer la même chose ne fausse le calcul de similarité, des listes de symboles possibles ont été définies pour certains critères. Nous donnons la composition de certains d'entre eux ci-après. Ces listes sont issues de nos besoins et ne sont bien sûr pas exhaustives, elles devront être complétées par de nouvelles expérimentations.

- Ensemble des verbes : amincir, binariser, calculer, classier, créer, convertir, détecter, déterminer, dilater, éliminer, épaissir, éroder, étiqueter, extraire, former, inverser, isoler, lisser, localiser, numéroter, reconstruire, réduire, préserver, segmenter, sélectionner, séparer, seuiller, supprimer, trier, unir, ...
- Ensemble des noms : arc, bord, bruit, carte de régions, centre de gravité, chaîne de contours, composante connexe, convexité, contour, contraste, croissance, différence, distance, étiquette, fond, forme, frontière, germe, graphe, maximum, minimum, niveau de gris, objet, pixel, point, région, regroupement, sommet, surface, valeur, variance, zone, ...
- Ensemble des adjectifs : bas, commun, complémentaire, extérieur, interclasse, intraclasse, grand, global, haut, intérieur, local, partiel, petit, régional, total, uniforme, ...
- Ensemble des phases (et types de traitement) : restauration, détection, segmentation, extraction d'attributs, pré-traitement, détermination des germes, localisation des contours, partition en régions, regroupement/décomposition, caractérisation de textures, ...
- Ensemble des types de bruit : gaussien, exponentiel, uniforme, ...
- Ensemble des positions : haut, centre, bas, gauche, milieu, droite.

Autres classes utilisées

Nous présentons ici deux classes dont les instances sont utilisées par les tâches de métacontrôle pour la gestion des différentes fonctionnalités proposées à l'utilisateur :

- UTILISATEUR : un utilisateur est défini par son *nom* et son *niveau* (entier entre 1 et 3),
- CONTEXTE : le contexte d'une session connaît toutes les informations nécessaires à la mise en œuvre d'une fonctionnalité. Il est défini par l'utilisateur_courant (instance de la classe utilisateur), le fichier_d_exécution_courant (le fichier qui contient les éléments du plan sur lequel

on travaille en exécution) et le fichier_de_création_courant (le fichier qui contient les éléments du plan sur lequel on travaille en création).

V.1.2. Les fonctionnalités de l'interface graphique

L'interface graphique du système propose différentes fonctionnalités qui sont présentées dans les menus déroulants de la fenêtre principale (fig.55). Chacune de ces fonctionnalités est modélisée sous forme d'un plan de contrôle ou de métacontrôle.

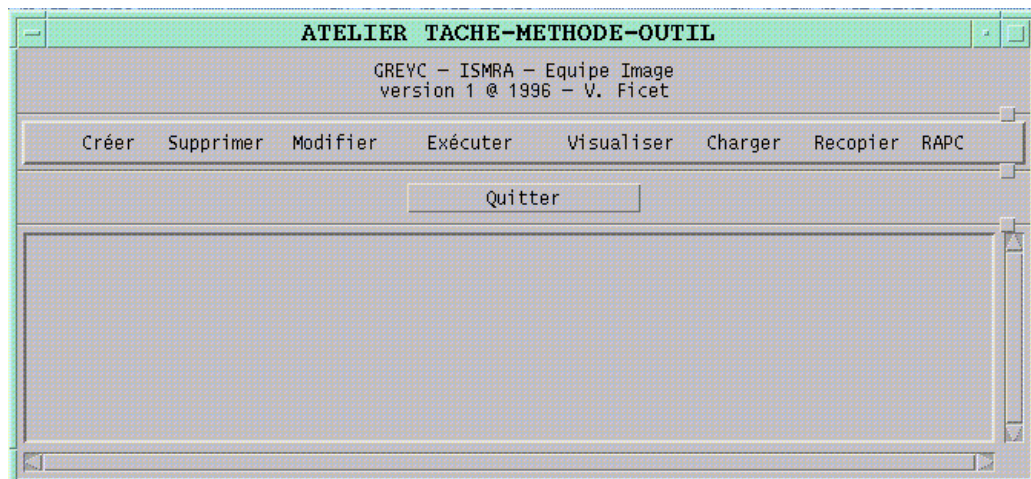


Fig.55 : fenêtre principale de l'interface graphique

Nous détaillons ci-dessous chacun des menus proposés et l'allure générale de l'arbre de tâches correspondant.

Menu Créer

Le menu *Créer* propose trois options pour créer, soit une tâche, soit une méthode, soit un outil en définissant les valeurs des attributs et les relations entre éléments (pour les méthodes). Les flux de données étant dépendants de la méthode choisie, ils sont définis lors de la création de la méthode (fig.57). Les plans de création d'une tâche et de création d'un outil étant similaires nous n'en présentons qu'un des deux (fig.56).

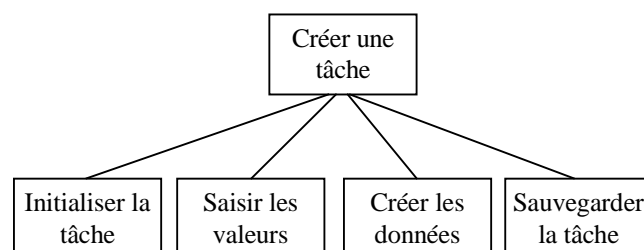


Fig.56 : plan pour la création d'une tâche

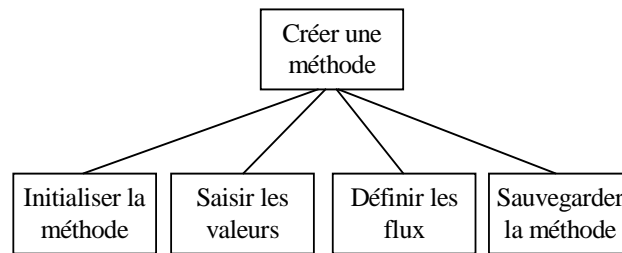


Fig.57 : plan pour la création d'une méthode

Menu Supprimer

Le menu *Supprimer* propose trois options pour supprimer, soit une tâche, soit une méthode, soit un outil. Les plans de suppression d'une tâche (ou d'un outil) (fig.58) et d'une méthode (fig.59) sont présentés ci-dessous.

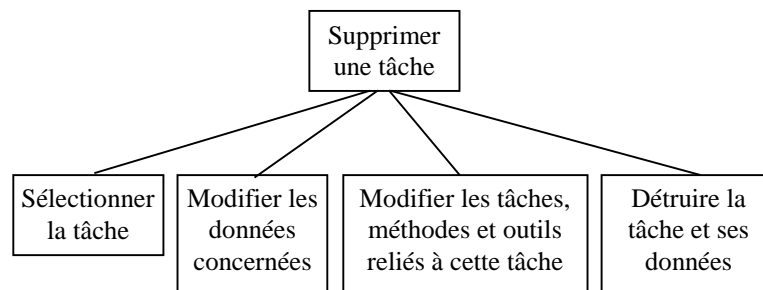


Fig.58 : plan pour la suppression d'une tâche

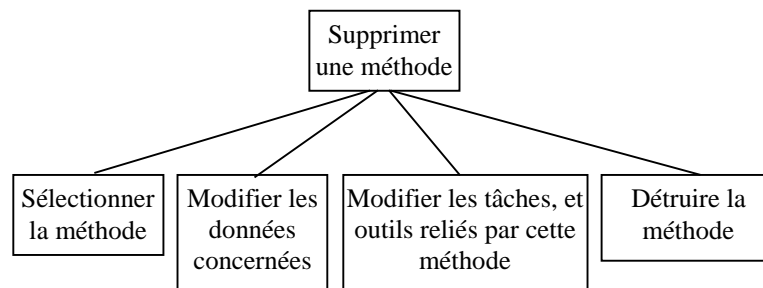


Fig.59 : plan pour la suppression d'une méthode

Menu Modifier

Le menu *Modifier* propose trois options pour modifier, soit une tâche, soit une méthode, soit un outil. Nous présentons ci-dessous l'allure générale des plans de modification d'une tâche (fig.60) et de modification d'une méthode (fig.61), le plan de modification d'un outil est le même que celui de modification de la tâche.

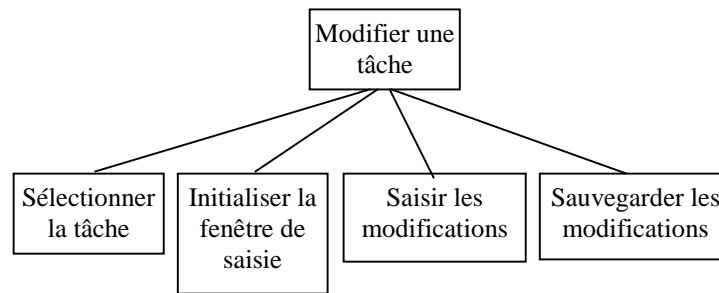


Fig.60 : plan pour la modification d'une tâche

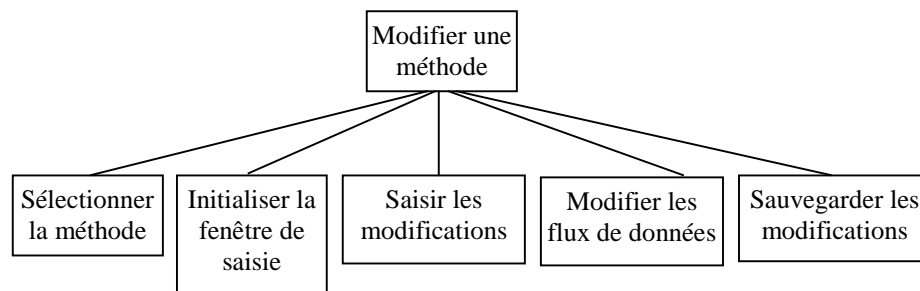


Fig.61 : plan pour la modification d'une méthode

Menu Exécuter

Le menu *Exécuter* permet d'exécuter un plan de TI (fig.62) en sélectionnant la tâche racine de ce plan.

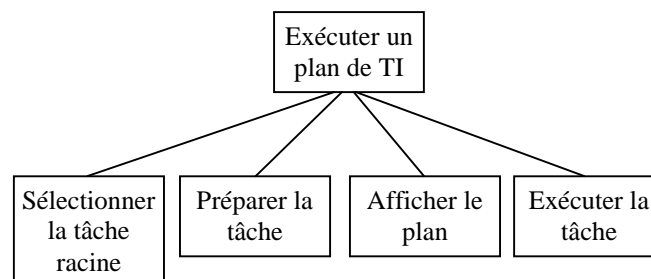


Fig.62 : plan pour l'exécution d'un plan de TI

Menu Visualiser

Le menu *Visualiser* permet de visualiser un plan de TI (fig.63) sous forme d'une représentation schématique des tâches, des méthodes et des outils en sélectionnant la tâche racine de ce plan.

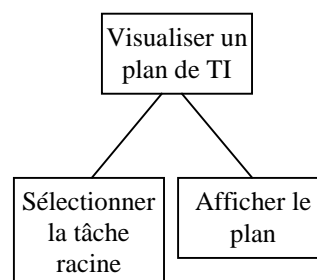


Fig.63 : plan pour la visualisation d'un plan de TI

Menu Charger

Le menu *Charger* propose deux options pour modifier le contexte de la session, soit modification du fichier de création courant, soit modification du fichier d'exécution courant. Dans les deux cas il s'agit d'abord de sélectionner le fichier désiré pour modifier le contexte puis de charger les éléments du plan contenus dans ce fichier.

Menu Recopier

Le menu *Recopier* propose trois options pour recopier une tâche, un outil ou un plan. Ces options permettent de dupliquer des éléments connus pour éviter de les redéfinir entièrement. Les trois plans sont similaires (fig.64) même si les codes associés aux outils sont plus complexes pour la recopie d'un plan complet.

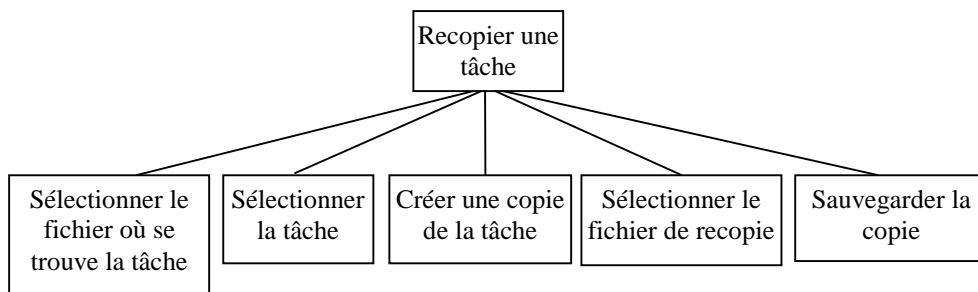


Fig.64 : plan pour la recopie d'une tâche

Menu RàPC

Le menu *RàPC* propose trois options correspondant aux différentes phases du RàPC : rechercher un cas source (fig.65), adapter un cas (fig.66) et mémoriser un cas (fig.67).

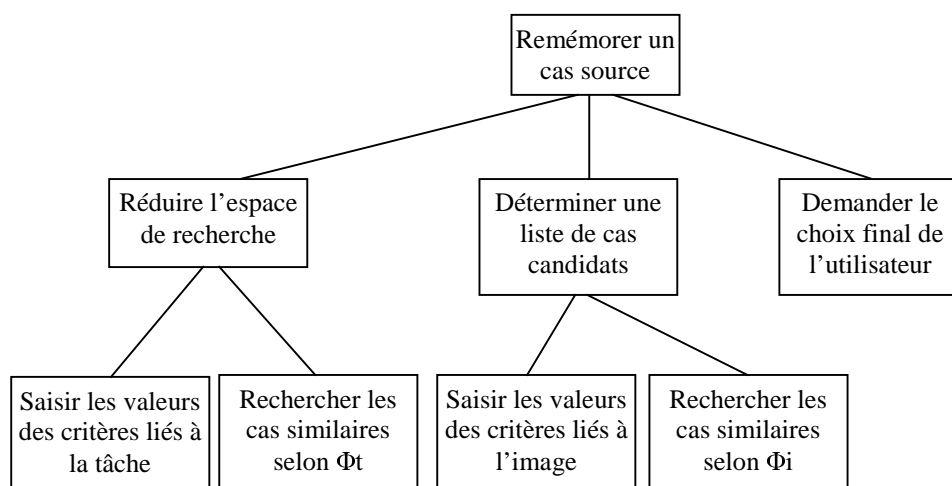


Fig.65 : plan pour la recherche d'un cas source

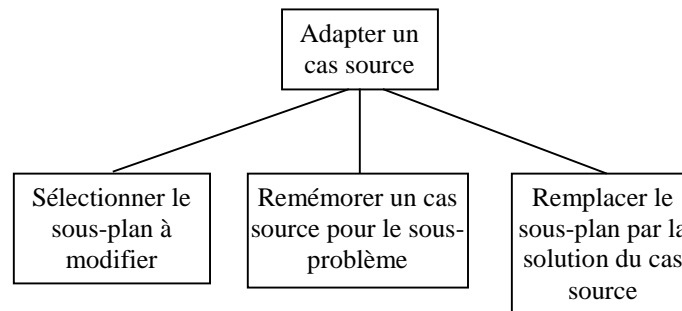


Fig.66 : plan pour l'adaptation d'un cas source

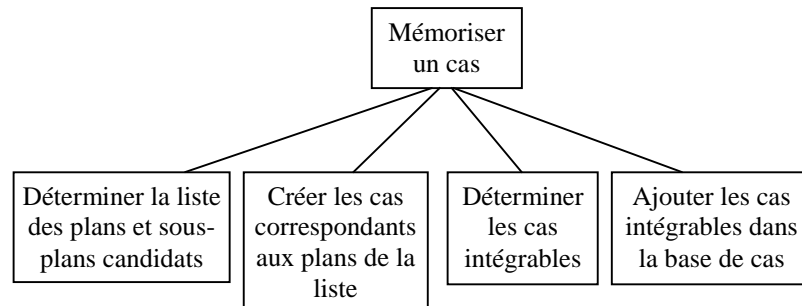


Fig.67 : plan pour la mémorisation d'un nouveau cas

V.2. Déroulement d'une session de création d'un plan à l'aide de l'interface

La création d'un plan peut se faire, soit en suivant une démarche ascendante, c'est-à-dire par regroupement de tâches simples en tâches de plus en plus complexes (des feuilles vers la racine de l'arbre), soit en suivant une démarche descendante, c'est à dire en décomposant des tâches complexes en tâches de plus en plus simples, soit encore en alternant les deux démarches. L'utilisateur peut définir ses tâches et ses outils dans l'ordre qu'il veut avant de les relier en définissant les méthodes. Nous présentons dans la suite de cette section quelques unes des étapes de la création d'un plan de TI qui a été construit pour traiter des images de cytologie dans lesquelles on cherche à quantifier la présence de contacts focaux sur les bords de cellules de culture. Il a été mis au point par François Angot [Angot 99] dans le cadre d'une étude réalisée en collaboration avec le service de Cancérologie Expérimentale du Centre Régional de Lutte Contre le Cancer de Caen. Ce plan correspond au plan C de la section suivante.

Lors de la demande de création d'une tâche, la fenêtre de saisie des caractéristiques de la tâche (fig.68) apparaît à l'écran. L'utilisateur doit alors remplir les champs *Titre* (description du but en 250 caractères maximum), *Dénomination* (nom différenciant la tâche des autres dans une liste à choix en 30 caractères maximum), *Niveau* (par défaut le niveau domaine), *Nb d'entrées* (nombre

d'entrées, par défaut 0), *Nb de sorties* (par défaut 0), *Nb de résultats* (par défaut 0), *Nb de paramètres* (par défaut 0) et *Niveau requis* (par défaut 1)

The image shows a Windows-style dialog box titled "Création d'une tâche". It has several input fields and radio buttons. The fields are: "Titre" with the text "localiser les petits objets brillants", "Dénomination" with "petits obj bri", "Nb d'entrées" with "1", "Nb de sorties" with "2", "Nb de résultats" with "0", and "Nb de paramètres" with "1". There are three radio buttons for "Niveau": "Domaine" (selected), "Contrôle", and "MétaContrôle". At the bottom, there are three radio buttons for "Niveau requis": "1" (selected), "2", and "3". At the very bottom are three buttons: "OK", "Annuler", and "Aide".

Fig.68 : fenêtre de saisie des caractéristiques d'une tâche

La validation d'un élément (tâche, méthode ou outil) entraîne sa sauvegarde dans le fichier de création courant. Elle se fait en cliquant sur le bouton OK et n'est réalisable que si tous les champs indispensables sont remplis. Dans le cas contraire un message d'erreur s'affiche (fig.69).

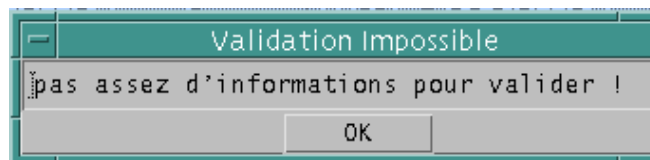


Fig.69 : message d'erreur quand la validation d'un élément n'est pas possible

De même, lors de la demande de création d'un outil, la fenêtre adaptée apparaît (fig.70) et l'utilisateur doit alors saisir les champs *Titre* (description du but en 250 caractères maximum), *Dénomination* (nom différenciant l'outil des autres dans une liste à choix en 30 caractères maximum), *Type* (par défaut Pandore), *Mode d'exécution* (par défaut Normal), *Nb d'entrées* (nombre d'entrées, par défaut 0), *Nb de sorties* (par défaut 0), *Nb de paramètres* (par défaut 0) et *Opérateur ou fonction*. Si le mode d'exécution choisi est « pour », l'utilisateur doit remplir le champ *Nb d'itérations* (par défaut 1) ; si ce mode est « optimisation », il doit remplir les champs *Mode d'optimisation* et *Fct d'évaluation* ; si ce mode est « tant que » ou « jusqu'à », il doit remplir le champ *Fct d'évaluation*.

Création d'un outil

Titre : éroder n fois pour localiser le fond

Dénomination : nerod loc fond

Type : ☐ Lisp ☒ Pandore ☐ C

Mode d'exécution : ☒ Normal ☐ Optimisation ☐ Pour ☐ Tant que ☐ Jusqu'à

Mode d'optimisation : ☐ Premier ☐ Meilleurs

Nb d'itérations : 1

Nb d'entrées : 1

Nb de sorties : 1

Nb de paramètres : 2

Fct d'évaluation : 1

Opérateur ou fonction : nerosion

OK Annuler Aide

Fig.70 : fenêtre de saisie des caractéristiques d'un outil

Lorsque les tâches et les outils à relier sont définis, on peut créer une méthode explicitant la relation entre ces différents éléments. La création d'une méthode se déroule en plusieurs étapes. La première consiste à choisir la « Tâche Mère » de la méthode, c'est-à-dire, la tâche que la méthode doit résoudre. Ceci est effectué par sélection d'une tâche parmi celles du fichier de création courant (fig.71).

Tâche Mère

49 dil ptt obj bril

50 erod ptt obj bril

51 el pts loin bords

52 epaissir bords

53 bords reg

54 el ptts obj

55 el pttes parties

56 localiser fond

57 petits obj bril

58 loc pts bril

OK Aide

Fig.71 : fenêtre de sélection de la tâche mère d'une méthode

La deuxième étape consiste à remplir les champs propres à la méthode (fig.72). Il s'agit des champs *Titre* (description de la méthode en 250 caractères maximum), *Dénomination* (nom différenciant la méthode des autres dans une liste à choix en 30 caractères maximum), *Niveau requis* (par défaut 1) et *Descendant* (outil ou décomposition en sous-tâches).

Fig.72 : fenêtre de saisie des caractéristique d'une méthode

Ensuite, suivant qu'il s'agit d'une méthode terminale ou d'une méthode complexe, il faut choisir l'outil associé ou la suite de sous-tâches associée (fig.73), la fenêtre correspondant à ce choix s'ouvrant automatiquement après la sélection du *Descendant*.

Fig.73 : fenêtre de sélection de la suite de sous-tâches lors de la création d'une méthode

Il faut enfin définir les liens existant entre les données de la tâche mère et celles des sous-tâches ou de l'outil. Les entrées et sorties doivent obligatoirement être reliées entre elles. La définition des relations se fait par l'intermédiaire de boutons modélisant les données. Sur l'exemple de la figure 74, pour indiquer que la première sortie de la tâche « localiser petits objets brillants » (modélisée par le bouton **S000**) doit aller chercher sa valeur dans la sortie de « dilater pour petits objets brillants » (modélisée par le bouton **S003**), on les reliera en cliquant d'abord sur le bouton **S003** (bouton émetteur) puis sur le bouton **S000** (bouton récepteur), ce dernier prendra alors le nom **S003**. Lorsque l'on clique sur un bouton émetteur, celui-ci reste enfoncé jusqu'à la sélection d'un bouton récepteur ou l'annulation de la sélection en cliquant à nouveau. Pour annuler un lien, il suffit de définir le lien qui le remplace.

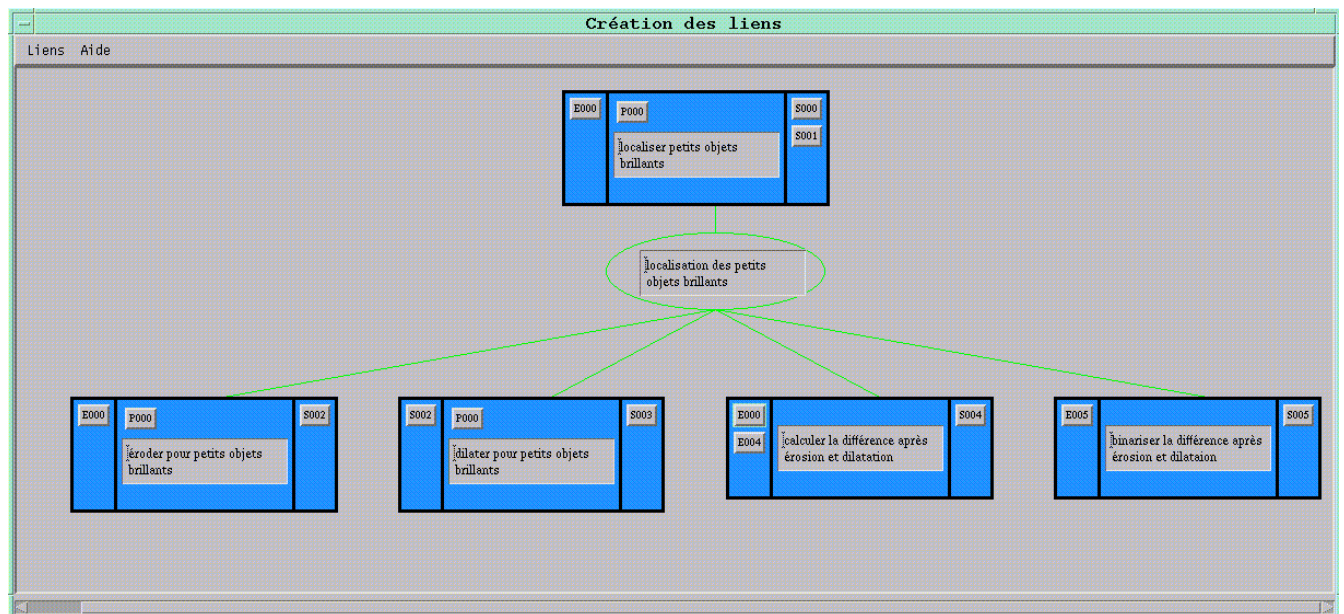


Fig.74 : fenêtre pour la définition des flux de données d'une méthode complexe

Les paramètres peuvent, soit être reliés à d'autres paramètres ou à des résultats (comme les entrées et les sorties), soit être définis par des *valeurs fixes* (même valeur pour toutes les exécutions) ou par des *valeurs utilisateur* (à demander à l'utilisateur lors de l'exécution), comme c'est le cas pour les paramètres de l'outil « binariser pour épaissir les bords » dans la figure 75. En cliquant sur un paramètre, s'il n'y a pas de bouton émetteur sélectionné, la fenêtre *définition d'un paramètre* s'ouvre (fig.76).

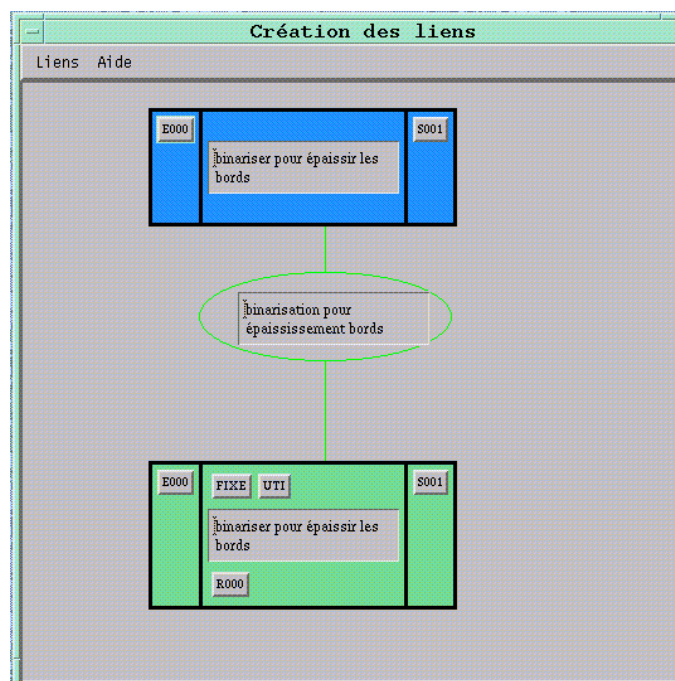


Fig.75 : fenêtre pour la définition des flux de données d'une méthode terminale

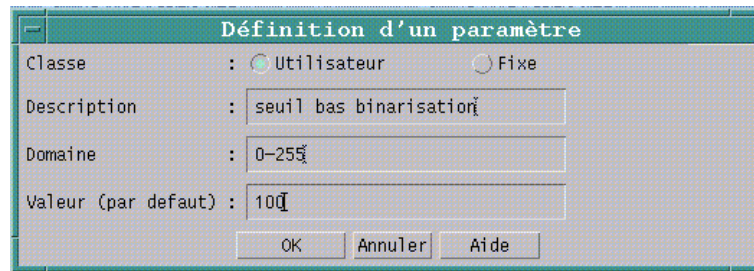


Fig.76 : fenêtre de définition d'un paramètre à « valeur fixe » ou à « valeur utilisateur »

Pour valider les liens entre données, il faut finalement sélectionner *valider* dans le menu *liens* de la fenêtre *créer les liens*.

Si les résultats de l'exécution de son plan ne sont pas satisfaisants, l'utilisateur peut demander la modification de certains éléments. Le système lui propose de sélectionner l'élément à modifier et une fenêtre similaire à celle de la création de l'élément apparaît avec les valeurs des différents champs. Il peut alors modifier les caractéristiques désirées puis les valider.

V.3. Déroulement d'une session d'exécution d'un plan

Lorsque l'utilisateur veut exécuter un plan de TI, le système lui propose de sélectionner la tâche racine de ce plan parmi celles sauvegardées dans le fichier d'exécution courant (fig.77). Si c'est la première exécution de la session ou s'il veut changer de plan, il devra d'abord choisir le fichier d'exécution courant (fig.78).

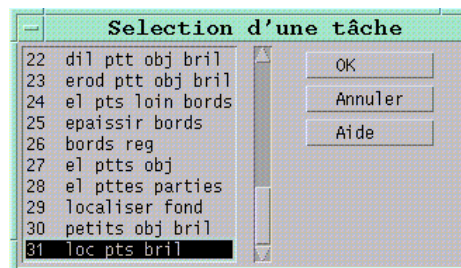


Fig.77 : fenêtre de sélection de la tâche à exécuter

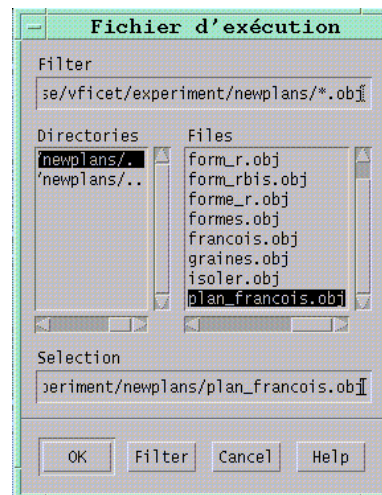


Fig.78 : fenêtre de sélection du fichier d'exécution courant

Le plan dont la tâche sélectionnée est la racine s'affiche alors dans la fenêtre principale sous forme d'un arbre (fig.79).

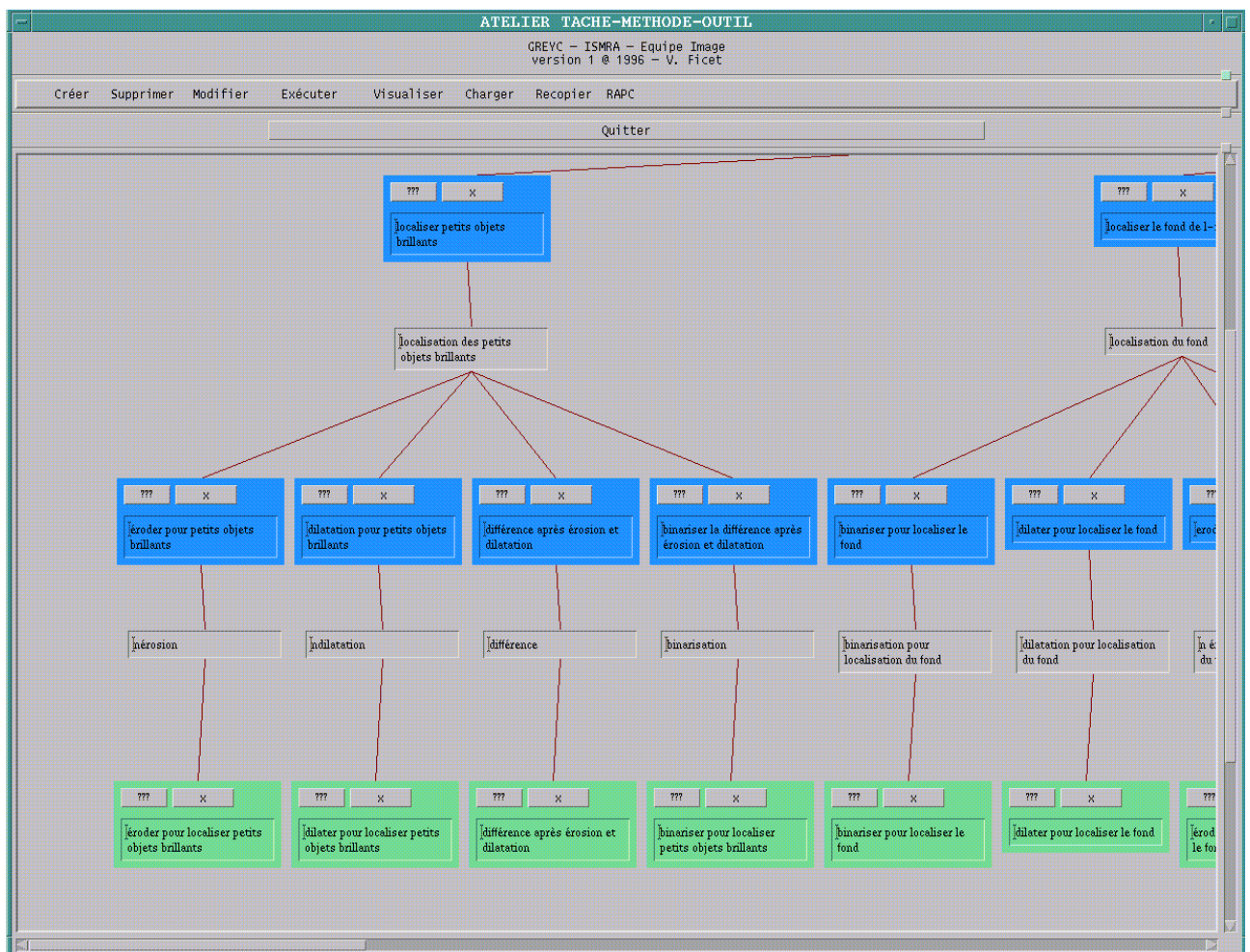


Fig.79 : affichage du plan de TI dans la fenêtre principale

Puis le plan s'exécute en demandant à l'utilisateur au fur et à mesure de sélectionner la (ou les) image(s) d'entrée (fig.80), de donner une valeur aux paramètres définis comme *valeur utilisateur* (fig.81), et de choisir entre les méthodes lorsqu'il en existe plusieurs pour réaliser une tâche.

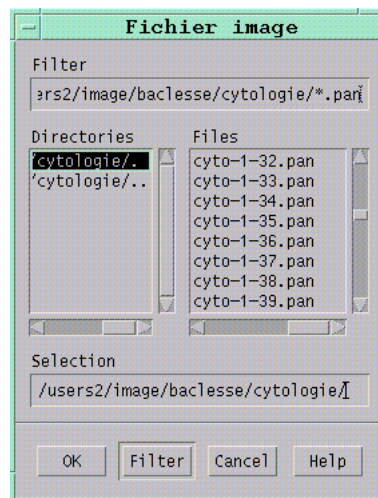


Fig.80 : fenêtre de sélection de l'image d'entrée

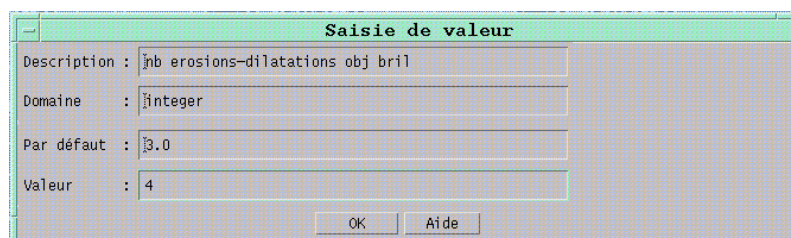




Fig.81 : fenêtre de saisie d'une valeur utilisateur

Il est possible de (ré)exécuter une tâche (ou une sous-tâche) en cliquant sur le bouton  qui lui correspond.

Après exécution, l'utilisateur peut accéder aux informations sur une tâche ou un outil du plan en cliquant sur le bouton  lui correspondant : une fenêtre avec les valeurs de différents champs de la tâche ou de l'outil s'affiche (fig.82 et fig.84), ainsi que les images d'entrée et de sortie (fig.83 et fig.85).

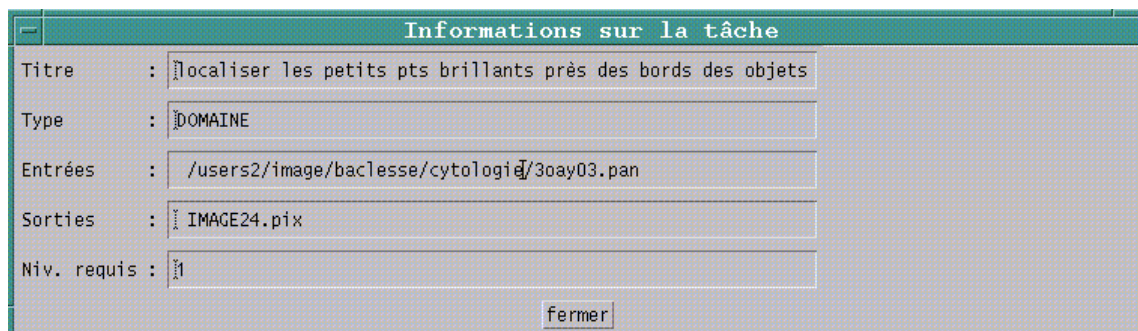


Fig.82 : fenêtre d'information sur une tâche

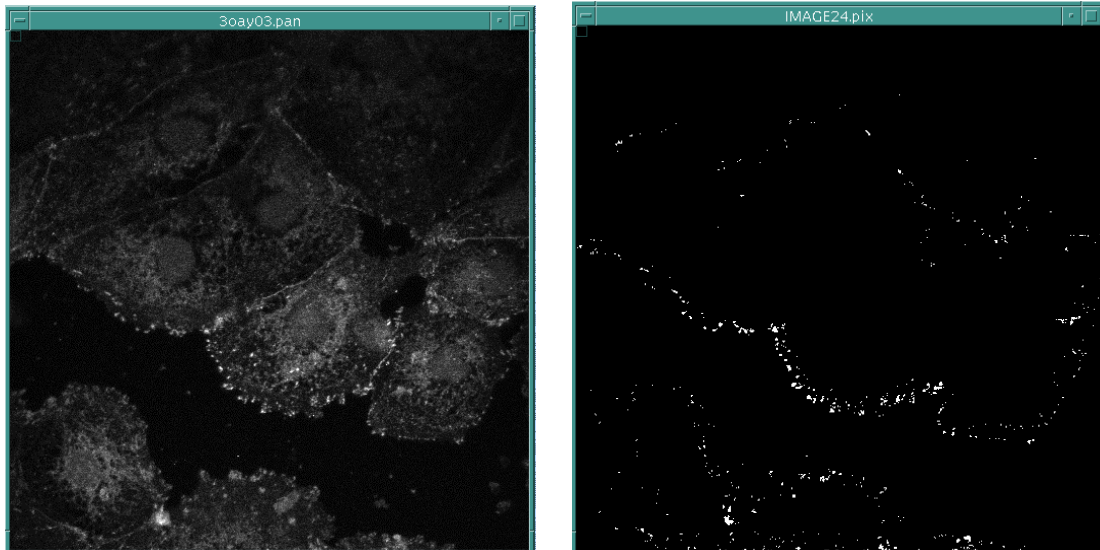


Fig.83: image d'entrée et image de sortie de la tâche

Informations sur l'outil	
Titre	: éroder n fois pour localiser le fond
Type	: PANDORE
Entrées	: IMAGE7.pix
Sorties	: IMAGE8.pix
Paramètres	: 6.0 8.0
Mode d'exécution	: NORMAL
Mode d'optimisation	: <input type="checkbox"/> A
Nombre d'itérations	: 1
Opérateur	: EROSION
<input type="button" value="fermer"/>	

Fig.84 : fenêtre d'information sur un outil

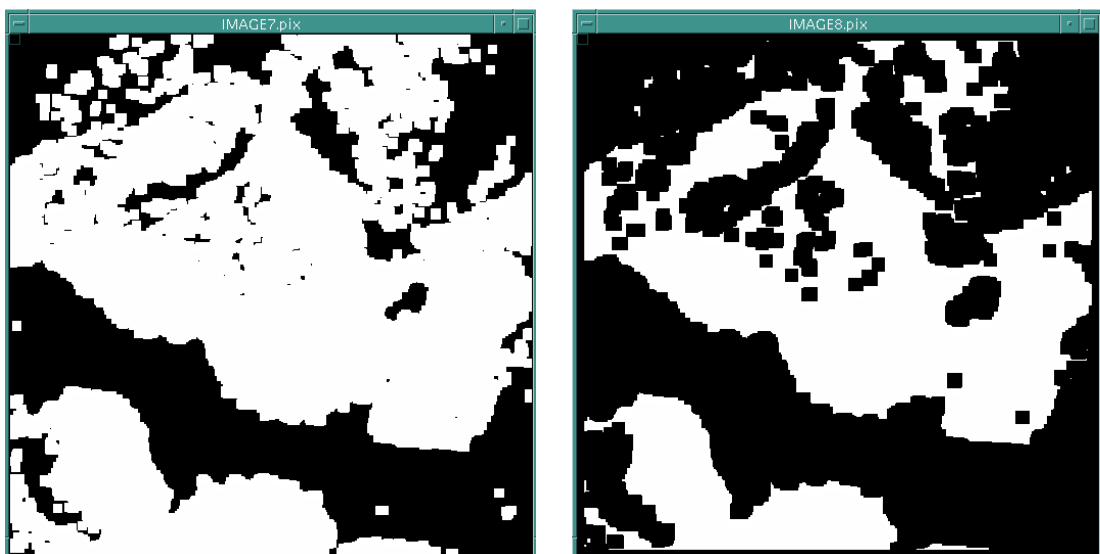


Fig.85 : image d'entrée et image de sortie de l'outil

L'accès à ces informations et en particulier aux images intermédiaires est très important car il permet à l'utilisateur de réviser son plan en modifiant certains éléments ou d'ajuster l'exécution en modifiant les valeurs des paramètres des outils.

V.4. Déroulement d'une session de création d'un plan par RàPC

Recherche d'un cas source

Pour lancer l'algorithme de recherche d'un cas source qui fournisse une première solution à son problème, l'utilisateur doit définir un cas cible par l'intermédiaire d'une fenêtre de saisie (fig.86). Pour indiquer qu'une caractéristique doit être considérée comme un critère important, l'utilisateur coche la case correspondante (devant le nom du critère).

The screenshot shows a window titled "Description du cas cible" with a light green header. The window contains several input fields and checkboxes for defining a target case. The fields are organized into sections:

- Description:**
 - Verbes: EXTRAIRE
 - Noms: OBJET
 - Adjectifs: (empty)
- Phase:** SEGMENTATION
- Niveau:** ☒ intentionnel ☐ fonctionnel ☐ opérationnel
- Type bruit:** ☐ (empty)
- Quantité bruit:** ☐ nul ☐ très faible ☒ faible ☐ moyen ☐ fort ☐ très fort
- Qualité contraste:** ☐ très faible ☐ faible ☒ moyen ☐ fort ☐ très fort
- Présence fond:** ☒ ☐ oui ☐ non
- Aspect objets:** NG CLAIR
- Forme objets:** CONVEXE
- Taille objets:** ☐ très grand ☒ grand ☐ moyen ☐ petit ☐ très petit
- Position objets:** ☐ gauche ☐ milieu ☐ droite ☐ haut ☐ centre ☐ bas
- Relation objets:** CONNEXE

At the bottom of the window, there are three buttons: "Valider", "Annuler", and "Aide".

Fig.86 : fenêtre de saisie des caractéristiques du cas cible

La sélection des termes pour lesquels des listes de termes ont été définies, se fait par l'intermédiaire de listes de choix (fig.87).



Fig.87 : liste de choix pour les termes pour les critères de description du problème

Dans l'exemple décrit ci-dessous, le nouveau problème consiste à isoler les objets dans une image industrielle (fig.88) ; les objets sont de niveau de gris clair et de forme relativement convexe.



Fig.88 : image d'entrée du plan en cours de création

Lorsque l'utilisateur a validé son cas cible, le système lance l'algorithme de remémoration proposé au chapitre précédent. Il retourne alors le résultat de la recherche sous forme d'une liste de 2 à 5 cas sources : sont présentés la similarité entre le cas source et le cas cible suivant les critères liés à l'image et la définition de la tâche associée au cas source (fig.89).

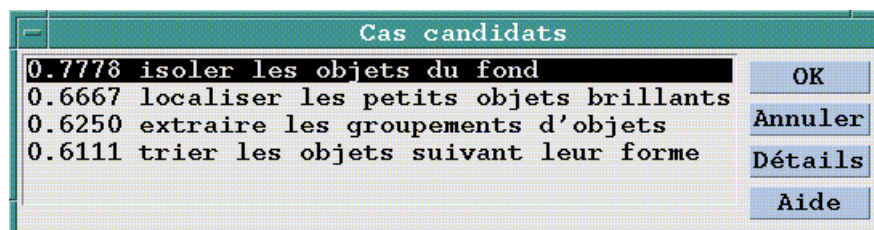


Fig.89 : fenêtre pour le choix du cas source parmi les cas candidats

L'utilisateur peut alors consulter la description d'un cas source en cliquant sur le bouton « Détails » de façon à comparer les différents critères des cas avant de faire son choix. La description du cas apparaît dans une fenêtre adaptée où sont donnés les valeurs des critères, le nom de la tâche racine du plan solution et le domaine d'origine de l'image pour laquelle le plan avait été construit initialement (fig.90). Puis il fera son choix final en cliquant sur le bouton « ok » si un cas lui convient ou sur le bouton « annuler » sinon.

Description du cas source	
Tâche concernée : isoler les objets du fond	
<input type="checkbox"/> Description	Verbes : isoler Noms : objet fond Adjectifs :
<input type="checkbox"/> Phase	segmentation
<input type="checkbox"/> Niveau	<input checked="" type="radio"/> intentionnel <input type="radio"/> fonctionnel <input type="radio"/> opérationnel
<input type="checkbox"/> Type bruit	gaussien
<input type="checkbox"/> Quantité bruit	<input type="radio"/> nul <input type="radio"/> très faible <input checked="" type="radio"/> faible <input type="radio"/> moyen <input type="radio"/> fort <input type="radio"/> très fort
<input type="checkbox"/> Qualité contraste	<input type="radio"/> très faible <input type="radio"/> faible <input checked="" type="radio"/> moyen <input type="radio"/> fort <input type="radio"/> très fort
<input type="checkbox"/> Présence fond	<input checked="" type="radio"/> oui <input type="radio"/> non
<input type="checkbox"/> Aspect objets	ng homogene fonce
<input type="checkbox"/> Forme objets	convexe
<input type="checkbox"/> Taille objets	<input type="radio"/> très grand <input checked="" type="radio"/> grand <input type="radio"/> moyen <input type="radio"/> petit <input type="radio"/> très petit
<input type="checkbox"/> Position objets	<input type="radio"/> gauche <input type="radio"/> milieu <input type="radio"/> droite <input type="radio"/> haut <input type="radio"/> centre <input type="radio"/> bas
<input type="checkbox"/> Relation objets	
<input type="checkbox"/> Domaine image	cytologie
OK	

Fig.90 : fenêtre de description d'un cas source

Pour illustrer la phase d'adaptation nous supposons que l'utilisateur a choisi le premier cas de la liste correspondant à la tâche « isoler les objets du fond » ; il correspond au plan A détaillé dans la section suivante.

Adaptation du cas source au cas courant

L'utilisateur peut commencer par visualiser le plan solution du cas source pour étudier la stratégie proposée (fig.91). Une première modification consiste à supprimer la tâche « calculer

l'image complémentaire » : le plan initial a été conçu pour isoler des objets foncés sur un fond clair alors qu'ici les objets à isoler sont clairs sur un fond foncé. Ceci correspond en fait à remplacer la tâche « marquer les régions » par la tâche « étiqueter les régions ».

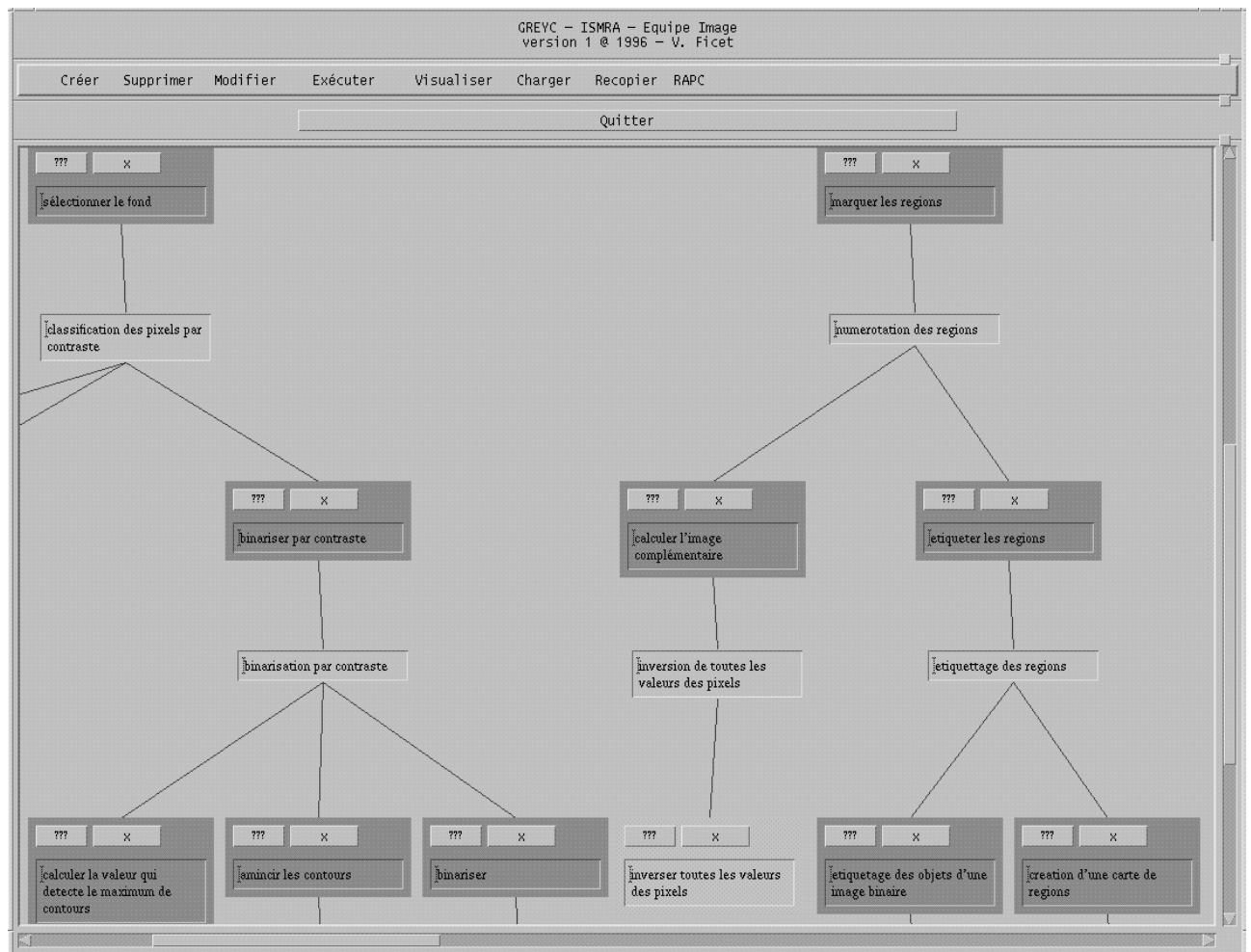


Fig.91 : visualisation du plan solution proposé

Puis il peut exécuter le plan pour analyser les résultats. Ici les résultats finaux sont insatisfaisants : l'ensemble des objets a été détecté, mais ils ne sont pas différenciés les uns des autres (fig.92). On va donc chercher à adapter le plan en relançant l'algorithme pour trouver un autre sous-plan pour former les différents objets.



Fig.92 : résultat après exécution du plan initial

Pour cela, on définit un nouveau cas cible correspondant au sous-problème à résoudre. Ici le système va proposer plusieurs solutions dont la tâche « former les objets à partir des régions » qui est issue d'un plan créé pour détecter des regroupements d'objets (voir plan B section suivante). Cette tâche va remplacer celle du même nom dans le plan initial, ces deux tâches possèdent le même nom car elles ont le même objectif, mais elles utilisent deux stratégies différentes. On obtient ainsi un nouveau plan qui peut à nouveau être testé. Quelques améliorations locales peuvent ensuite être réalisées : par exemple, on va modifier la technique de détection des germes nécessaires à la Lpe. Le résultat n'est pas encore parfait (fig.93) et le plan devra être amélioré, mais les objets sont pour la plupart dissociés les uns des autres et leurs formes irrégulières sont mieux respectées.



Fig.93 : résultat après adaptation

Le plan résultant correspond au plan F présenté dans la section suivante. La figure 94 montre les premiers niveaux de ce plan : le sous-plan A1 provient du plan initialement remémoré et les sous-plans F1 et F2 sont les parties qui ont subies des modifications ou qui ont été remplacées.

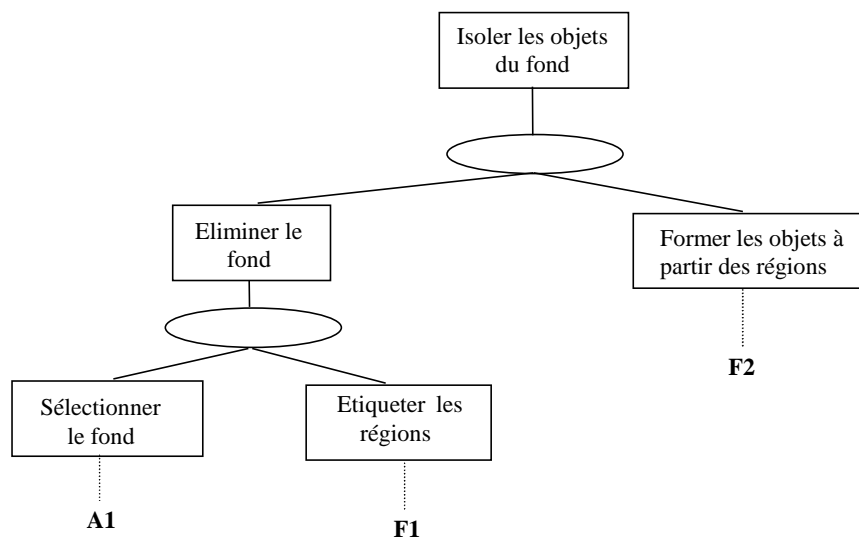


Fig.94 : premiers niveaux de décomposition du plan résultat adapté

Mémorisation du nouveau cas

Une fois l'adaptation terminée, on peut définir les nouveaux cas à intégrer à la base. Pour cela, le système va d'abord déterminer l'ensemble des sous-plans candidats à la définition d'un nouveau cas : ils correspondent aux sous-plans ayant subi une adaptation et à leurs ascendants. Ici sont candidats : les plans de tâche racine « étiqueter les régions », « former les objets à partir des régions », « éliminer le fond » et « isoler les objets du fond ». Le plan de la tâche racine « étiqueter les régions » est éliminé car de trop petite taille (inférieur à trois niveaux de tâches).

Ensuite pour chacun des plans candidats, l'utilisateur doit définir le cas correspondant. Pour cela, il fournit les valeurs des différents critères dans une fenêtre adaptée (fig.95). Certains cas seront entièrement décrits, d'autres, correspondant à un cas cible précédemment défini seront juste complétés. Ainsi le cas correspondant au cas cible initial sera complété par l'ajout d'un adjectif dans la définition du problème, cet adjectif précisant que l'on veut des objets « séparés ».

The screenshot shows a window titled "Saisie d'un nouveau cas" with a light green header. Below the header, the "Tâche concernée" is set to "former les objets à partir des régions". The main area contains a list of criteria on the left and their corresponding input fields on the right. The criteria and their values are as follows:

Critère	Valeur
Verbes	séparer
Noms	objet
Adjectifs	
Phase	décomposition
Niveau	<input type="radio"/> intentionnel <input checked="" type="radio"/> fonctionnel <input type="radio"/> opérationnel
Type bruit	
Quantité bruit	<input type="radio"/> nul <input type="radio"/> très faible <input type="radio"/> faible <input type="radio"/> moyen <input type="radio"/> fort <input type="radio"/> très fort
Qualité contraste	<input type="radio"/> très faible <input type="radio"/> faible <input type="radio"/> moyen <input type="radio"/> fort <input type="radio"/> très fort
Présence fond	<input checked="" type="radio"/> oui <input type="radio"/> non
Aspect objets	ng clair homogène
Forme objets	convexe
Taille objets	<input type="radio"/> très grand <input checked="" type="radio"/> grand <input type="radio"/> moyen <input type="radio"/> petit <input type="radio"/> très petit
Position objets	<input type="radio"/> gauche <input type="radio"/> milieu <input type="radio"/> droite <input type="radio"/> haut <input type="radio"/> centre <input type="radio"/> bas
Relation objets	connexe
Domaine image	industriel

At the bottom of the window, there are three buttons: "Valider", "Annuler", and "Aide".

Fig.95 : définition des nouveaux cas candidats à l'intégration dans la base

Enfin pour chacun des cas candidats, on calcule la similarité maximale avec les cas de la base : si elle est inférieure à 0,9, on intègre le cas. Dans l'exemple, deux des trois cas seront intégrés, leur similarité maximale étant de 0,6 pour le cas associé à la tâche « isoler les objets du fond » et de 0,87 pour celui associé à la tâche « former les objets à partir des régions », le troisième correspondant à la tâche « éliminer le fond » ne sera pas intégré car sa similarité maximale est de 0,95.

V.5. Quelques exemples de plans intégrés

Dans cette section, nous présentons quelques exemples de plans intégrés dans notre système. Ces plans nous ont permis de tester notre système suivant trois axes principaux : validation du modèle de l'architecture, expérimentation de l'interface par un utilisateur novice et recherche de similarités entre applications de différents domaines.

Les trois premiers plans nous ont permis de valider le modèle TMO et l'architecture du système en prenant en compte des applications réelles. Ils ont été mis au point au laboratoire par A. Elmoataz [Revenu 93] et F. Angot [Angot 99] pour traiter des images biomédicales de cytologie et d'histologie fournies par le Centre de Lutte Contre le Cancer F. Baclesse de Caen. Ces images sont de bons supports pour tester notre système car ils soulèvent une grande diversité de problèmes concrets.

Le quatrième plan a été intégré dans le système par un novice du système qui était également débutant en TI. Cette expérience avait pour but de tester la facilité de prise en main du système par un nouvel utilisateur et de valider les fonctionnalités proposées par l'interface graphique.

Les deux derniers plans s'appliquent à des images d'origines différentes (image de synthèse et image industrielle). Leur intégration dans le système nous a permis de rechercher les critères de description de problème communs à des applications de différents domaines et de tester le module de RàPC.

Sur le premier plan, nous présentons en détail les relations entre les différentes données (entrées, sorties, résultats, paramètres) des tâches et des outils. Pour simplifier les schémas, sur les suivants, nous ne présentons que la stratégie décrite par l'enchaînement des différents éléments. Pour chaque plan, nous montrons un exemple d'exécution en fournissant une image d'entrée et la (ou les) image(s) de sortie.

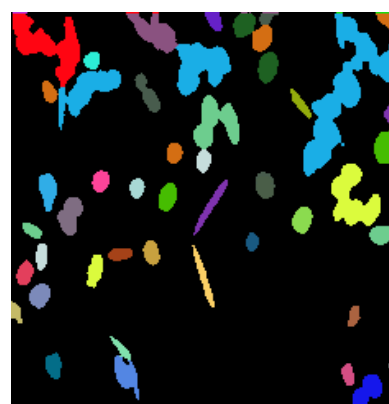
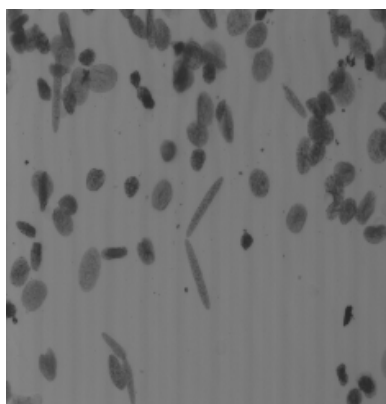
Plan A : Isoler les objets du fond

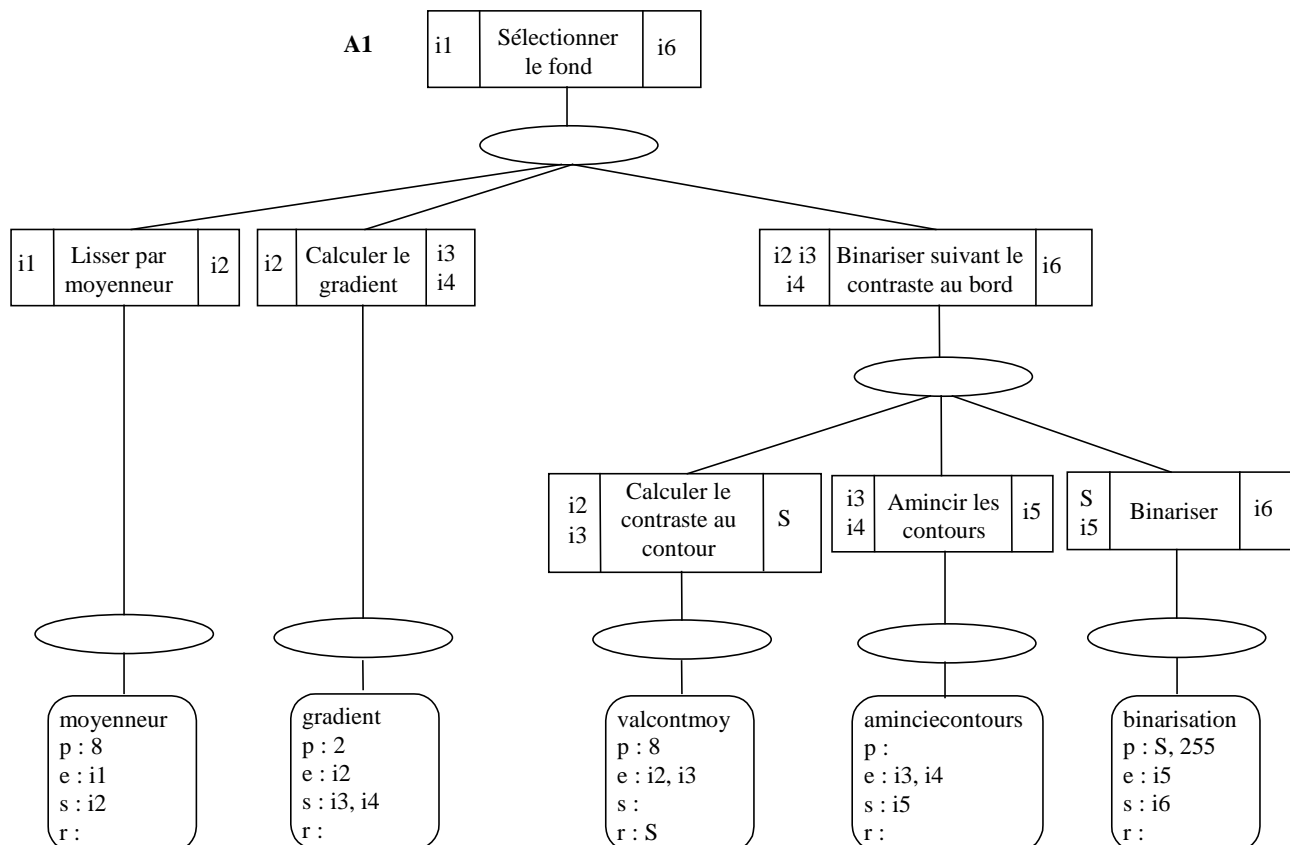
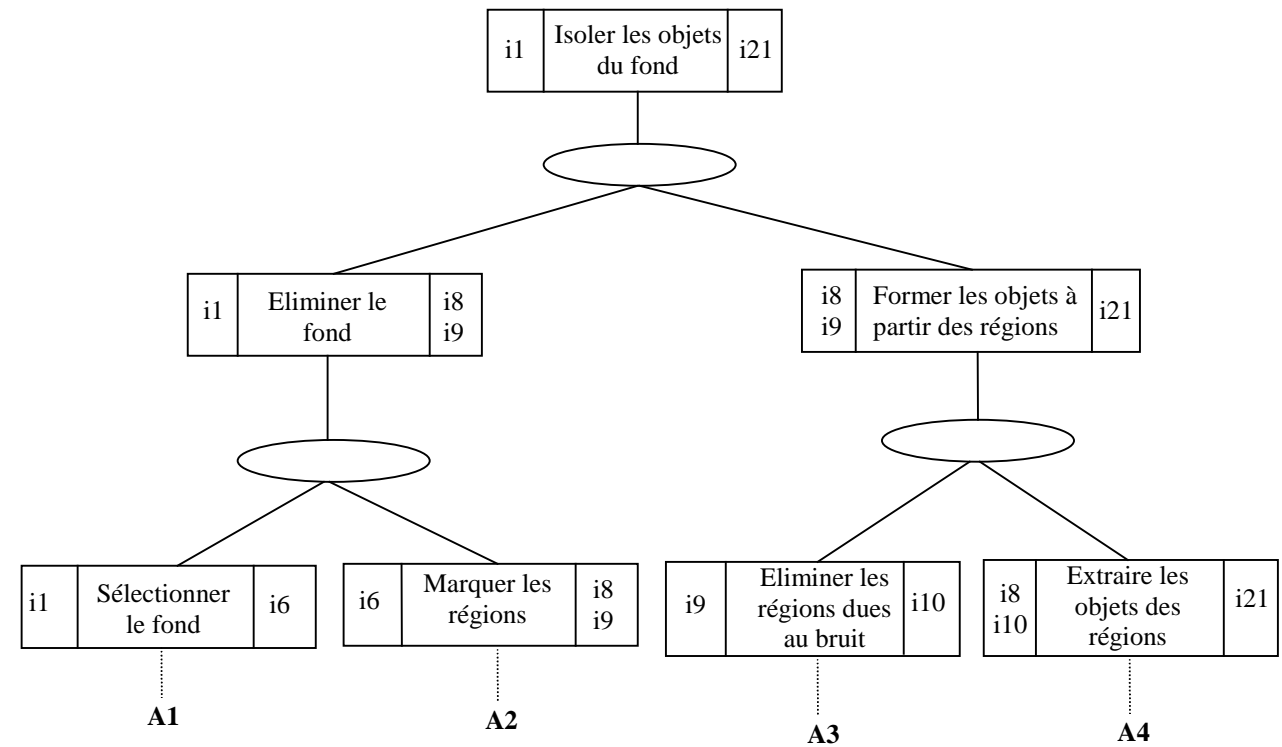
Ce plan a été construit pour s'appliquer à des images de cytologie. Il a pour but la détection des noyaux de cellules présents dans une image. Le but final de cette application est de calculer le taux d'ADN des cellules épithéliales, ce taux permettant de détecter la prolifération anarchique des cellules (cellules cancéreuses). Les différents objets présents dans l'image sont décrits comme suit :

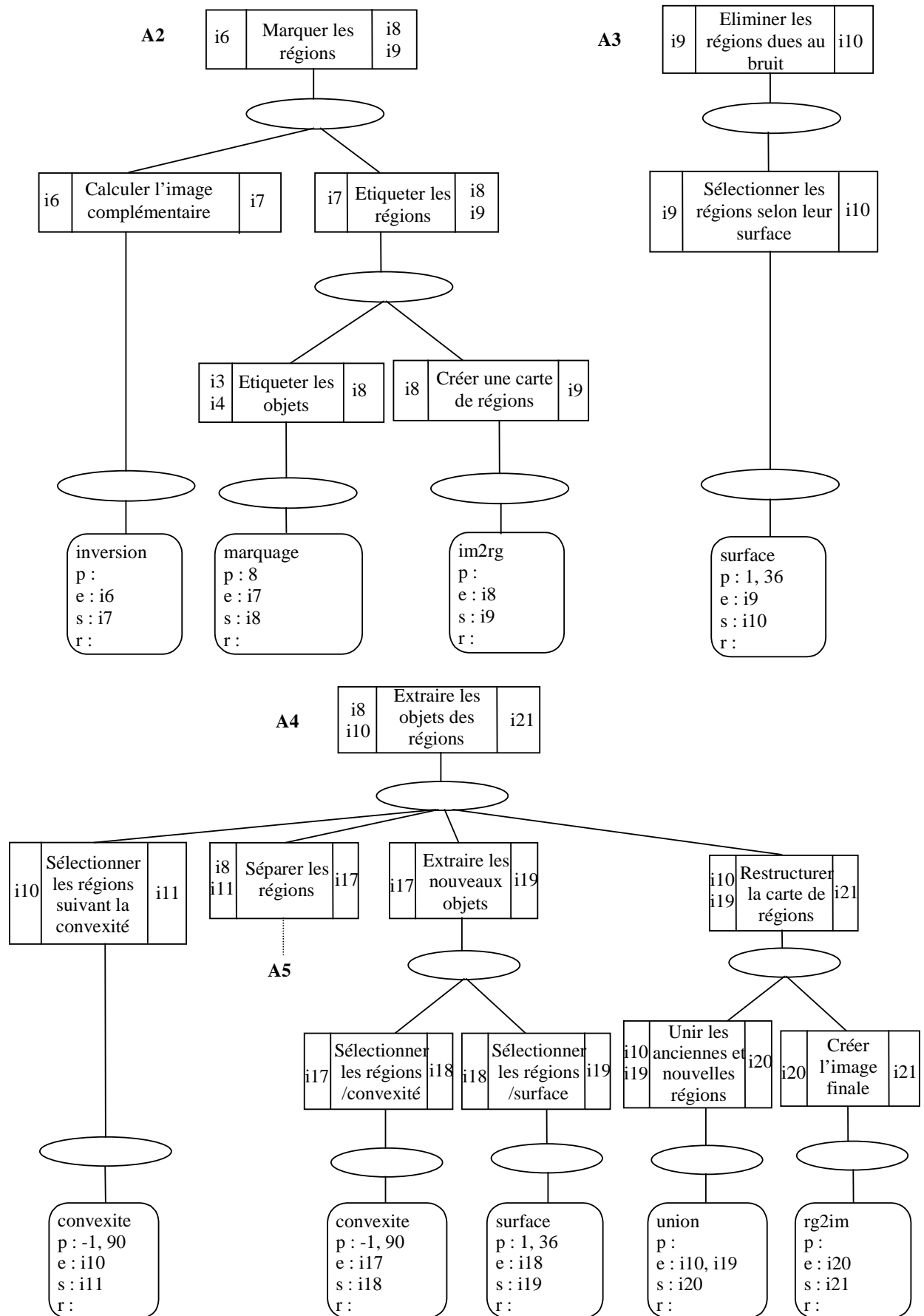
- le fond : texture homogène d'un niveau de gris plus clair que le reste de l'image,
- amas de cellules : de plus grande taille que les autres objets et de forme non convexe,
- cellule épithéliale : objet légèrement texturé, de faible contraste intérieur, de forme convexe et ovale et de taille moyenne,
- lymphocyte : objet non texturé, de forme convexe et ronde, plus foncé que les cellules épithéliales,
- cellule stromale : objet non texturé, de forme ovale allongée et de taille moyenne,
- débris : objet dissymétrique et de petite taille.

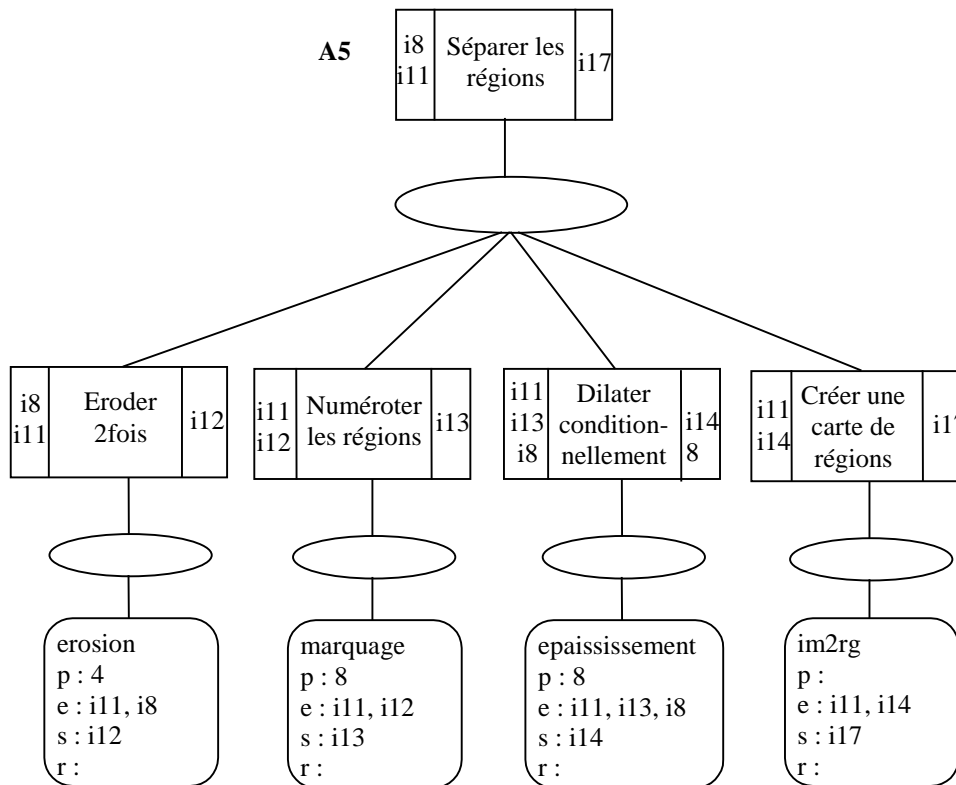
Le plan élimine tout d'abord le fond de l'image avant de détecter les différents objets présents en se basant sur les caractéristiques des cellules épithéliales (par exemple, on ne cherche pas à dissocier les amas car ils seront éliminés par la suite).

Il retourne une carte de régions où l'on peut distinguer les différents objets par la couleur qui leur est associée. A partir de cette carte de régions, on pourra éliminer les objets qui ne correspondent pas à des cellules épithéliales en fonction de leur taille, de leur forme ou de leur niveau de gris pour fournir au biologiste une image qui ne comporte que les cellules qui l'intéressent.









Le plan A est le premier à avoir été intégré dans notre système. Il a permis de valider l'architecture TMO en montrant que :

- la décomposition en tâches à plusieurs niveaux d'abstraction fournissait une bonne représentation des stratégies mises en œuvre et un bon support pour le dialogue entre experts,
- le plan modélisé était directement opérationnel.

Nous avons également pu vérifier le fonctionnement des différents modes d'exécution des opérateurs :

- l'opérateur « binarisation » (sous-plan A1) est exécuté en mode « optimisation » : plusieurs valeurs du seuil bas sont testées (entre S-3 et S+3), est conservée celle qui préserve le mieux les contours,
- l'opérateur « érosion » (sous-plan A5) est exécuté en mode « pour » : deux exécutions successives.

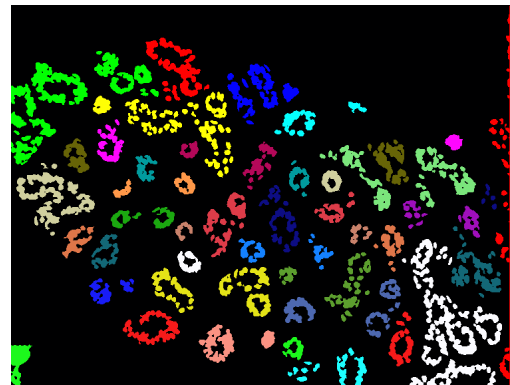
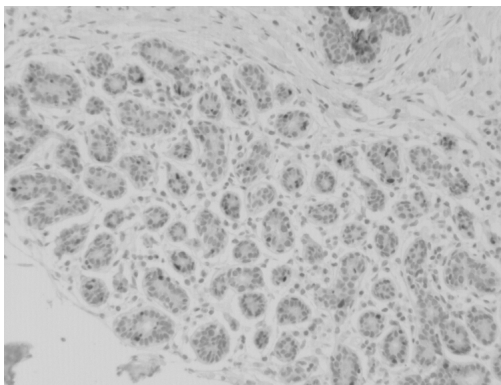
Il a été intégré tout d'abord dans une maquette du système TMO dépourvue d'interface graphique et mise au point au cours d'un stage de DEA [Ficet 95]. Son intégration dans le système actuel a montré les avantages apportés par l'interface utilisateur, tant au niveau de la convivialité qu'au niveau du gain de temps.

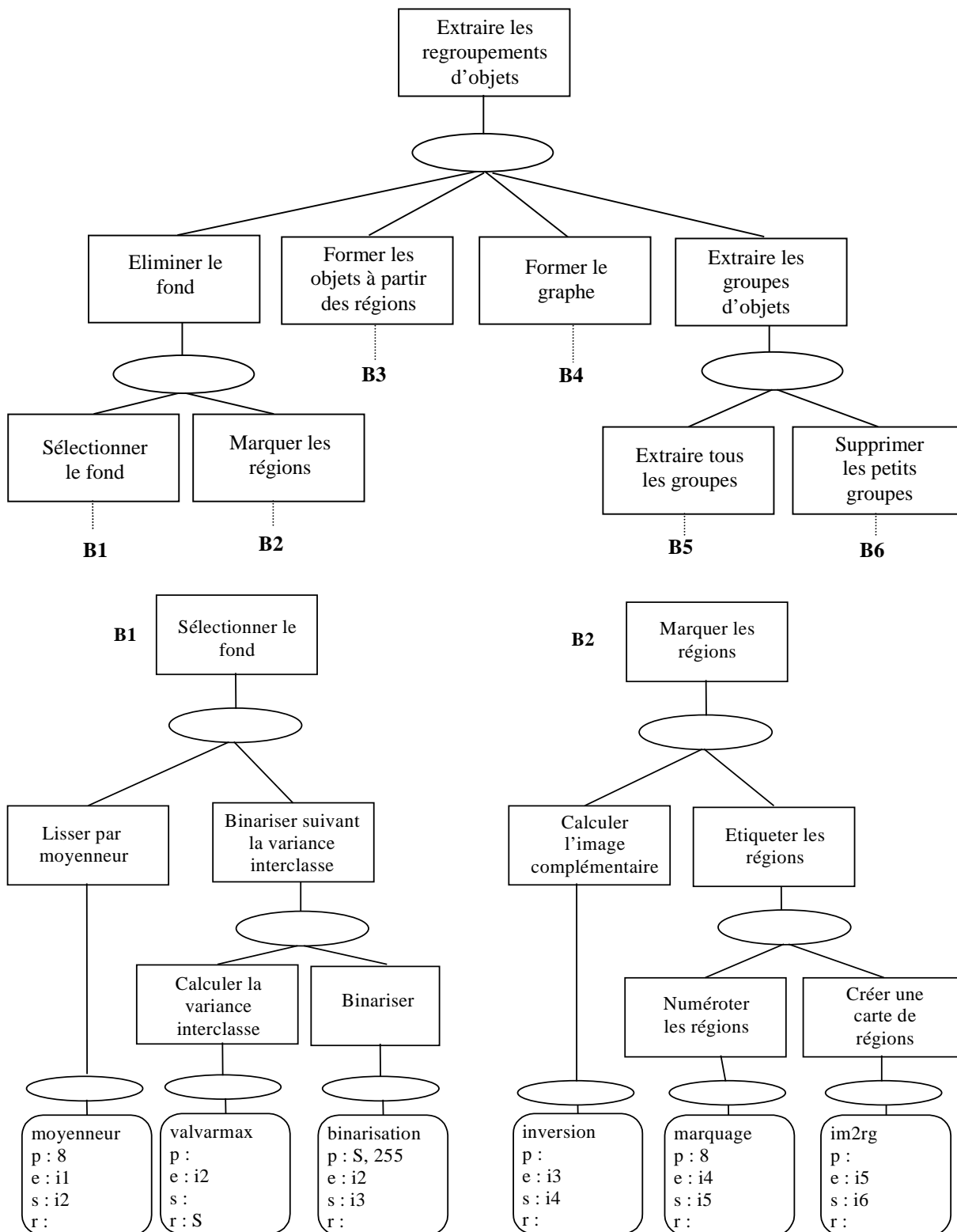
En outre, il nous a permis de définir plus précisément certaines fonctionnalités concernant la création et l'exécution des plans. Son intégration a également mis en évidence le besoin d'une validation syntaxique au fur et à mesure de la modélisation : en effet, les problèmes dûs à l'absence de vérification de la cohérence syntaxique, n'apparaissaient qu'au moment de l'exécution, et il était donc difficile alors d'en déterminer les causes.

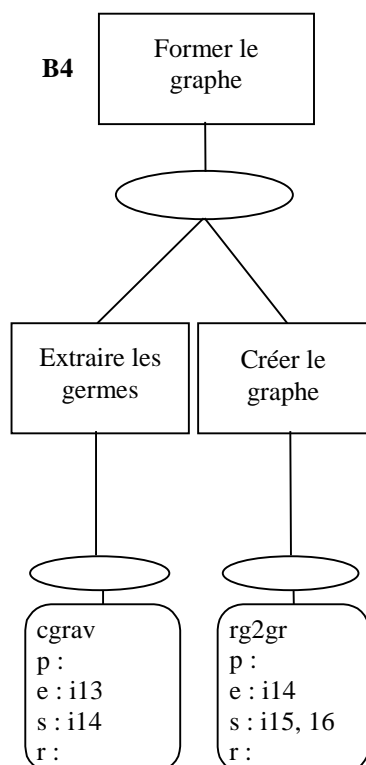
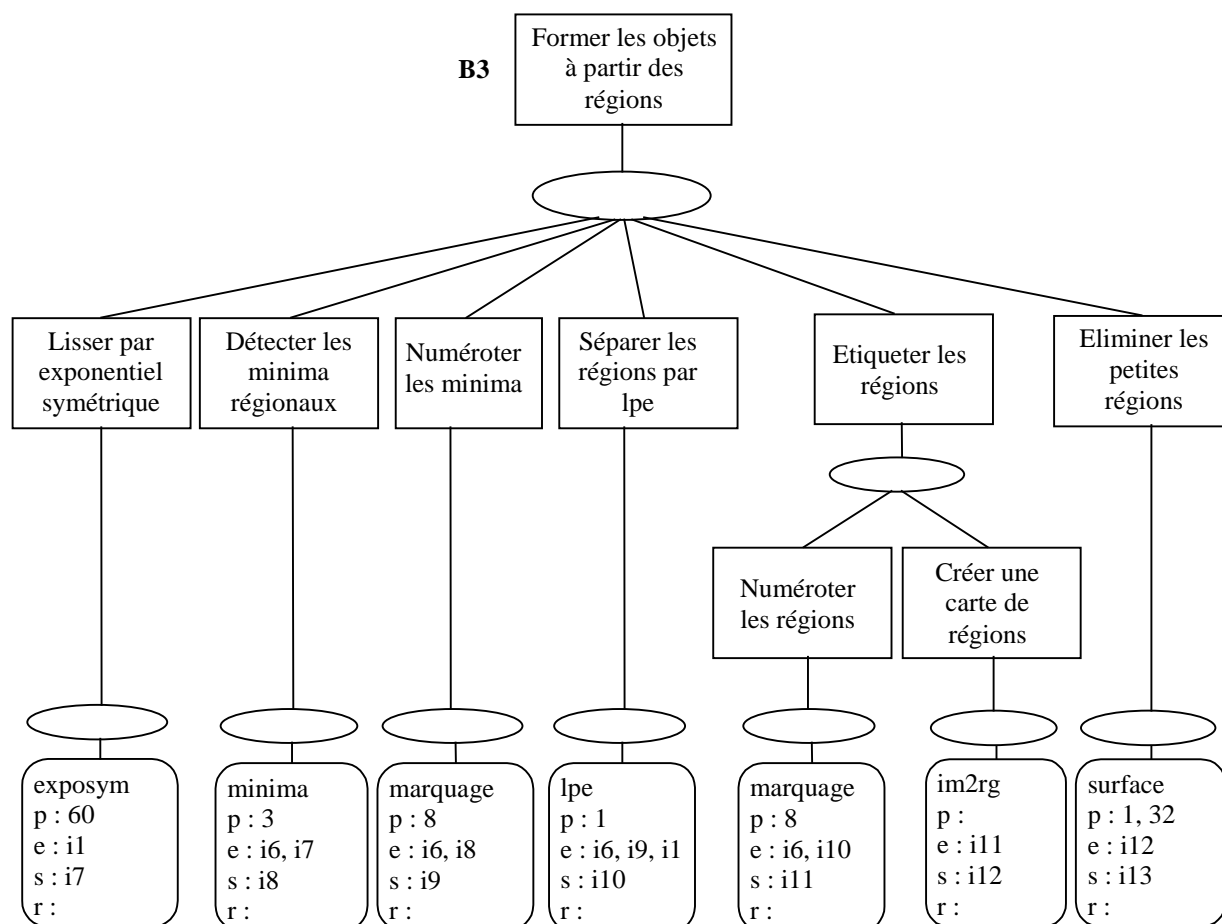
Plan B : Extraire les regroupements d'objets

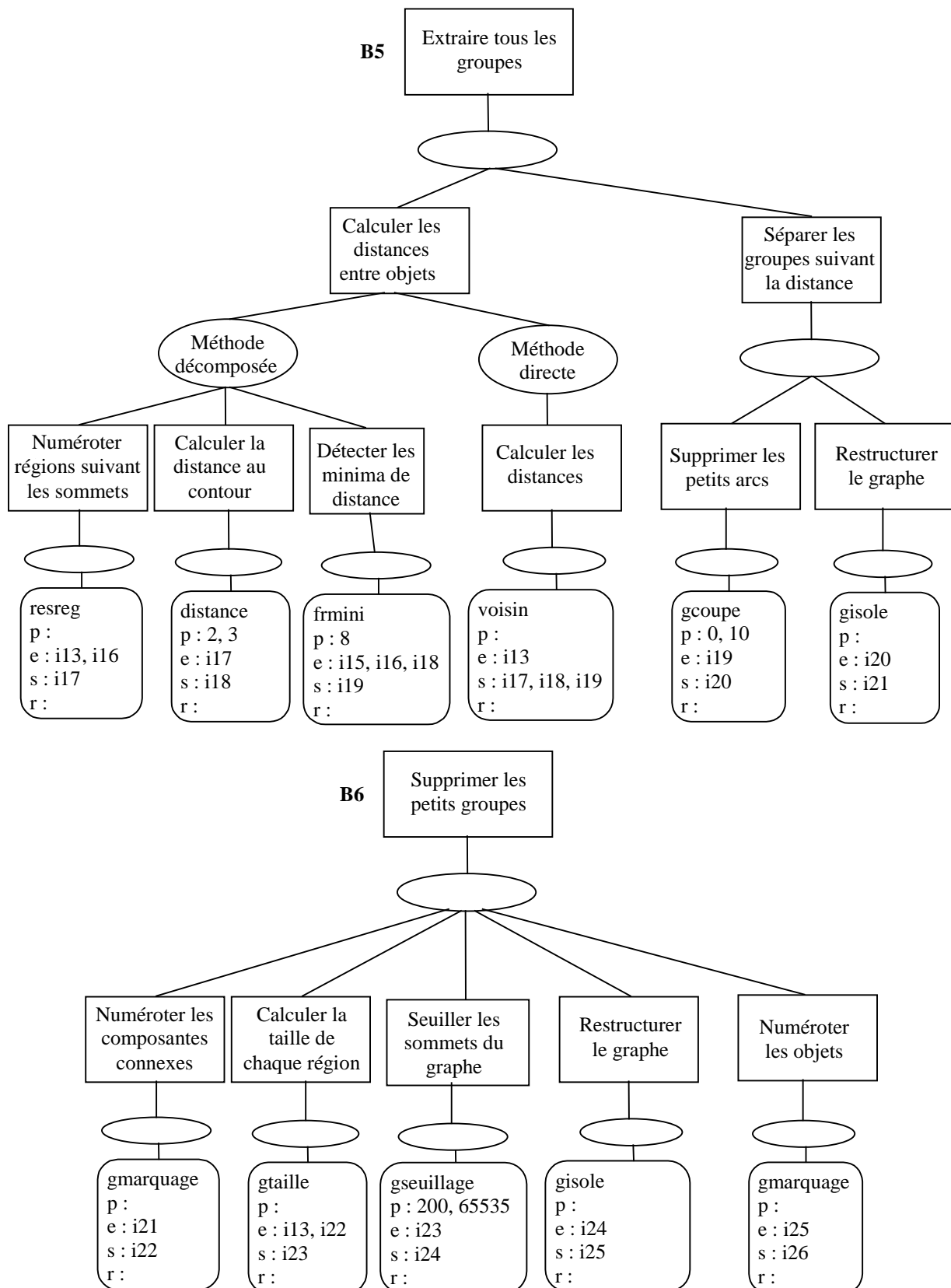
Le plan B a été créé pour s'appliquer à des images d'histologie. Son but est de mettre en évidence les groupements significatifs de noyaux, l'étude de ces groupements permettant de déterminer la présence de lobules tumoraux. Les noyaux sont des objets sombres sur un fond clair et homogène. Les regroupements comportent des noyaux de tailles voisines, ils peuvent être de forme grossièrement elliptique ou linéaire et comporter un nombre très variable de noyaux. Après avoir éliminé le fond de l'image, on sélectionne les objets correspondant à des noyaux puis on détermine les regroupements à l'aide de techniques basées sur les graphes en utilisant les propriétés de voisinage des noyaux.

Ce plan retourne une carte de régions où les groupements d'objets sont étiquetés par des couleurs distinctes.









Ce deuxième plan étant relativement complexe, il nous a permis de compléter et de valider les fonctions de vérification de la cohérence syntaxique d'un plan.

La dernière partie de ce plan (« extraire les groupes d'objets ») fait intervenir des critères sur les objets présents mais aussi sur les relations entre ces objets et constitue en cela un premier pas vers la reconnaissance de formes.

L'image d'origine de ce plan est du même type que celle du plan A (même domaine, même capteur). La modélisation de ces deux applications sous forme de plans TMO a fait apparaître l'utilisation de techniques différentes dans la première phase correspondant à la suppression du fond (utilisation de la valeur du contraste au contour dans A et de la variance interclasse dans B). Ces techniques semblant toutes les deux intéressantes pour le plan A, nous avons ajouté une deuxième méthode à la tâche « sélectionner le fond » de celui-ci (fig. 96), cette nouvelle méthode correspondant à la technique utilisée dans le plan B. Ceci montre le premier intérêt du choix entre deux méthodes : la possibilité de tester différentes techniques dans une même application. Le deuxième cas dans lequel on peut modéliser plusieurs méthodes pour une même tâche est illustré par la tâche du plan B « calculer les distances entre objets » : les techniques utilisées sont les mêmes, mais avec la première méthode, la tâche est décomposée en plusieurs sous-tâches permettant ainsi d'accéder aux images intermédiaires et de régler les paramètres d'exécution, alors qu'avec la deuxième méthode, l'exécution est plus rapide mais les paramètres ne peuvent pas être ajustés. En particulier, dans la première méthode, les paramètres de l'opérateur « distance » permettent de spécifier sa propre métrique (« 2, 3 » correspond à une approximation de la distance euclidienne par le chanfrein), alors que dans la deuxième, on utilisera obligatoirement la distance euclidienne.

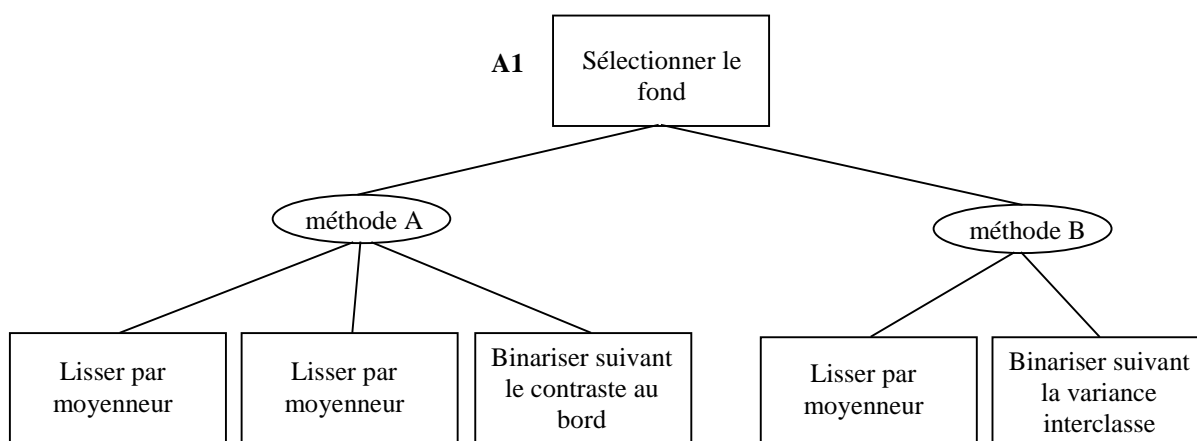
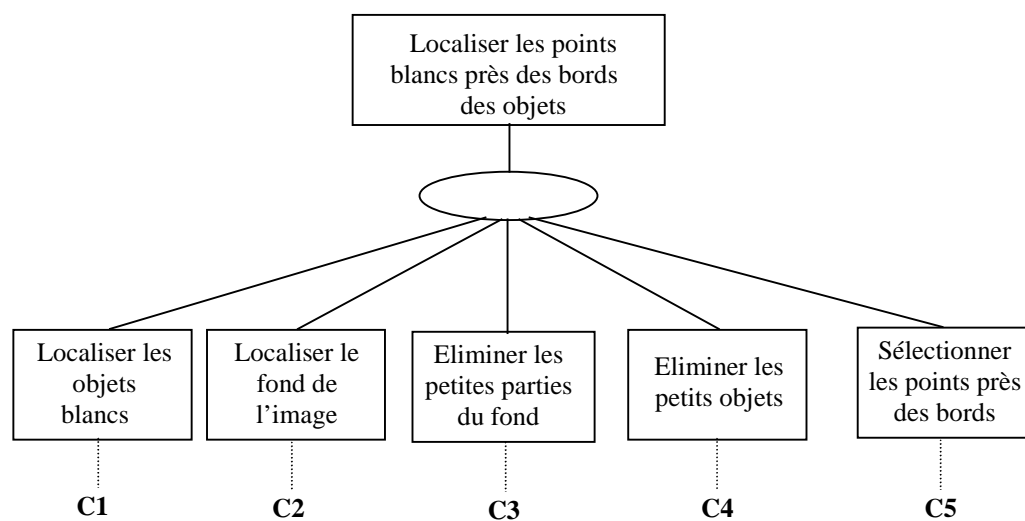
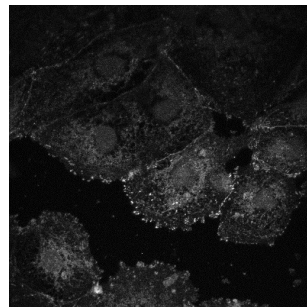


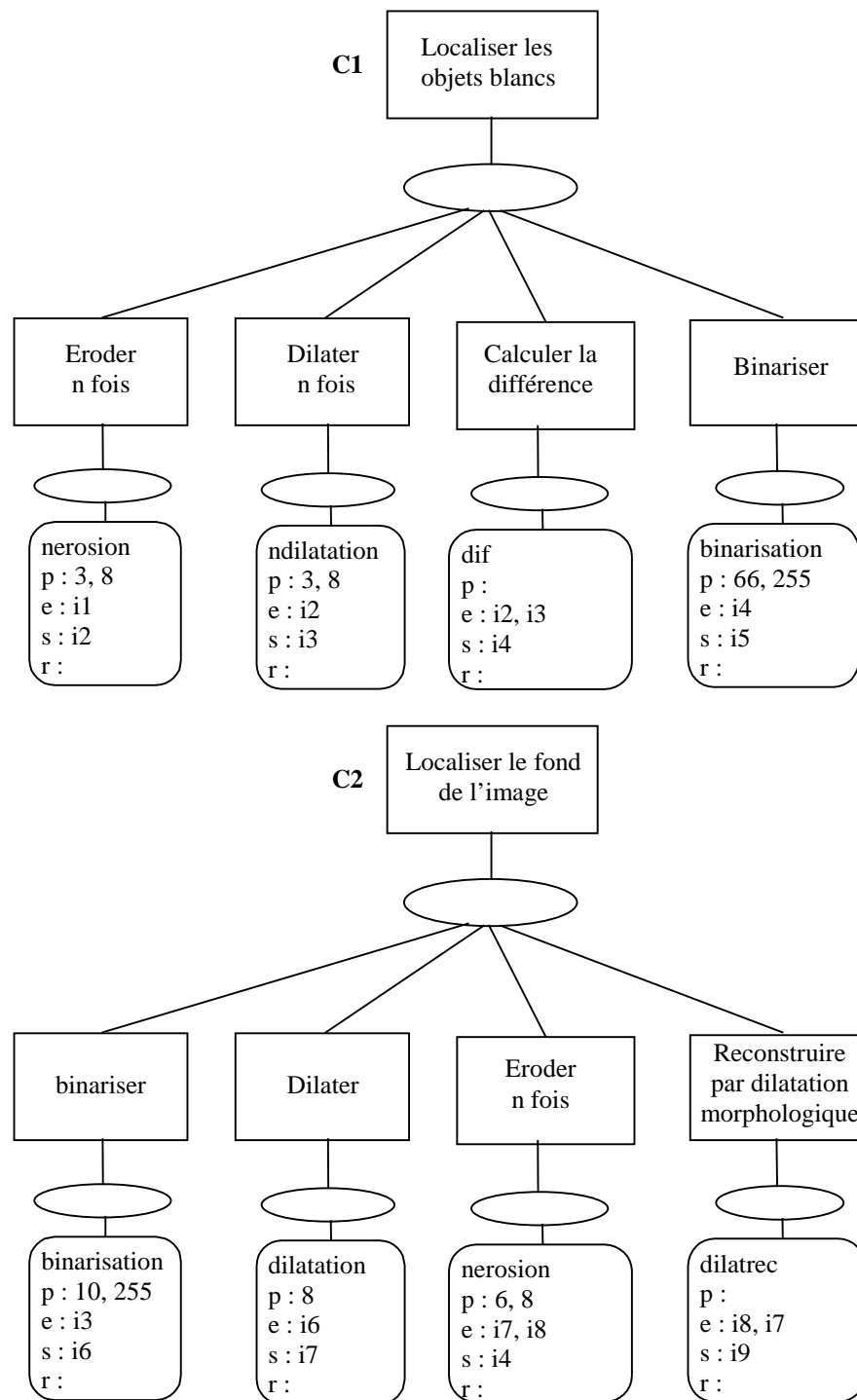
fig.96 : intégration d'une deuxième méthode dans le plan A1

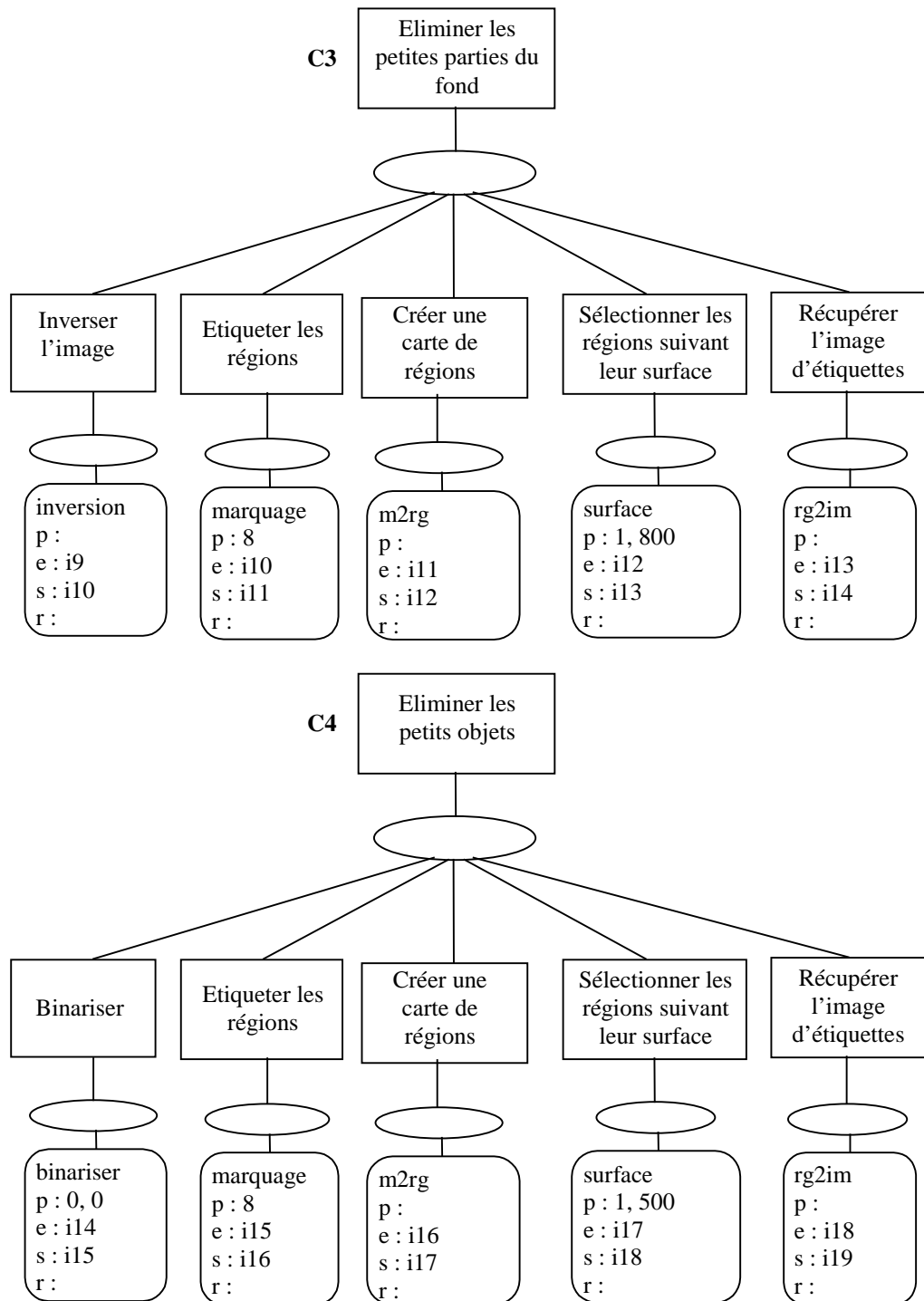
Ces tests ont également fait apparaître le besoin d'une nouvelle fonctionnalité : la recopie d'élément (tâche, outil ou partie entière de plan). En effet, le sous-plan ajouté au plan A étant déjà défini dans le plan B, sa redéfinition élément par élément s'est avérée fastidieuse.

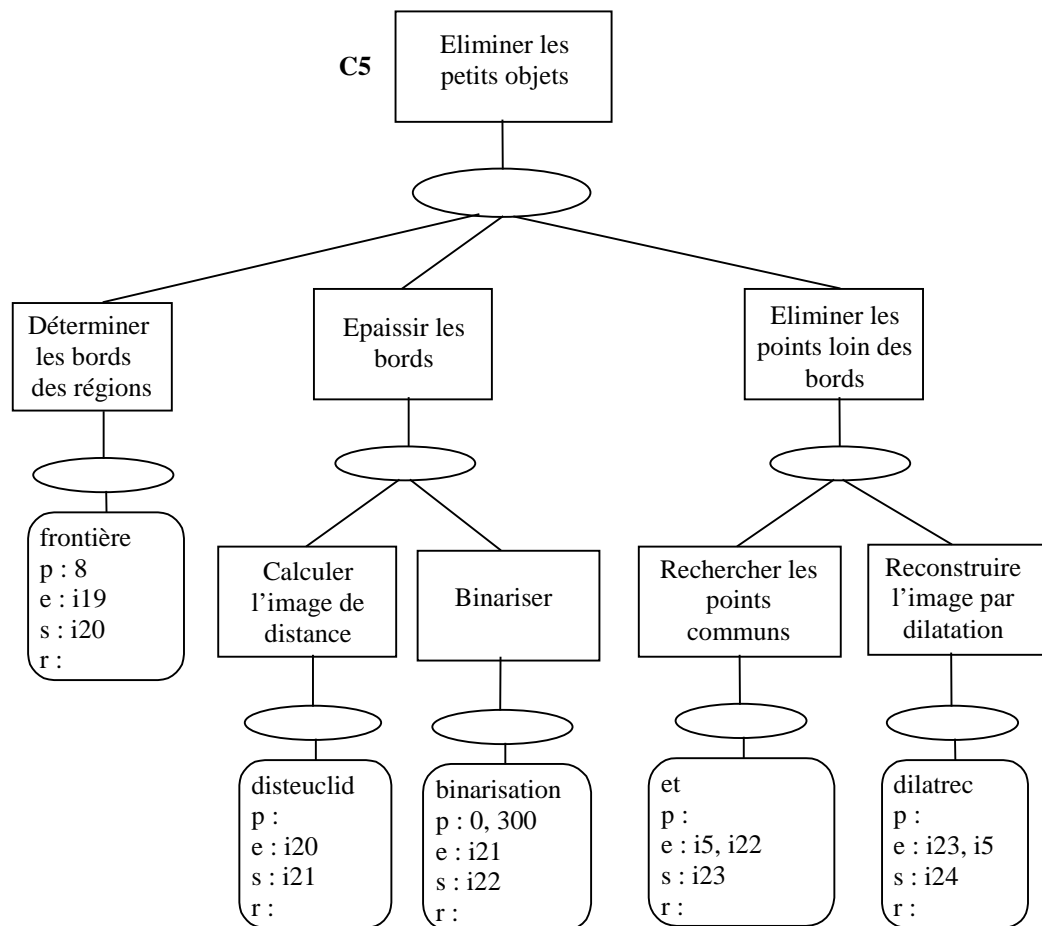
Plan C : Localiser les points brillants près des bords des objets

Ce plan a été créé pour traiter des images de cytologie dans lesquelles on cherche à quantifier la présence de contacts focaux sur les bords de cellules de culture. Ces mesures sont utilisées dans une étude réalisée en cancérologie expérimentale au centre F. Baclesse. Les cellules sont définies comme des objets légèrement plus clairs et plus inhomogènes que le fond et les contacts focaux comme des points blancs se trouvant près des bords des cellules. Le plan recherche donc séparément les deux types d'objets (cellules et contacts focaux) avant de prendre en compte la relation qui les lie. Il fournit une image binaire dans laquelle les points localisés apparaissent en blanc.







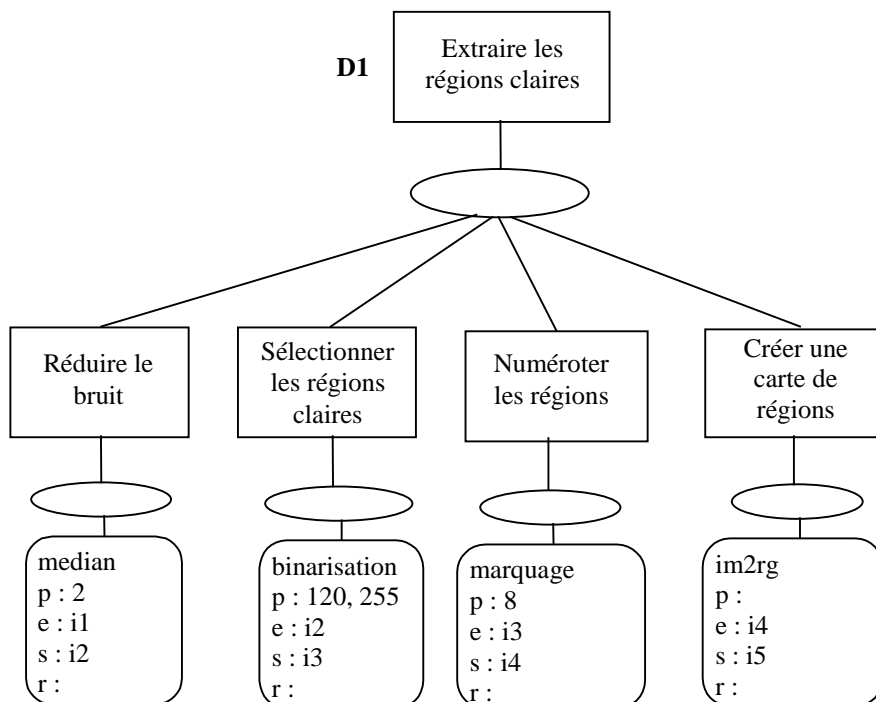
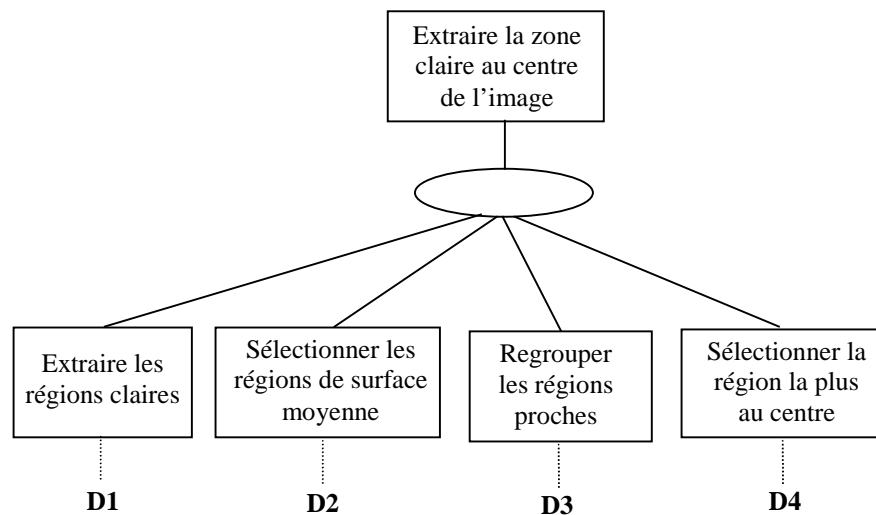


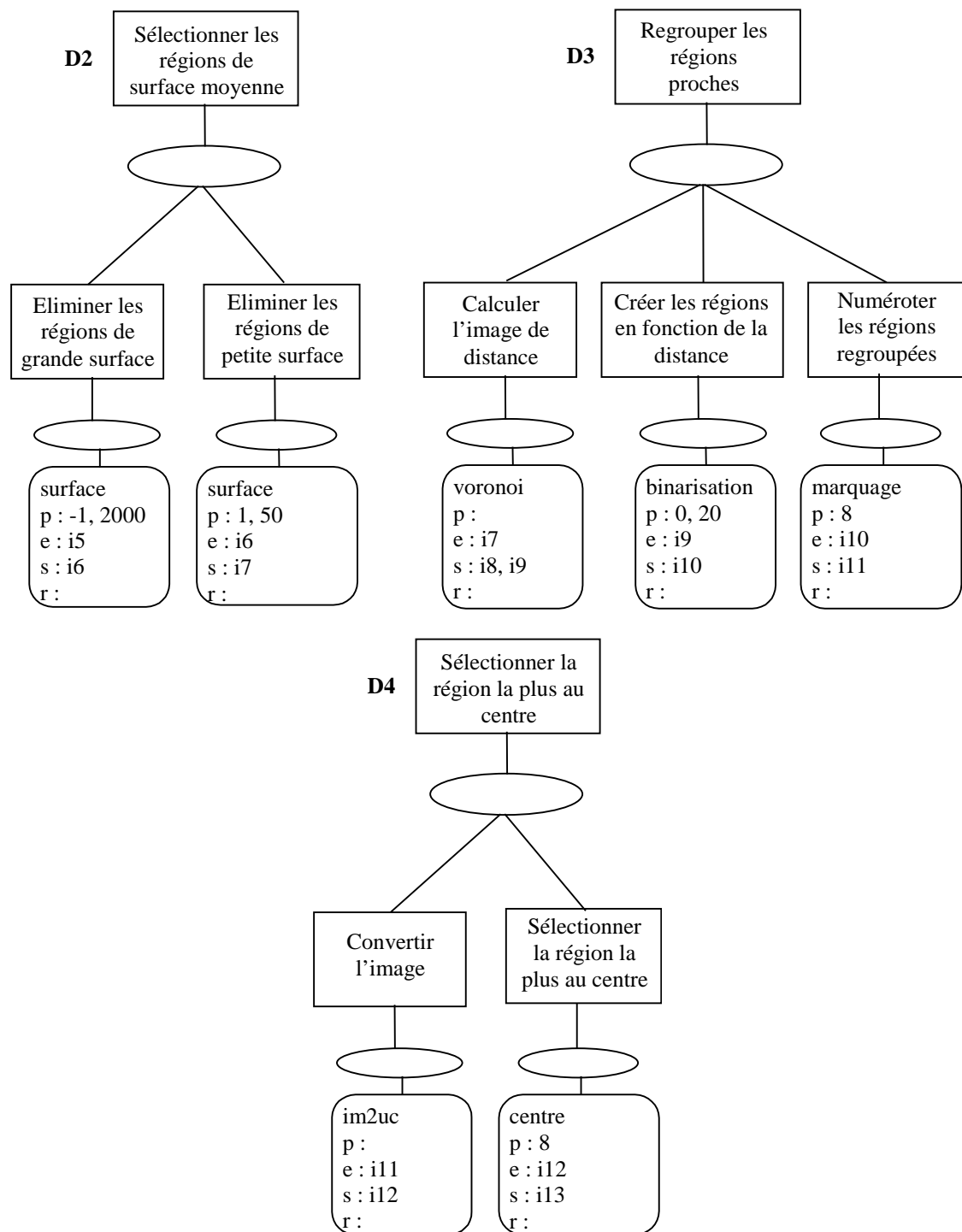
Le plan C est un exemple représentatif de l'utilisation d'un masque comme focus d'attention. Ce plan réalise en fait deux segmentations différentes (une pour la détection des points brillants et une pour la détection des cellules) avant de sélectionner les objets du premier résultat (*les points brillants*) qui possèdent une certaine relation (*près des bords de*) avec les objets du deuxième résultat (*les cellules*). On utilise donc une zone représentant le critère « près des bords des cellules » définie par un masque comme focus d'attention pour sélectionner les points recherchés.

Plan D : Extraire la zone claire au centre de l'image

Le plan D a été créé pour s'appliquer sur des images de visage. Il a été mis au point par J.P. Bastard dans le cadre d'un projet de 3^{ème} année d'école d'ingénieur, c'est à dire par un utilisateur novice du système et débutant en TI. Le problème initial a été posé par P. Deléglise de l'Université du Mans dans le cadre de l'action 10.2 du GDR-PRC ISIS. Ce problème consiste à repérer les points extrêmes intérieurs des lèvres. L'utilisateur a mis au point trois versions différentes (Da, Db, Dc) pour réaliser ce traitement. Nous présentons ici le premier de ces trois plans (Da) qui réalise la première phase du travail concernant l'extraction de la région correspondant aux dents et ne

s'applique qu'à des images de bouches ouvertes. La zone à extraire est définie comme une zone claire située au centre de l'image.





Ce plan a été intégré par un utilisateur novice, ce qui nous a tout d'abord permis de noter la facilité de prise en main du système TMO : même s'il n'est pas toujours aisé de nommer les tâches de haut niveau, les principes de modélisation lui sont apparus clairs et faciles à utiliser.

Ce travail nous ont permis de soulever de nouveaux problèmes concernant :

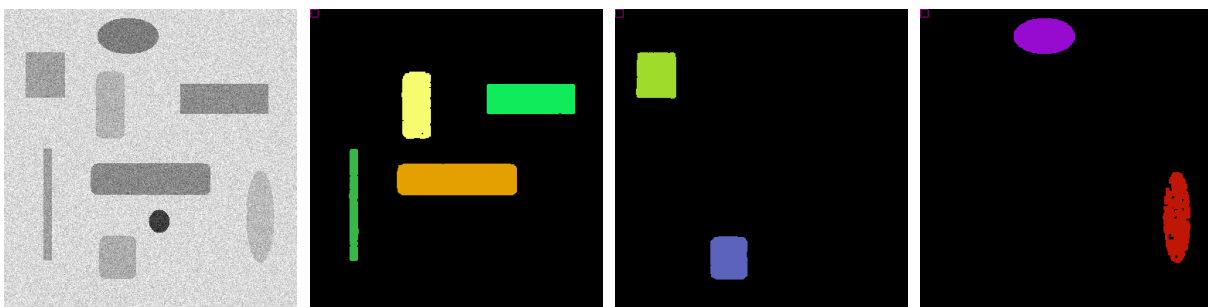
- la validation des connaissances intégrées : il est apparu le besoin de vérifier au maximum les erreurs de saisie de l'utilisateur qui peuvent avoir des conséquences sur l'exécution du plan,
- la communication système/utilisateur : certains modules de communication ont dû être redéfinis ou précisés, en particulier lorsque le vocabulaire utilisé était inadéquat.

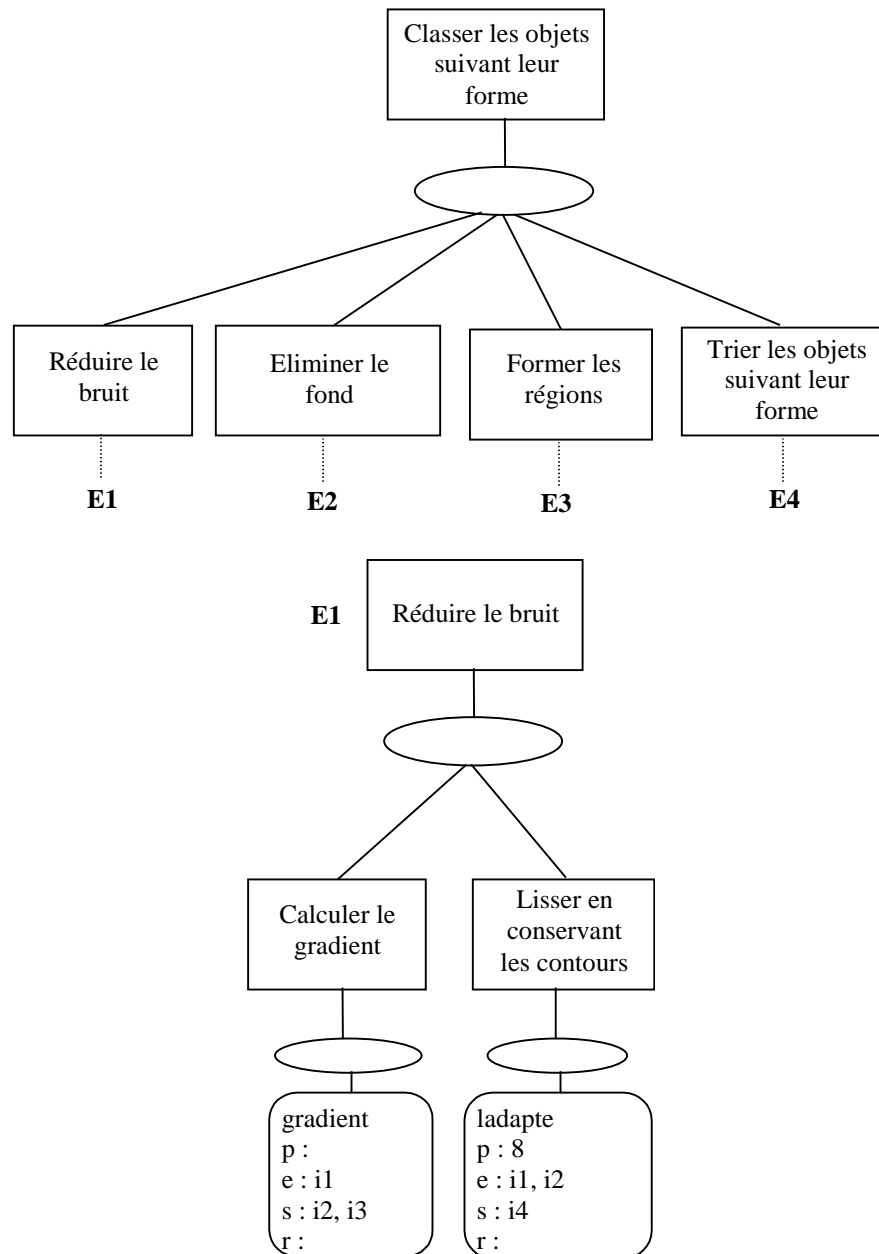
L'intégration de ce plan a également montré que le système n'était pas limité aux opérateurs de la bibliothèque (même si cette bibliothèque est relativement complète), puisque l'outil « sélectionner la région la plus au centre » est réalisé par une fonction C définie à cette occasion.

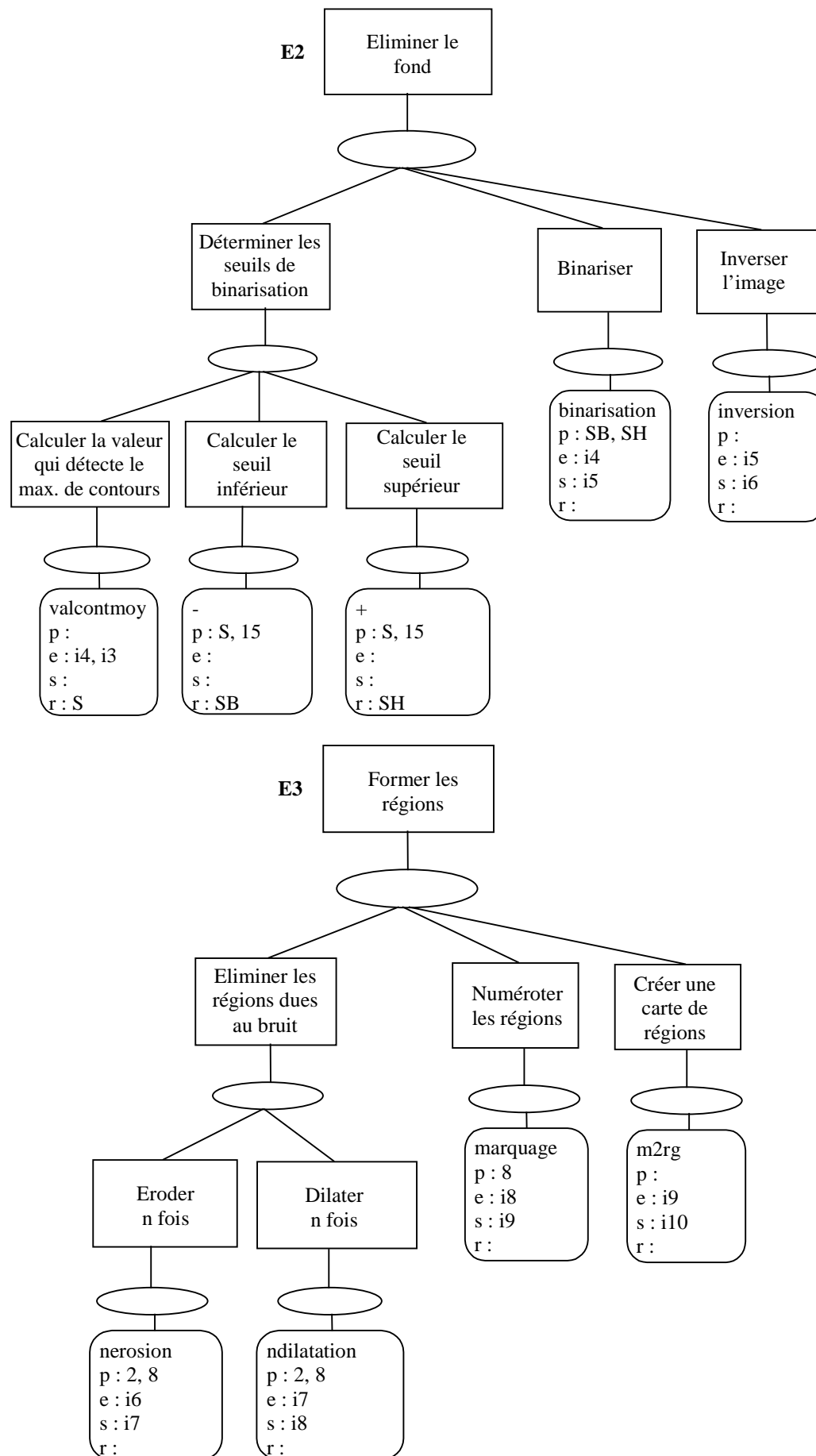
Pour se familiariser avec la bibliothèque Pandore, l'utilisateur a d'abord mis au point son application à l'aide d'un script. La modélisation de son application sous forme d'un plan TMO lui a permis de mettre à jour la stratégie utilisée et a joué en cela un rôle pédagogique. En effet, si le premier plan (Da) a été construit en utilisant une démarche ascendante (par regroupement d'opérateurs), les plans suivants (Db et Dc) ont été mis au point de façon descendante (par décomposition d'un problème en sous-problèmes) en se basant sur la stratégie découverte dans le premier.

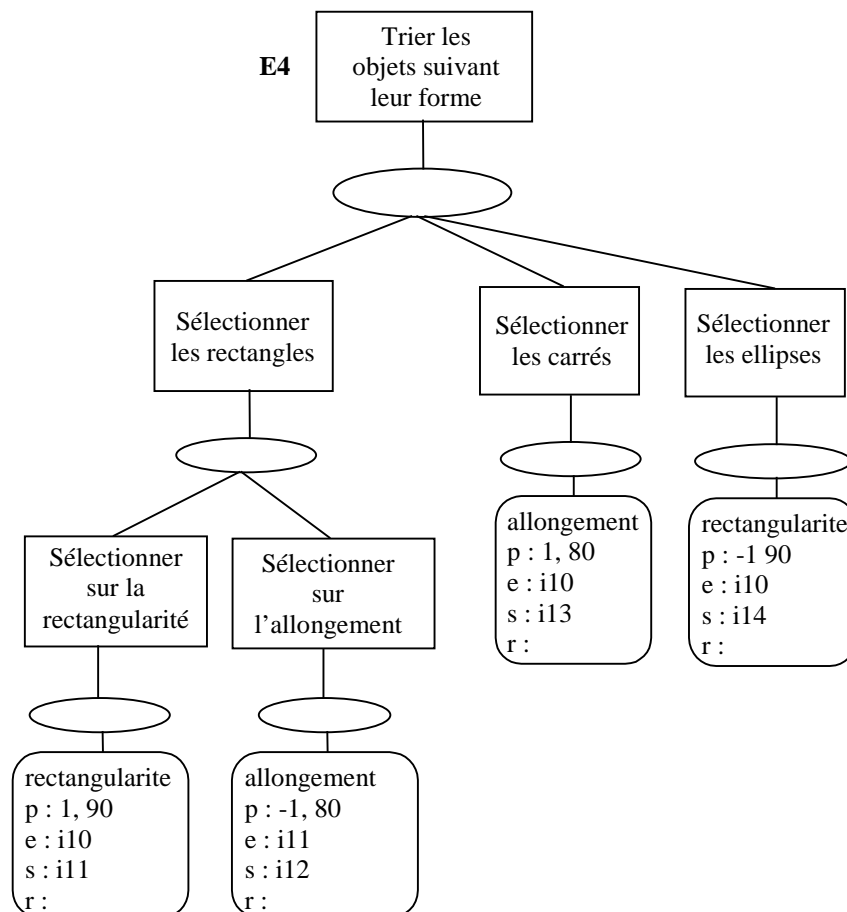
Plan E : Classer les objets suivants leur forme

Le plan E a été réalisé dans le cadre des tests du système TMO et des opérateurs de sélection. Il s'applique à des images de synthèse dans lequel on cherche à trier les objets suivant leur forme géométrique. Il fournit trois résultats : l'image des rectangles, l'image des carrés et l'image des ellipses.







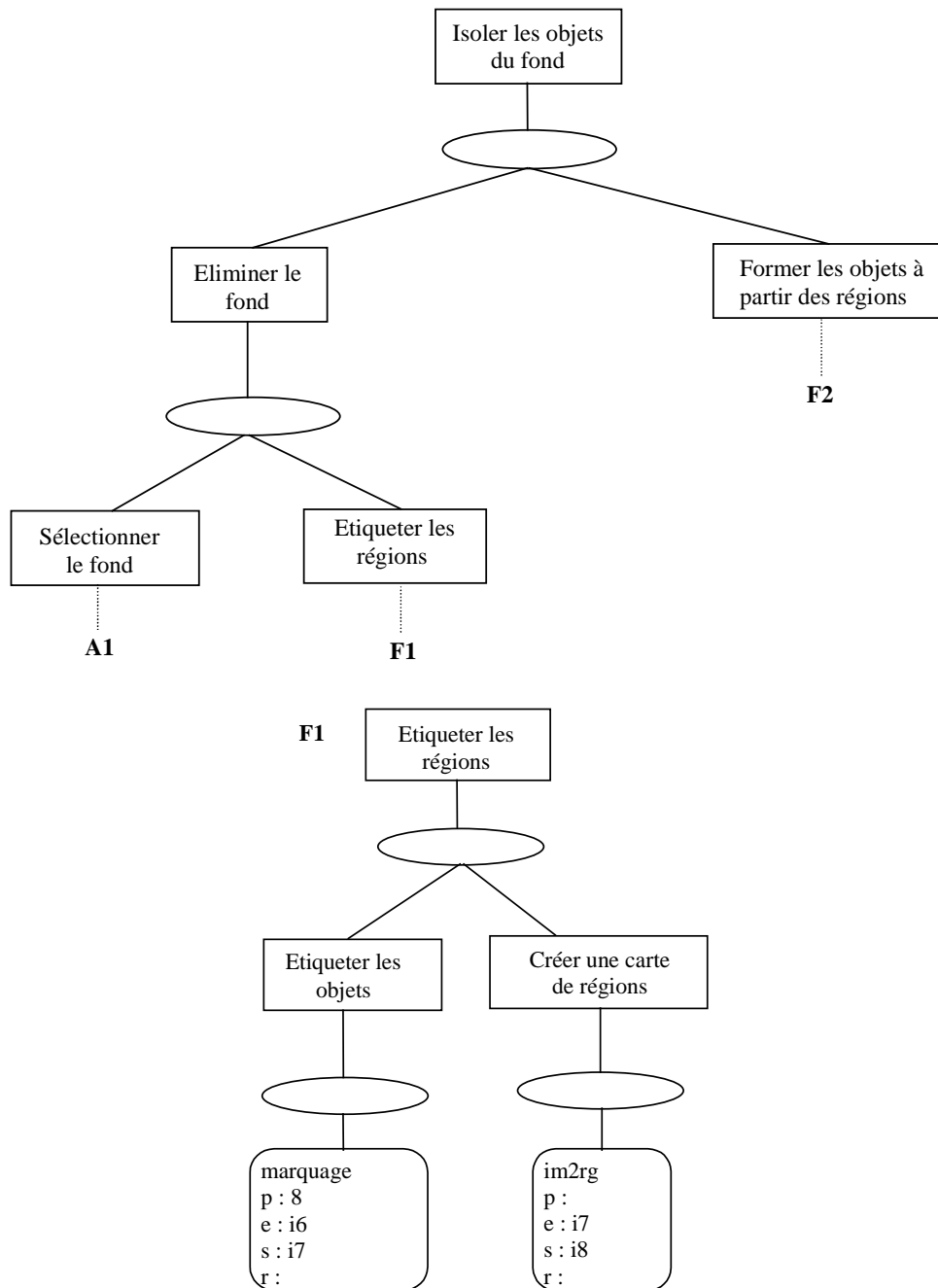


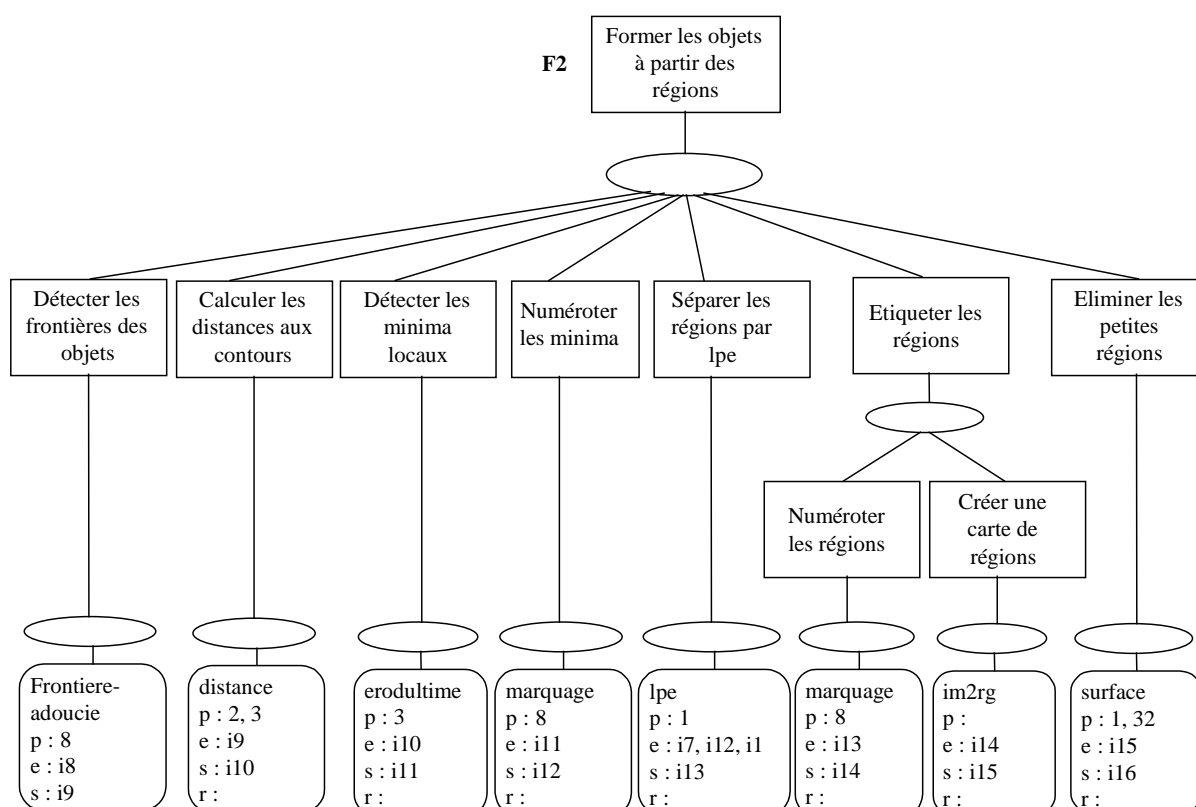
Ce plan a permis de tester l'imbrication d'outils de plusieurs types : on enchaîne ici des outils faisant appel à des opérateurs Pandore et des outils faisant appel à des fonctions Lisp. Cela montre qu'une programmation au niveau connaissance permet aisément la réutilisation d'éléments de programmation de différents langages.

Par ailleurs, ce plan réalise un nouveau type de traitement (détection de forme particulière) et s'exécute sur un nouveau type d'image. Il nous a ainsi permis d'enrichir la base de cas avec des cas relatifs à des tâches de reconnaissance d'une forme particulière et d'accroître l'ensemble des termes de description proposés.

Plan F : Isoler les objets du fond (2^{ème} stratégie)

Le plan F a été réalisé en utilisant le module de RàPC (voir § V.4). Il cherche à isoler et à séparer les objets présents dans une image industrielle. Il fournit une carte de régions où l'on distingue les objets suivant leur couleur.





Ce plan est le premier mis au point à l'aide du module de RàPC. Il a montré la pertinence des critères de sélection et l'efficacité de l'algorithme de sélection/adaptation : l'interactivité et la récursivité de cet algorithme permettent d'obtenir rapidement une solution satisfaisante. Cependant le nombre d'adaptations locales ayant dû être réalisées révèle la pauvreté de la base de cas qui nécessite donc d'être enrichie à l'aide de plans réalisant des traitements plus variés.

Bilan sur les expérimentations

Ces expérimentations ont été réalisées au fur et à mesure de la conception du système. Chaque fonctionnalité a ainsi mis en évidence dès que possible sa validation ou ses faiblesses, permettant alors d'étudier rapidement ces dernières, de façon à les contrer.

La poursuite de ces expérimentations devra comprendre l'intégration d'un grand nombre d'applications réalisant de nouveaux types de traitement et s'appliquant à des domaines divers, ceci dans le but d'enrichir le vocabulaire proposé pour la description des cas et d'augmenter les possibilités du module de RàPC. Notons en particulier, que l'enrichissement actuel de la bibliothèque Pandore avec des opérateurs d'interprétation d'images, nous permettra d'élargir le champs de nos applications.

Enfin, il sera indispensable de tester ce système dans des « conditions réelles », c'est à dire de le faire valider par un expert en TI.

Conclusion

Nous avons réalisé un système interactif destiné à aider un expert en TI à modéliser ses connaissances, dans un premier temps pour les capitaliser, et ensuite pour les réutiliser.

Grâce à l'interactivité du système, l'acquisition des connaissances s'effectue directement auprès de l'expert et ainsi l'intégration progressive des connaissances est réalisée au fur et à mesure de leur apparition dans de nouvelles applications.

D'une part, le modèle proposé « tâche - méthode - outil » est facilement compréhensible par l'utilisateur et est directement opérationnalisable, ce qui facilite la coopération système/utilisateur. D'autre part, la modélisation sous forme de plan hiérarchique permet de représenter toutes les étapes de raisonnement de l'expert et d'analyser celui-ci à plusieurs niveaux d'abstraction. Par ailleurs, la représentation des connaissances de contrôle et des connaissances du domaine à l'aide du même modèle donne un caractère évolutif au système proposé en facilitant la définition de nouvelles fonctionnalités. La présentation des différentes fonctionnalités à l'aide d'une interface graphique conviviale améliore la coopération entre le système et l'utilisateur. Ce dernier peut créer ses applications de façon incrémentale en alternant le développement et les tests des différentes parties. La visualisation des applications sous forme d'arbre de tâches et l'accès aux informations sur chacune de ces tâches fournit une vision de l'application à plusieurs niveaux et facilite la coopération entre l'expert en TI et l'expert du domaine de l'image.

L'intégration d'un module de RàPC a permis d'augmenter considérablement l'aide apportée par le système. A l'aide de celui-ci, l'expert en TI peut rechercher un plan existant qui résout un problème proche du sien et l'adapter à sa nouvelle situation. Il peut ainsi réutiliser ses propres connaissances ou celles qu'un autre expert en TI aura modélisées au préalable, réalisant alors un « partage des connaissances ».

L'algorithme récursif de RàPC mis en œuvre alterne les phases de remémoration et d'adaptation, permettant ainsi de concevoir un plan par assemblage de parties d'autres plans. Les critères de sélection d'un cas concernent la définition des traitements réalisés et la description des images sur lesquelles sont réalisés ces traitements. Les tests que nous avons effectués ont montré la pertinence du choix de ces critères.

Pour restreindre l'étendue du problème, les tests ont été réalisés sur des applications de segmentation d'images. Il sera intéressant de diversifier le contenu de nos bases (base de plans et base de cas) en intégrant des applications réalisant des traitements plus diversifiés (allant de la restauration à l'interprétation) et s'appliquant à des images de différents domaines. Ceci nous

permettra également d'enrichir le vocabulaire utilisé pour la description des cas et ainsi de compléter notre ensemble de critères et de proposer à l'utilisateur des listes de termes plus exhaustives.

Plusieurs voies sont envisagées pour les perspectives de ce travail.

Pour améliorer l'efficacité de notre architecture, il serait intéressant d'étendre aux tâches la notion de mode d'exécution utilisée dans les outils : on pourrait ainsi exécuter une tâche soit une seule fois comme c'est le cas actuellement, soit dans une boucle itérative ou conditionnelle.

Il serait également intéressant de réduire la tâche de l'utilisateur lors de l'utilisation du module de RàPC, en particulier dans la phase d'adaptation. L'utilisation de règles basées sur la comparaison des valeurs de certains critères permettrait de guider l'utilisateur en lui indiquant les parties de plans nécessitant une adaptation. Par exemple, si le type de bruit est différent, il faut modifier le sous-plan correspondant à la réduction du bruit, si la forme des objets est différente, il faut modifier le sous-plan correspondant à la séparation ou au regroupement des régions. Il est également à noter que, lors de la définition d'un cas, certains critères pourraient être calculés ou déduits du domaine d'application.

L'enrichissement de la base de cas doit passer par sa hiérarchisation et par la définition d'index qui autorisent un accès rapide aux cas.

Nous devons également envisager la gestion des échecs de l'algorithme de remémoration. Si aucune tâche ne peut être proposée ou qu'aucune ne convient à l'utilisateur, le système peut lui proposer de choisir une démarche de travail. Par exemple, une façon d'aborder un problème de segmentation est d'enchaîner les étapes « prétraiter », « détecter », « localiser » et « regrouper », chacune de ces étapes étant un « problème abstrait » de TI (abstrait dans le sens où il doit être défini plus précisément pour être résolu). On pourrait donc proposer à l'utilisateur de choisir sa démarche de travail en sélectionnant la « suite de problèmes abstraits » correspondante. Cette notion de « suite de problèmes » s'inspire de celle présentée dans CommonKADS [Breuker 94b], dans lequel la réutilisation s'appuie sur une typologie des problèmes et sur le fait qu'il existe des dépendances entre ces différents types de problèmes. Après que l'utilisateur ait choisi une « suite de problèmes abstraits », le système pourrait alors relancer le processus de remémoration/adaptation sur chacun des problèmes de la suite, en demandant à l'utilisateur de définir plus précisément les cas cibles correspondants.

Comme on vient de le voir par les perspectives de recherche qu'il propose, ce travail reste ouvert. Mais dans l'état actuel, il propose déjà un cadre permettant de sensibiliser les chercheurs développant des applications à l'intérêt de modéliser les connaissances pour la diffusion du savoir-faire.

Bibliographie

- [Angot 99] F. Angot, *Segmentation d'images 3D ; application à la quantification d'images de tissus biologiques obtenues par microscopie confocale*, Thèse de Doctorat, Université de Caen, 1999.
- [Aussenac 89] N. Aussenac, *Conception d'une méthodologie et d'un outil d'acquisition des connaissances expertes*, Thèse de doctorat, Université Paul Sabatier, Toulouse, 1989.
- [Aussenac 92] N. Aussenac, J.P. Krivine & J. Sallantin, Acquisition des connaissances pour les systèmes à base de connaissances, *Revue d'Intelligence Artificielle*, vol. 6, n° 1-2, 1992.
- [Benamins 92] V.R. Benamins, A. Abu-Hanna & W.N.H. Janweijer, Integrating Problem Solving Methods into KADS, *Interpretation Models for Kads*, 2nd KADS User Meeting, pages 365-362, Sankt Augustin, 1992.
- [Bodington 95] R. Bodington, A software environment for the automatic configuration of inspection systems, *KBUP'95*, Sophia Antipolis, pages 100-108, nov. 1995.
- [Bonzano 96] A. Bonzano, P. Cunningham & C. Meckiff, ISAC : A CBR System for Decision Support in Air Traffic Control, *EWCBR'96*, Lausanne, Switzerland, nov. 1996.
- [Bonzano 97a] A. Bonzano, P. Cunningham & B. Smyth, Using introspective learning to improve retrieval in CBR : A case study in air traffic control, *ICCBR'97*, Rhode Island, USA, juil. 97.
- [Bonzano 97b] A. Bonzano, P. Cunningham & B. Smyth, Learning Feature Weights for CBR : Global versus Local, *AIIA'97*, Roma, Italy, sept. 97.
- [Breuker 94] J.A. Breuker & W. Van de Velde, *The CommonKADS Library for Expertise Modelling*, IOS Press, Amsterdam, The Netherlands, 1994.
- [Breuker 94b] J. Breuker, A suite of Problem Types, *EKAU'94*, Hoegaarden, Belgium, sept. 1994.
- [Capdevielle 95] O. Capdevielle, *Formulation d'objectifs de TI*, Thèse de doctorat, Université Paul Sabatier, Toulouse, janv. 1995.
- [Caulier 95] P. Caulier & B. Houriez, Apport de la modélisation des connaissances à partir de cas pour la capitalisation et la réutilisation des connaissances, *JAVA'95*, Grenoble, France, avr. 1995.
- [Chandrasekaran 87] B. Chandrasekaran, Towards a functional Architecture for Intelligence Based on Generic Information Processing Tasks, *IJCAI'87*, Milan, Italy, pages 1183-1192, 1987.
- [Chandrasekaran 94] B. Chandrasekaran, Understanding Control at the Knowledge Level, *AAAI Fall Symposium*, 1994.
- [Chandrasekaran 96] B. Chandrasekaran & H. Kaindl, Representing Functional Requirements and User-System Interactions, *AAAI-96 Workshop on Modelling and Reasoning about Function*, Portland, OR, August 1996.

- [Charlebois 91] D. Charlebois, J.F. Deguise, D.G. Goodenough, S. Matwin & M. Robson, A Cased-Based Planner to Automate Reuse of ES Software for Analysis of Remote Sensing Data, *IGARSS'91*, Helsinki, Finland, vol.3, pages 1.851-1.854, 1991.
- [Charlebois 94] D. Charlebois, D.G. Goodenough & S. Matwin, Cased-Based Reasoning in an Intelligent Information System for Forestry, *SPIE'94*, Orlando, Florida, pages 64-74, 1994.
- [Charlebois 96a] D. Charlebois, *A Planning System Based on Plan Re-Use and Its Application to Geographical Information Systems and Remote Sensing*, Ph.D. of Computer Science, Ottawa, Canada, july 1996.
- [Charlebois 96b] D. Charlebois, D.G. Goodenough, A.S. Bhogal & S. Matwin, Cased-Based Reasoning and Software Agents for Intelligent Forest Information Management, *IGARSS'96*, Lincoln, Nebraska, pages 2303-2306, 1996.
- [Chassery 91] J.M. Chassery & A. Montanvert, *Géométrie discrète en analyse d'image*, Hermès, Paris, 1991.
- [Chien 96] S.A. Chien & H.B. Mortensen, Automating Image Processing for Scientific Data Analysis of a Large Image Database, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no.8, august 1996.
- [Chien 97] S.A. Chien, F. Fisher, H.B. Mortensen, E. Lo & R. Greeley, Using Artificial Intelligence Planning to Automate Science Data Analysis for Large Image Databases, *KDD'97*, Newport Beach, CA, August 1997.
- [Chiron 97] B. Chiron & A. Mille, Aide à la conception d'environnements de supervision par réutilisation de l'expérience, *JICAA'97*, Roscoff, France, pages 617-628, mai 1997.
- [Chiu 96] T.Z Chiu, J. Archibald & C. Hardy, A Case-Based Reasoning System for Pilot Production Using FMEA Data, *Expersys'96*, Paris, France, oct. 1996.
- [Clément 93] V. Clément & M. Thonnat, A Knowledge-Based Approach to Integration of Image Processing Procedures, *CVGIP Image Understanding*, vol.57, no. 2, march 1993.
- [Clouard 94] R. Clouard, *Planification incrémentale et opportuniste appliquée à la construction de graphes d'opérateurs de Traitement d'Images*, Thèse de doctorat, Caen, fév. 1994.
- [Clouard 95] R. Clouard, C. Porquet, A. Elmoataz & M. Revenu, Why building knowledge based image segmentation systems is so difficult ?, *KBUP'95*, Sophia-Antipolis, nov. 1995.
- [Clouard 97] R. Clouard, A. Elmoataz, F. Angot & O. Lezoray, PANDORE : une bibliothèque et un environnement de programmation d'opérateurs de traitement d'images, *Rapport interne du GREYC*, Caen, France, 1997, (<http://www.greyc.ismra.fr/~regis/Pandora>).
- [Clouard 98] R. Clouard, A. Elmoataz & M. Revenu, Une modélisation explicite et opérationnelle de la connaissance de Traitement d'Images, *RFIA'98*, Clermont-Ferrand, vol. II, pages 65-74, janv. 1998.
- [Cocquerez 95] J.P. Cocquerez & S. Philipp, *Analyse d'image : filtrage et segmentation*, Masson, Paris, 1995.

- [Dejean 96a] P. Dejean & P. Dalle, Modèle symbolique de la donnée de traitement d'images, *RFIA'96*, Rennes, janv. 1996.
- [Dejean 96b] P. Dejean & P. Dalle, Un langage de description de concepts pour la formalisation d'objectifs d'analyse, *ORASIS'96*, Clermont-Ferrand, mai 1996.
- [Delouis 94] I. Delouis, *LISA, un langage réflexif pour la modélisation du contrôle dans les systèmes à base de connaissances, application à la planification de réseaux électriques*, thèse de doctorat, Université de Paris Sud, Orsay, 1994.
- [Deslandes 94] J. Deslandes, *GOLEM : architecture réflexive pour l'introduction des utilisateurs dans la boucle de conception, application à la conception d'un outil d'aide à la création de blés hybrides*, Thèse de doctorat, Université de Caen, 1994.
- [Elmoataz 90] A. Elmoataz, *Mécanismes opératoires d'un segmenteur d'images non dédié : définition d'une base d'opérateurs et implémentation*, Thèse de doctorat, Caen, 1994.
- [Ficet 95] V. Ficet, *Atelier logiciel d'intégration de connaissances en traitement et interprétation d'images*, Rapport de stage de DEA, Université de Caen, 1995.
- [Ficet 96] V. Ficet, Construction interactive d'un modèle conceptuel d'applications de traitement d'images, *RJC-IA*, Nantes, Août 1996, pages 95-102.
- [Ficet 97] V. Ficet-Cauchard, C. Porquet, M. Revenu & R. Clouard, Un système interactif d'aide à la construction d'applications de traitement d'images, *GRETSI'97*, Grenoble, Septembre 1997, pages 949-952.
- [Ficet 98a] V. Ficet-Cauchard., M. Revenu. & C. Porquet, Aide à la conception d'applications de Traitement d'Images : une approche basée sur le Raisonnement à Partir de Cas, *IC'98*, Nancy, Mai 1998, pages 1-10.
- [Ficet 98b] V. Ficet-Cauchard, C. Porquet & M. Revenu, An Interactive Case-Based Reasoning System for the Development of Image Processing Applications, *EWCBR'98*, Dublin, Irlande, Septembre 1998, pages 437-447.
- [Fikes 81] R. Fikes, P. Hart & N. Nilsson, Learning and Executing Generalised Robot Plans, *Readings in AI*, B. Webber and N. Nilsson, Palo Alto, CA, 1981.
- [Fuchs 97] B. Fuchs, *Représentation des connaissances pour le raisonnement à partir de cas : le système ROCADE*, thèse de doctorat, Université Jean Monnet, Saint Etienne, oct. 1997.
- [Gong 95] L. Gong & C. Kulikowski, Composition of Image Analysis Processes Through Object-Centred Hierarchical Planning, *IEEE Transaction on Pattern Analysis an Machine Intelligence*, vol. 17, no. 10, oct. 1995.
- [Grimnes 96] M. Grimnes & A. Aamodt, A two layer case-based reasoning architecture for medical image understanding, *EWCBR'96*, Lausanne, Suisse, nov. 1996.
- [Hasegawa 86] J. Hasegawa, H. Kubota & J. Toriwaki, Automated Construction of Image Processing Procedures by Sample-Figure Presentation, *ICPR'86*, Paris, pages 586-588, 1986.

- [Haton 91] J.P. Haton, N. Bouzid, F. Charpillet, M.C. Haton, B. Lâasri, H. Lâasri, P. Marquis, T. Mondot & A. Napoli, *Le raisonnement en IA*, Chapitre 8-9, InterEditions, Paris, France, 1991.
- [Hayes-Roth 95] B. Hayes-Roth, A blackboard architecture for control, *Artificial Intelligence*, vol. 26, 1985.
- [Iris 93] *IRIS Explorer User's Guide*, Silicon Graphics, Inc., Mountain View, California, n°007-1371-020, 1993.
- [Jacob-Delouis 96] I. Jacob-Delouis & O. Jehl, LISA-runtime (LIS@RT) : vers une utilisation industrielle du langage LISA, JAVA'96, Sète, mai 1996.
- [Kolodner 91] J.L. Kolodner, Improving Human Decision Making through Cased-Based Decision Aiding, *AI Magazine*, vol. 12, pages 52-68, 1991.
- [Kunt 93] M. Kunt, *Traitement de l'information vol. II : traitement numérique des images*, Presses Polytechniques et Universitaires Romandes, Lausanne, Suisse, 1993.
- [Lefevre 93] V. Lefèvre & Y. Pollet, BBI : un système multi-agents d'aide à la photo-interprétation, *13^{ième} journées internationales Intelligence Artificielle, Système Expert et Langage Naturel*, Avignon, pages 473-482, mai 1993.
- [Lefevre 95] V. Lefèvre, Y. Pollet & S. Philipp, Image processing co-operation using Dempster-Shafer theory, *Scandinavian Conference on Image Analysis*, Uppsala, Sweden, 1995.
- [Lefevre 96] V. Lefèvre, Y. Pollet, S. Philipp & S. Brunesseaux, Un système multi-agents pour la fusion de données en analyse d'images, *Traitement du Signal*, vol. 13, no. 1, pages 100-111, janv.1996.
- [Lieber 97] J. Lieber & A. Napoli, Planification à partir de cas et classification, *JICAA'97*, Roscoff, France, pages 617-628, mai 1997.
- [Maes 87] P. Maes, Computational reflection, technical report, *MIT Lab*, n° 87-2, 1987.
- [Marion 87] A. Marion, *Introduction aux techniques du traitement d'images*, Eyrolles, Paris, 1987.
- [Marr 82] D. Marr, Vision : A computational investigation into the human representation and processing of visual information, *W. H. Freeman & Co Eds.*, San Francisco, USA.
- [Matta 95] N. Matta, *Méthodes de résolution de problèmes : leur explicitation et leur représentation dans MACAO II*, thèse de doctorat, Université de Toulouse, 1995.
- [Moire 94] T. Moire, *Etude d'une architecture logicielle basée sur un modèle opérationnel de la connaissance, application à la définition d'un système d'aide à la conduite de projet*, Thèse de doctorat, Université de Caen, 1994.
- [Monclar 94] F.R. Monclar, I. Delouis & J.P. Krivine, Rôles et modes de coopération dans les SBC coopératifs, *Atelier SBC coopératifs, IA'94*, Paris, 1994.
- [Monclar 96] F.R. Monclar, ELICO pour le développement de systèmes à base de connaissances coopératifs, Rennes, janv. 1996.

- [Muñoz-Avila 96] H. Muñoz-Avila & J. Hüllen, Feature Weighting by Explaining Cased-Based Problem Solving Episodes, *EWCBR'96*, Lausanne, Suisse, nov. 1996.
- [Netten 96a] B.D. Netten & R.A. Vingerhoeds, Incremental Adaptation for Conceptual Design in EADOCS, *Workshop on Adaptation in Case-Based Reasoning, ECAI'96*, Budapest, Hungary, pages 29-34, aug. 1996.
- [Netten 96b] B.D. Netten & R.A. Vingerhoeds, Case Combination for Conceptual Design, *EWCBR'96*, Lausanne, Suisse, nov. 1996.
- [Netten 97] B.D. Netten & R.A. Vingerhoeds, Structural Adaptation by Case Combination in EADOCS, *5th German Workshop on Case-Based Reasoning, GWCBR'96*, Bad Honnef, Germany, march 1997.
- [Newell 82] A. Newell, The knowledge level, *Artificial Intelligence*, n°18, pages 87-127, 1982.
- [Pitrat 90] J. Pitrat, *Métaconnaissance : futur de l'intelligence artificielle*, Hermès, Paris, 1990.
- [Prasad 95] B. PRASAD, Planning With Case-Based Structures, *AAAI Fall Symposium*, MIT Campus, Cambridge, Massachusetts, nov. 95.
- [Rasure 94] J. Rasure & S. Kubica, The Khoros Application Development Environment, Experimental Environments for Computer Vision and Image Processing, editor H.I. Christensen and J.L. Crowley, *Word Scientific*, Singapore, pages 1-32, 1994.
- [Rechenman 90] F. Rechenman, P. Fontanille & P. Uvietta, *Shirka, système de gestion de base de connaissances centrées objets*, manuel d'utilisation, INRIA Rhône-Alpes, 1990.
- [Revenu 93] M. Revenu, A. Elmoataz, C. Porquet & H. Cardot, An automatic system for the classification of cellular categories in cytological images, *Intelligent Robots and Computer Vision XII: Algorithms and Techniques*, SPIE, Boston, USA, vol. 2055, pages 967-970, sept.1993.
- [Schreiber 94] G. Schreiber, B. Wielinga, R. de Hoog, H. Akkermans & W. Van de Velde, CommonKADS : A Comprehensive Methodology for KBS Development, *IEEE Expert*, vol. 9 (6), pages 28-37, déc. 1994.
- [Smyth 95] B. Smyth & M.T. Keane, Retrieval & Adaptation in Déjà Vu, a Case-Based Reasoning System for Software Design, *AAAI Fall Symposium*, MIT Campus, Cambridge, Massachusetts, nov. 1995.
- [Smyth 96] B. Smyth, *Case-Based Design*, Doctoral Thesis of the Trinity College, Dublin, Ireland, april 1996.
- [Theot 92] C. Théot, E. Gallier & D. Mischler, VSDE, un environnement de développement automatisé de systèmes de vision, *Revue technique Thomson-CSF*, vol. 24, no. 4, pages 867-885, déc. 1992.
- [Thomas 96] J. Thomas, *Vers l'intégration de l'apprentissage symbolique et de l'acquisition de connaissances basée sur des modèles : le système ENIGME*, thèse de doctorat, Université Pierre et Marie Curie, Paris VI, déc. 1996.

- [Thonnat 93] M. Thonnat, V. Clément & J. van den Elst, Supervision of perception tasks for autonomous systems : the OCAPI approach, *Rapport de recherche no. 2000*, INRIA, juin 1993.
- [Thonnat 95] M. Thonnat & S. Moisan, Knowledge-based systems for program supervision, *KBUP'95*, Sophia-Antipolis, nov. 1995.
- [Van den Elst 94] J. van den Elst, F. van Harmelen, G. Scheiber & M. Thonnat, A fonctionnal specification of reusing software components, *SEKE'94*, Tallin, Estonie, june 1994.
- [Van den Elst 95] J. van den Elst, F. van Harmelen & M. Thonnat, Modelling software components for reuse, *SEKE'95*, Knowledge Systems Institute, Illinois, june 1995.
- [Veloso 96] M. Veloso, H. Munoz-Avila & R. Bergmann, cased-based planning : selected methods and systems, *AI Communications*, vol. 9, n. 3, sept. 1996.
- [Vincent 94] R. Vincent, S. Moisan & M. Thonnat, Learning as mean to refine a KBS, *3rd Japanese Knowledge Acquisition for Knowledge Based Systems Workshop*, 1994
- [Wielinga 92] B. Wielinga, A.T. Schreiber & JA breuker, KADS : a modelling approach to knowledge engineering, *Knowledge Acquisition*, vol. 4 (1), pages 5-53, 1992.
- [Willamowski 94a] J. Willamowski, Modélisation de tâches pour la résolution de problèmes en coopération avec l'utilisateur, *RFIA'94*, Paris, janv. 1994.
- [Willamowski 94b] J. Willamowski, *Modélisation de tâches pour la résolution de problèmes en coopération avec l'utilisateur*, thèse de doctorat, Université Joseph Fournier, Grenoble, avr. 1994.