Une méthode de développement d'applications de traitement d'images

An image processing applications development method

Régis Clouard
GREYC - IMAGE, 6 boulevard Maréchal Juin, F-14050 Caen cedex, France
Mèl: Regis.Clouard@greyc.ensicaen.fr
Tél: 02 31 45 29 22
Fax: 02 31 45 26 98

Manuscrit reçu le 30 mars 2004

Résumé et mots clés

Nous proposons une méthode de développement d'applications de traitement d'images qui se présente comme un guide complet et rigoureux pour la gestion du cycle de vie entier d'une application. Cette méthode met en avant des capacités d'aide, de réutilisabilité d'expériences, de reproduction des résultats, d'uniformisation des représentations et de communication entre les différents partenaires du développement, par la définition d'une part de modèles destinés à collecter et organiser la connaissance mise en jeu et d'autre part de cycles destinés à conduire la mise en oeuvre. Cet article se focalise sur la description des modèles qui font le coeur et l'originalité de cette méthode. Nous montrons que notre vision de la modélisation d'une application se fonde sur l'idée force qu'une application de traitement d'images s'observe selon quatre points de vue complémentaires dont la sémantique est capturée par quatre modèles spécifiques : le modèle du système, le modèle du domaine, le modèle des tâches et le modèle du programme.

Traitement d'images, Ingénierie des connaissances, Génie logiciel, Pilotage de programmes, Graphe d'opérateurs, Programmation visuelle.

Abstract and key words

A new image processing application development method is presented, which is a complete and rigorous guide for the management of the whole life cycle of an application. This method points out aids, reusing, reproducibility and unifying capabilities for knowledge acquisition and for communication between the different intervening party, by providing in one hand structured models in order to collect and to organize involved knowledge and in the other hand rational cycles in order to make use of the knowledge. This article focuses on the description of the models that are the heart and the originality of the method. We show that our vision of application modeling relies on the idee-force that an image processing application is studied through out four points of view whose semantic is captured by four related models: system model, domain model, tasks model and program model.

Image processing, Knowledge engineering, Software engineering, Program supervision, Graph of operators, Visual programming.

1. Introduction

Le besoin en applications de traitement d'images se fait de plus en plus pressant à mesure que l'image numérique s'impose comme un support et une source d'information privilégiés. La multiplication des dispositifs d'acquisition conduit à la production d'un très grand nombre d'images qu'il devient difficile d'exploiter manuellement. Ces dispositifs permettent d'accéder à des informations jusqu'ici inconnues ou inaccessibles, qui prennent aujourd'hui une importance stratégique dans de nombreux domaines comme la biomédecine, la télésurveillance, le contrôle qualité, etc. Le traitement d'images s'inscrit dans un processus préliminaire destiné à préparer automatiquement les images à leur analyse, leur interprétation, leur restauration, leur archivage ou leur transmission. Il n'a aucun pouvoir décisionnel, mais son rôle est crucial puisqu'il consiste à extraire des images des informations qualitatives ou quantitatives par des procédures de réduction et d'abstraction de l'information initiale sans perte ni falsification de l'information pertinente.

Notre définition d'une application de traitement d'images s'arrête à *un logiciel spécialisé dans l'accomplissement d'un ensemble d'objectifs de transformations d'images en images et dont les images d'entrée relèvent d'une classe définie*. Nous supposons l'application de traitement d'images parfaitement localisée dans l'application globale qui la contient. Cette définition détermine complètement une application à partir d'un ensemble d'objectifs et d'une classe d'images. Les objectifs se rapportent aux tâches couvertes par le traitement d'images : améliorer la qualité subjective ou objective des images, segmenter en régions, compresser le contenu, restaurer des informations manquantes ou détériorées, détecter la présence d'objets, extraire des objets ou calculer des caractéristiques. La classe d'images est un ensemble d'images numériques défini par intention à travers une liste d'invariants que vérifient les images. La notion d'image s'entend dans le sens le plus général, qu'elle soit la représentation d'un phénomène mesuré ou calculé. Les invariants sont des descripteurs d'image qui synthétisent significativement le contenu des images et le phénomène observé. Cette définition implique que toute modification d'un objectif ou de la classe d'images d'une application conduit irrémédiablement à changer d'application. Mais, elle impose aussi qu'une application possède toutes les capacités de flexibilité et de robustesse nécessaires à l'adaptation de son comportement face à la variété inhérente des images qui relèvent de sa classe d'entrée.

Il existe une grande variété d'applications réelles décrites dans la littérature. Nous prenons en exemple fil rouge le cas d'une application biomédicale tirée de l'étude détaillée dans [Lezoray-00]. Cette application concerne l'analyse d'images microscopiques de la cytologie des séreuses : *La cytologie est l'examen d'un étalement de cellules présenté sur une lame. En anatomie pathologique, elle est notamment utilisée pour le diagnostic et le suivi du cancer à partir d'une «lecture» des lames. Le but de cette lecture est la détection de cellules anormales ou suspectes. L'objectif de l'application globale est d'assister le cytotechnicien lors de la lecture des lames. L'objectif particulier du traitement des images est de segmenter automatiquement les images pour isoler chaque cellule de séreuse dans une région. Les régions sont ensuite passées dans un classificateur qui ne retient que celles localisant des cellules de séreuse susceptibles d'être malignes. La classe d'images est composée d'images microscopiques acquises avec un grossissement* $\times 20$ *qui visualisent des cellules de séreuse et des globules rouges répartis sur une lame (exemple Fig. 1). Cette application est d'un intérêt capital puisque la lecture d'une lame est décrite comme une activité humaine particulièrement laborieuse et le risque de faux-négatifs (la non détection de cellules malignes) est la principale cause d'erreur de diagnostic.*

Malheureusement, le développement d'une nouvelle application reste une tâche difficile à la fois fastidieuse et complexe [Crevier-97]. La principale raison est le manque de méthode idoine qui assiste le spécialiste de traitement d'images dans l'organisation de ses développements. Le manque de modèle empêche notamment de profiter pleinement de l'expérience accumulée que ce soit pour la réutilisation ou pour la simple reproduction de solutions. La plupart du temps les applications de traitement d'images sont développées de manière empirique, ce qui constitue le frein à une réelle pénétration du traitement d'images dans le génie industriel.

Notre contribution porte sur la définition d'une méthode qui est destinée à rendre intelligible l'activité des spécialistes, qu'elle soit de développer de nouvelles applications (ingénierie) ou de modéliser des applications existantes (re-ingénierie). La méthode propose des modèles pour collecter et organiser les informations en connaissances ainsi que des cycles pour conduire la mise en oeuvre. L'enjeu n'est pas encore de rendre le traitement

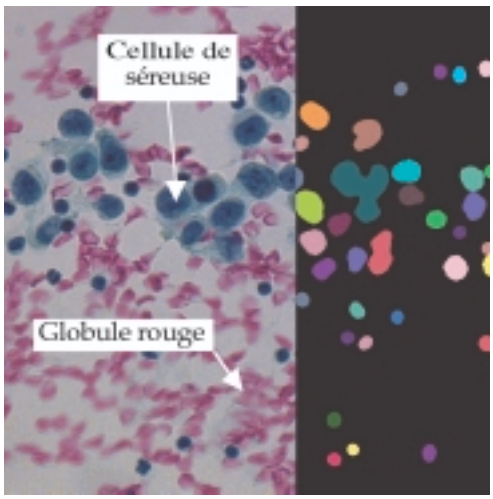

Figure 1. L'objectif du traitement d'image est de segmenter les images de cytologie pour isoler les cellules de séreuse marquées en bleu (partie gauche de l'image) dans une région (partie droite de l'image).

d'images accessible aux non-spécialistes mais de fournir aux spécialistes des schémas de développement plus complets et plus généraux qui visent à rendre leurs productions plus efficaces, plus robustes et plus fiables. Dans cet article, nous nous limitons à la présentation des modèles qui font le coeur et l'originalité de la méthode. Les cycles de développement qui organisent l'élaboration et la validation des modélisations s'inspirent de ceux du génie logiciel, notamment le cycle de vie RUP [Kruchten-96] et le cycle auteur/lecteur de SADT [Marca-88]. Dans la section 2, nous justifions la nécessité de développer une méthode idoine en démontrant que la modélisation mathématique est insuffisante, que les méthodes du génie logiciel sont impuissantes et que les systèmes à base de connaissances sont encore immatures. Dans la section 3, nous présentons les fondements de la modélisation d'applications que nous proposons. Nous montrons qu'elle se définit autour de quatre modèles distincts qui abordent chacun un aspect de l'application. Chaque modèle est ensuite détaillé respectivement dans les sections 4, 5, 6 et 7. Enfin, en conclusion, nous dressons un bilan en terme de bénéfices et de limites d'utilisation de cette méthode.

2. La problématique du traitement d'images

2.1. Un problème non purement numérique

Pour développer une application, les solutions sont bien évidemment à rechercher dans les résultats scientifiques du domaine du traitement d'images. Les efforts de recherche menés depuis les années 50 ont conduit à la réalisation d'une quantité impressionnante d'algorithmes aussi divers que les applications abordées. Ils permettent maintenant d'élaborer des modèles formels des traitements de plus en plus généraux. Différentes approches de modélisation sont ainsi explorées simultanément, parmi lesquelles l'analyse mathématique, les approches variationnelles, statistiques ou géométriques, la théorie des graphes, la morphologie mathématique ou encore les problèmes inverses. Mais, bien que le traitement d'images repose sur de nombreuses références mathématiques, la résolution d'objectifs définissant une application n'est pas un problème purement numérique. Il n'existe pas de théorie formelle globale qui prenne en compte la totalité d'un problème de traitement d'images. Chaque approche s'élabore sur des modèles déterminés de l'information à traiter et s'emploie comme une théorie particulière adaptée seulement à la résolution d'une partie des objectifs sur une partie des types d'images. Par exemple, l'approche variationnelle est adaptée à la détection de contours quand l'image peut se voir comme une fonction continue par morceaux de l'intensité (cas des images de scènes naturelles). Par contre, elle est peu adaptée quand l'image est de nature statistique (cas des images de Tomographie par Émission de Positon), car la notion de contour n'est plus basée sur le modèle de la variation d'intensité. Inversement, plusieurs approches peuvent concourir à une même tâche sur un même type d'image. Par exemple, la morphologie mathématique est une alternative mais aussi un complément de l'approche variationnelle pour la détection de contours dans le cas d'images de scènes naturelles. Néanmoins, ces deux approches ne reposent pas sur une même formulation des connaissances *a priori* et n'utilisent pas nécessairement les mêmes descripteurs (*e.g.*, la forme des objets en morphologie mathématique versus la forme du signal pour les approches variationnelles).
La résolution d'objectifs d'application passe donc par l'intégration de plusieurs approches que le développeur doit savoir choisir et adapter en fonction des objectifs ponctuels à atteindre et des caractéristiques de la classe d'images [Cocquerez-95].

2.2. Un problème complexe

L'absence de théorie formelle globale fait du développement d'applications un problème complexe. En réduisant le développement logiciel aux quatre activités nominales : analyse des besoins, conception, programmation et évaluation, on peut montrer que les difficultés émanent de chacune des activités.

1. L'*analyse des besoins* est essentiellement de nature qualitative au sens de [Reichgelt-91]. Cela signifie que l'expression d'un problème ne peut se réduire à une tension entre un état de départ (la classe d'images) et un état d'arrivée (les résultats attendus). En effet, d'une part la caractérisation de la classe d'images ne peut être ni exhaustive ni exacte [Poggio-85], parce qu'une image sous-contraint par nature la scène qu'elle représente (*e.g.*, perte de la troisième dimension) et parce que plusieurs facteurs se retrouvent confondus dans l'apparence d'un objet (*e.g.*, couleur, réflexion, illumination, conditions atmosphériques, point de vue) dont il est impossible de mesurer les contributions relatives. D'autre part, la meilleure description des résultats formulable reste généralement insuffisante pour constituer un état d'arrivée consistant, à cause de la quantité d'information à caractériser, de la faiblesse d'expressivité des descripteurs et de la variabilité des configurations possibles. L'analyse des besoins nécessite donc de savoir faire des choix subjectifs qui vont approximer au mieux les besoins à travers la recherche d'un compromis entre l'intention que le client possède dans les termes de sa spécialité et sa formulation que le développeur doit énoncer en termes de tâches à accomplir sur des images d'une classe définie par des caractéristiques observables.
2. La *conception* de solutions est typiquement une activité complexe au sens de [deRosnay-75], dans la mesure où elle fait appel à l'intégration de connaissances de formes et d'origines diverses, pour résoudre des problèmes dont les sorties ne sont pas en relation causale simple avec les entrées. En effet, premièrement, les informations sur les images sont disponibles à la fois sous forme numérique et symbolique, tandis que les

traitements qui les utilisent reposent sur des modèles numériques plus ou moins explicités. Par exemple, l'utilisation d'un détecteur de contours suppose de savoir reconnaître sur l'image à traiter le modèle particulier de contours que le détecteur code numériquement (*e.g.*, la forme, le contraste, la largeur). Deuxièmement, les différentes approches du traitement des images s'appuient sur des points de vue différents sur les connaissances et prennent leurs références dans des domaines théoriques également différents (*e.g.*, traitement du signal, analyse numérique, statistiques, optique, algorithmique) [Garbay-01]. Chacune d'elles conduit à une formulation propre des problèmes et de leurs solutions qui rend souvent difficile la comparaison et la coopération d'approches. Enfin troisièmement, la construction des résultats nécessite de fréquents changements de représentations intermédiaires comme les changements de niveau d'abstraction (*e.g.*, pixel en contour, contour en région) ou de référentiel (*e.g.*, spatial, fréquentiel, photométrique) qui ne peuvent pas être décidés à partir des données d'entrée.

3. La *programmation* est ici une activité compliquée. Elle s'inscrit dans un environnement qui comporte un très grand nombre d'éléments à combiner. La masse de données et d'opérations rend les algorithmes de traitement laborieux à coder. Les tests en particulier, ne peuvent s'appuyer ni sur le contrôle visuel des résultats, qui est insuffisant, ni sur l'examen des valeurs des pixels des images de sortie, qui est fastidieux.
4. L'*évaluation* des résultats est confrontée aux difficultés de communication entre le concepteur et le client pour valider la définition des objectifs, et aux difficultés de diagnostiquer les causes d'erreurs pour vérifier les résultats. Les critères d'évaluation existent bien mais ils ne s'expriment pas toujours quantitativement. Quand ils le sont, ils portent sur les résultats finals et très rarement sur les résultats intermédiaires. Il n'est alors pas trivial de remonter la chaîne de traitements pour localiser les causes des erreurs. *e.g.*, *dans l'application de cytologie, une erreur de localisation des frontières des cellules peut être la cause d'une description incomplète de l'objet cellule, d'un mauvais choix de la méthode de localisation ou de ses paramètres*.

En conséquence, pour une même application il existe une très grande diversité de solutions plus ou moins adaptées, dont les caractéristiques dépendent étroitement des domaines de compétence de leur concepteur.

2.3. Un problème non purement logiciel

À défaut de théorie globale, et en s'appuyant sur le fait qu'une application est un logiciel à part entière, il est alors naturel de se tourner vers le génie logiciel. Mais les méthodes du génie logiciel et leur langage de modélisation, comme SADT [Ross-77], JSD [Jackson-83], OMT [Rumbaugh-91] ou UML [OMG], s'avèrent n'être que de peu d'utilité pour développer des applications de traitement d'images. Leur champ d'action se limite quasiment à l'activité de programmation. Elles n'apportent en effet aucune solution aux difficultés énoncées précédemment et laissent alors le développeur seul avec ses compétences « naturelles » pour découvrir le problème à résoudre, proposer des solutions et les évaluer [Zamperoni-96]. Le développement d'une application de traitement d'images n'est donc pas un problème purement logiciel.

2.4. Immaturité des systèmes à base de connaissances

Pour réduire la charge cognitive du développeur, différentes propositions ont été faites principalement dans les années 1990. Les plus ambitieuses d'entre elles ont porté sur la réalisation de systèmes à base de connaissances capables d'automatiser entièrement le développement d'une application pour peu que son utilisateur soit capable de formuler le problème à résoudre dans le formalisme du système (*e.g.*, [Liedtke-92], [Clément-93], [Chien-96], [Clouard-99]). Mais tous ces systèmes ont montré leurs limites, essentiellement dans le nombre de configurations qu'ils sont capables d'envisager. La cause est toujours l'absence de modélisation globale. Néanmoins, ces travaux ont pour la plupart érigé le pilotage d'une bibliothèque de codes comme le paradigme de base de la conception d'applications [Thonnat-99]. Les techniques de traitement d'images sont implantées sous forme de codes exécutables indépendants et regroupés dans une bibliothèque. L'expertise (les stratégies et les méthodes) est codée par des plans associant des objectifs aux chaînes de codes adaptées à leur accomplissement.

2.5. Vers une méthode pour le traitement d'images

Pour pallier aux limitations des systèmes précédents, S. Moisan [Moisan-00] propose de rendre l'utilisateur plus responsable du développement en étant plus informé sur le système. Cela se fait en couplant un système de pilotage qui détient l'expertise pour réaliser des tâches métiers, avec un livre des connaissances (élaboré à partir de la méthode MKSM [Ermine-00]) qui renferme la connaissance sur le savoir-faire nécessaire à la réalisation de ces tâches. L'utilisateur peut ainsi mieux exploiter la bibliothèque de codes à sa disposition et envisager d'adapter les solutions proposées ou d'en construire de nouvelles parce qu'il a une meilleure maîtrise de ces solutions.

Notre motivation est aussi de favoriser la réutilisation du patrimoine d'expertise déjà bien riche mais sous exploité parce que non structuré. Mais notre approche est différente puisqu'elle se veut un cadre général pour modéliser ce patrimoine, c'est-à-dire qu'elle prend en compte non seulement les solutions opérationnelles mais aussi la formulation des problèmes et la logique de conception qui les supportent.

3. Les fondements de la modélisation

Notre modélisation tire sa définition d'une part de contraintes formulées à partir de la problématique du traitement d'images et d'autre part de solutions inspirées des travaux portant sur l'automatisation du développement d'applications de traitement d'images.

3.1. Les contraintes fixées par le traitement d'images

La prise en compte de la complexité du développement d'une application de traitement d'images dans une modélisation passe par l'acceptation de quatre contraintes fortes :

1. *Elle doit être le support de la simulation*. L'exécution est une activité essentielle pour le développement. D'une part, elle permet un développement itératif et incrémental basé sur l'évolution de prototypes mesurables, pour assurer des cycles spécification-évaluation afin de faire converger simultanément la définition du problème et le logiciel correspondant. D'autre part, certaines informations sur la définition du problème lui-même ne sont disponibles qu'après l'exécution d'opérations, soit parce que ces informations sont noyées dans la masse des données, soit parce qu'elles n'existent que par rapport à un autre référentiel (*e.g.*, spectre de Fourier, espace de Hough) ou à une autre représentation (*e.g.*, matrice de co-occurrences, carte de contours).
2. *Elle doit englober les différentes approches du traitement des images*. Aucune approche ne peut être privilégiée. Au contraire, une application se construit par coopération ou compétition de plusieurs approches (*e.g.*, morphologie mathématique, variationnelle, stochastique, fréquentielle). Toutes les formulations de problèmes et de leurs solutions doivent pouvoir s'y modéliser.
3. *Elle doit intégrer la multi-disciplinarité*. Le développement d'une solution logicielle complète fait appel à trois types de partenaires : le client, le concepteur et le programmeur. Le client possède le problème à résoudre dans les termes de sa spécialité. Il est capable d'évaluer visuellement les résultats finals ou de proposer des moyens de les évaluer. Il est supposé ignorant en traitement d'images. Le concepteur est un expert du traitement d'images qui a la connaissance des outils de développement. Il pose le problème en termes de traitement d'images et élabore sa solution. Il est supposé ignorant dans le domaine métier. Enfin, le programmeur est un informaticien dont le rôle est d'écrire les codes de la bibliothèque.
4. *Elle doit définir un référentiel d'évaluation des applications*. Ce référentiel doit distinguer les trois axes que sont : la validation, la vérification et les tests logiciels. La validation a pour but de s'assurer que les objectifs définis répondent aux attentes du client et aux contraintes de l'environnement. La vérification a pour but de prouver que la solution satisfait aux objectifs définis. Enfin les tests logiciels ont pour but de garantir l'intégrité du code produit.

3.2. Les solutions inspirées des travaux sur l'automatisation

Profitant de l'expérience acquise autour des travaux sur l'automatisation du traitement d'images, nous avons choisi de fonder la modélisation d'applications sur le paradigme du pilotage d'une bibliothèque de codes. Ce paradigme définit deux principes de base :

1. *La conception d'une solution est vue comme l'élaboration d'un plan d'analyse hiérarchique*. Ce plan est destiné à guider le choix et l'enchaînement des traitements à opérer. Il s'exprime par un arbre de tâches où une tâche de l'arbre est la description abstraite d'un traitement pour un niveau d'abstraction. L'arbre se construit par décomposition de tâches complexes en tâches plus simples jusqu'à des tâches décrivant des opérations directement réalisables. Ce paradigme a déjà démontré son efficacité pour la formalisation d'expertises en analyse d'images [Thonnat-99].
2. *La programmation est vue comme la construction d'une chaîne de codes exécutables*. Elle s'apparente alors à la programmation visuelle par flot de données qui s'est largement répandue dans la communauté du traitement d'images autour de logiciels comme Khoros ou Matlab. Les avantages de ce paradigme pour la programmation d'applications de traitement d'images sont bien connus ([Baroth-94], [Whitley-01]) ; il constitue en particulier une réponse assez satisfaisante à la nature compliquée de la programmation.

Le paradigme du pilotage d'une bibliothèque de codes fait de la modélisation des solutions conceptuelles et des programmes une approche résolument fonctionnelle. L'élaboration d'une solution est envisagée sous l'angle des traitements ; les données ne sont considérées que comme un flot d'informations échangé entre les traitements.

3.3. Présentation globale de la modélisation

Notre vision de la modélisation d'une application de traitement d'images repose sur l'idée force qu'une application s'analyse selon quatre points de vue distincts et complémentaires. Les quatre points de vue correspondent à une discrétisation de l'application qui isole : la formulation des objectifs, la définition de la classe d'images, la conception d'une solution et le codage du programme exécutable. À chaque point de vue est consacré un modèle qui en propose une vue abstraite élaborée à partir d'une hypothèse et qui est destiné à capturer la partie de la sémantique de l'application vue sous cet angle. Dans l'ordre des points de vue, nous distinguons les modèles respectifs :

1. *Le modèle du système*. C'est le point de vue externe sur l'application où l'on cherche à définir « Quel est le problème à

résoudre ?». Ce modèle se fonde sur l'hypothèse systémique qui considère une application comme un **système de type boîte noire** transformant des images d'entrée en images de sortie. C'est une hypothèse délibérément simplificatrice mais qui envisage l'application dans sa réalité opératoire. Ce modèle vise ainsi à identifier l'application à travers ses finalités et ses interactions avec son environnement et rend compte du fait qu'une application de traitement d'images n'est pas une fin en soi, mais une partie d'une application plus globale dont elle dépend fonctionnellement. Il devient le support indiqué de la simulation et définit l'axe de la validation de l'application car ses informations sont accessibles à la critique du client.

2. *Le modèle du domaine*. C'est le point de vue contextuel sur l'application où l'on cherche à découvrir «Quel est le contexte de l'application ?». La définition de ce modèle repose sur l'hypothèse phénoménologique qui postule qu'une dénotation des images est nécessaire et suffisante pour concevoir des solutions satisfaisant les objectifs de l'application. Une dénotation se distingue d'une connotation en ce qu'elle ne cherche à donner un sens aux images à traiter que par désignation extensionnelle des informations à considérer comme pertinentes. C'est à dire que l'on peut se contenter d'une description «objective» des concepts métiers qui soit isolée de leur interprétation. Néanmoins, la connotation est nécessaire au client pour identifier des dénotations valides ; il est supposé expert dans le domaine de l'étude. L'intérêt d'une dénotation par rapport à une connotation, c'est qu'elle n'a pas besoin d'être exhaustive ni exacte pour le domaine d'étude, mais simplement suffisante pour les objectifs de traitement considérés. *e.g., pour traiter les images de cytologie de séreuse, il n'est nul besoin d'être averti en matière de dépistage du cancer. Le concept de cellule peut se réduire à une définition «visuelle» telle que «un objet uniforme, de forme oblongue, de petite taille, de couleur bleu…»*. Toutefois, une dénotation représente un parti pris sur les images selon les informations que l'on veut privilégier. Il existe donc beaucoup de dénotations valides d'une même classe d'images pour une même connotation, ce qui explique le grand nombre de solutions satisfaisantes pour une même application.
3. *Le modèle des tâches*. C'est le point de vue analytique sur l'application où l'on tente de comprendre «Comment doit fonctionner l'application ?». Ce modèle repose sur l'hypothèse réductionniste qui propose de formuler une solution conceptuelle sous la forme d'un **plan d'analyse hiérarchique** qui associe en plusieurs niveaux d'abstraction les objectifs initiaux à une chaîne d'algorithmes de traitement. La modélisation des solutions par arbres de tâches permet l'encapsulation et l'abstraction de l'expertise sous forme de granules de connaissances hiérarchiques définissant ainsi un moyen d'intégration des différentes approches du traitement d'images. Ce modèle définit l'axe de la vérification des solutions et profite d'une graduation des solutions en niveaux d'abstraction.
4. *Le modèle du programme*. C'est le point de vue logiciel où l'on s'intéresse à la question «Comment doit être codée l'application ?». L'hypothèse de base est celle de la programmation visuelle qui suppose que l'on peut représenter n'importe quel programme de traitement d'images par un **graphe d'opérateurs** individualisés. C'est à dire qu'un processus de traitement aussi complexe qu'il soit peut toujours se discrétiser en une séquence de traitements ponctuels, dont les liens peuvent se réduire à des images ou des valeurs numériques. Ce mode de programmation est le moyen de prototypage par excellence qui rend possible la simulation. Ce modèle définit l'axe des tests logiciels.

L'élaboration et l'évaluation de ces modèles résultent de négociations entre les trois types de partenaires. Plus exactement, chaque point de vue est le siège d'un dialogue, légitimant alors la multi-disciplinarité. Le modèle du système se définit entre le concepteur et le client, le modèle du domaine se découvre entre le concepteur et le client, le modèle des tâches s'élabore entre concepteurs (le mode de raffinement des solutions s'inspire largement du cycle auteur/lecteurs de SADT [Marca-88]) et le modèle du programme se met au point entre le programmeur et le concepteur.

Dans les sections suivantes, nous détaillons successivement les quatre modèles et pour chacun d'eux nous définissons le référentiel de modélisation et nous proposons une représentation possible des informations perçues.

4. Le modèle du système

4.1. Définition

Le modèle du système identifie les informations qui définissent les objectifs de l'application. L'approche systémique donne le contexte théorique pour modéliser une application de traitement d'images en tant que système [Simon-69], [Le Moigne-77]. On y reconnaît alors un système de complexité de niveau 4, c'est-à-dire «un système ouvert qui s'informe avec un comportement théoriquement prévisible» (Fig. 2). Selon cette approche, un système s'étudie et se conçoit en considérant les informations selon 3 aspects entre ce qu'il est (aspect structurel), ce qu'il fait (aspect fonctionnel) et ce qu'il devient (aspect temporel).

4.1.1. Aspect structurel

Sous l'aspect structurel, un système de traitement d'images est principalement décrit par sa frontière (si nécessaire, la modélisation des composants internes se fait en empruntant les représentations diagrammatiques classiques du génie logiciel, par exemple les diagrammes de composants et de déploiement d'UML). La délimitation de la frontière coïncide ici avec la définition des objectifs à atteindre. Toutefois, il est nécessaire de

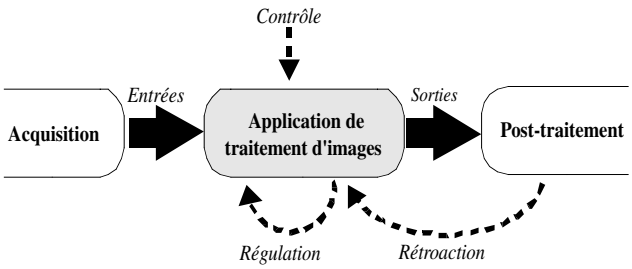

Figure 2. Une application de traitement d'images est vue comme un système intermédiaire consommant des images d'un système d'acquisition et produisant des images pour un système de post-traitement. Les données de contrôle, le processus de régulation et la boucle de rétroaction identifient les différentes formes d'interaction avec l'environnement d'exécution.

spécifier non seulement les objectifs (*i.e.*, les limites) du système de traitement d'images, mais aussi ceux des deux systèmes qui l'encadrent (acquisition et post-traitement) ainsi que ceux du système global qui les inclut tous :

- Les *objectifs globaux* cernent le domaine et renseignent le cadre et le but de l'étude dans laquelle s'intègre l'application de traitement d'images. *e.g.*, *l'application de cytologie fait partie d'un système général qui vise à assister le pathologiste lors de l'examen d'un étalement de cellules pour le dépistage et le suivi du cancer…*
- Les *objectifs du système d'acquisition* décrivent les modes de préparation de la scène pour mettre en évidence le phénomène à observer et détaillent la chaîne de production des images d'entrée. Le niveau de description doit être suffisant pour permettre à d'autres clients du domaine de reproduire la même classe. *e.g.*, *pour construire les images de cytologie, les prélèvements de séreuses sont étalés sur une lame puis fixés et colorés par la méthode de Papanicolaou afin de mettre en évidence les cellules de séreuse. Les images sont ensuite acquises sous microscope avec un grossissement de* $20\times$ *…*
- Les *objectifs du post-traitement* définissent la façon dont seront utilisés les résultats du traitement d'images et par qui. Le niveau de description doit garantir l'intégration du système de traitement d'images au système de post-traitement. *e.g.*, *les images résultats seront utilisées par un système de tri cellulaire à partir de la forme, la taille et de la texture des noyaux de cellules de séreuse*.
- Les *objectifs de traitement d'images* fournissent la raison d'être du système et justifient son intérêt pour l'accomplissement des objectifs globaux. Puisque la formulation des objectifs ne peut pas se faire par la description des résultats attendus (*cf.* §2.2), nous choisissons de la faire par l'énoncé des tâches à accomplir, qui reste lui toujours possible [Clouard-99]. *e.g.*, *l'objectif de l'application est d'individualiser un noyau dans un cytoplasme pour chaque cellule de séreuse présente sur la lame.*

4.1.2. L'aspect fonctionnel

Sous l'aspect fonctionnel, un système est caractérisé par ses entrées-sorties et par ses relations et interdépendances aux autres systèmes et à l'environnement d'exécution.

- Les *entrées* et les *sorties* ne concernent que les images, qui se réduisent dans ce modèle à de simples fichiers de données, c'est-à-dire uniquement comme une énergie (au sens de [Shannon-49]) qui circule sur le réseau de communication. Seuls leur nombre et leur nature importent. *e.g.*, *l'application prend en entrée une image d'intensité qui présente un étalement de cellules. La sortie est composée de deux cartes de régions : une marquant les noyaux cellulaires et une marquant les cytoplasmes*.

La prise en compte des relations et interdépendances se fait par l'intermédiaire d'un réseau de contraintes portant sur les objectifs. C'est un moyen simple et très efficace [Jolion-01]. Ainsi, le processus de régulation qui assure la stabilité du système face à la variabilité des images de la classe s'exprime par trois types de contraintes portant sur les tâches à accomplir :

- Les *niveaux de détail* fixent les limites hautes et basses de la portée d'une tâche. *e.g.*, *une limite haute de la tâche d'individualisation des noyaux de cellule est de séparer les objets qui se touchent partiellement. Sa limite basse est de ne pas séparer les objets qui se recouvrent (considérés alors comme des amas).*
- Les *critères à optimiser* indiquent les éléments d'une tâche sur lesquels sera portée l'attention. *e.g.*, *puisque le post-traitement inclut des mesures de surface des noyaux, alors un critère à optimiser pour la tâche d'individualisation des noyaux de cellule doit être la localisation des frontières des noyaux.*
- Les *erreurs acceptables* précisent les tolérances sur les deux contraintes précédentes, niveaux de détail et critères à optimiser, et décident des compromis. *e.g.*, *une erreur acceptable pour la séparation des objets est de préférer conserver agrégés des objets qui se touchent partiellement plutôt que séparer des objets qui se recouvrent, afin d'éviter les erreurs de mesure de surface. Une erreur acceptable pour la localisation des objets est de préférer empiéter sur le noyau plutôt que déborder sur le fond de l'image, afin d'éviter les erreurs de mesure d'intensité.*

La boucle de rétroaction qui traduit les exigences du système de post-traitement sur la nature et la qualité des résultats s'explicite par deux types de contraintes portant cette fois sur les résultats à construire :

- Les *éléments à inclure ou à exclure* donnent la composition attendue du résultat final. *e.g.*, *le résultat devra conserver les amas d'objets mais éliminer les globules rouges. Le résultat devra aussi éliminer les noyaux qui touchent le bord puisqu'ils faussent le calcul des caractéristiques, en particulier la taille.*
- Les *règles d'évaluation* décrivent des procédures et des mesures pour valider les résultats. L'application de ces règles d'évaluation doit permettre au mieux de mesurer l'adéquation de l'application aux attentes du client et au pire de détecter les aberrations de résultat. *e.g.*, *la mesure de qualité est estimée de*

manière statistique à partir d'évaluations visuelles faites par plusieurs pathologistes sur la base de deux critères : nombre de cellules oubliées et nombre de cellules mal segmentées.

Enfin, l'action de contrôle qui répercute les impératifs de l'environnement d'exécution se définit par deux types de contraintes portant sur les caractéristiques de la solution à produire :

- Les *critères de performances* expriment des exigences quantitatives sur les ressources à utiliser, le flux d'images à traiter et le temps de traitement. *e.g.*, *le temps de traitement d'une image doit être de l'ordre de la seconde* ;
- Les *critères de qualité* expriment des exigences plutôt qualitatives sur le taux de satisfaction (obtenu par application des règles d'évaluation) et la robustesse du système face à des situations particulières. *e.g.*, *le taux de satisfaction des résultats de la segmentation doit être de l'ordre du taux de comparaison inter-experts.*

4.1.3. L'aspect temporel

Sous l'aspect temporel, un système est envisagé par ses stades d'évolution. Les seules informations conservées ici concernent l'archivage des différents versions (date, numéro de version, auteurs,…).

4.2. Référentiel de modélisation

Les trois aspects du modèle forment le référentiel de modélisation de l'application vue du coté système (Fig. 3). L'écriture du modèle d'un système s'aborde par triangulation entre les trois pôles structurel, fonctionnel et temporel [Le Moigne-77].

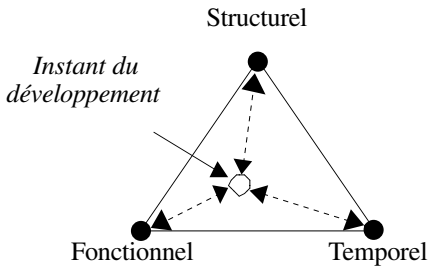

Figure 3. Selon le point de vue « modèle du système », on voit l'appplication au travers du triangle systémique qui définit le référentiel de modélisation de l'application.

Chaque instant du développement vu de ce coté se situe quelque part dans le référentiel selon que l'on se focalise sur la délimitation des frontières de l'application, sur ses interactions ou sur son évolution. Ceci correspond au fait qu'au cours du développement, on utilise ou on enrichit de façon pondérée les informations de ces trois aspects.

4.3. Formalisme de représentation

Le modèle du système d'une application se représente par un document composé d'un formulaire rassemblant les informations issues des trois aspects (*e.g.*, Table 1). Ce document est porteur des objectifs de l'application. La formulation est essentiellement textuelle et est destinée à une double lecture par le concepteur et par le client. Le vocabulaire emprunte à la fois celui du domaine métier et celui du traitement d'images ; il doit refléter l'effort de négociation entre le client et le concepteur.

Table 1. La modélisation d'une application de traitement d'images vue comme un système consiste à renseigner les rubriques identifiées dans les trois aspects : structurel, fonctionnel et temporel.

Aspect	Rubrique	Descripteur	*Exemple*
Structurel	Objectifs	Objectifs globaux	*Aide à la lecture des lames de cytologie des séreuses.*
		Objectifs de l'acquisition	*Marquage des cellules de séreuse par la méthode de coloration de Papanicolaou. Acquisition au microscope.*
		Objectifs des post-traitements	*Classification des catégories de cellules.*
		Objectifs de traitement d'images	*Segmenter les noyaux et les cytoplasmes.*
Fonctionnel	Entrées-sorties	Images d'entrée	*Une image d'étalement de cellules.*
		Images de sortie	*Une carte de régions des noyaux et une carte de régions des cytoplasmes.*
	Régulation	Niveaux de détail	*Individualiser les cellules qui se touchent. Ne pas séparer celles qui se chevauchent.*
		Critères à optimiser	*Localisation des frontières des noyaux.*
		Erreurs acceptables	*Préférer positionner la frontière d'un noyau à l'extérieur du noyau.*
	Rétroaction	Éléments à inclure	*Conserver les amas.*
		Éléments à exclure	*Éliminer les globules rouges.*
		Règles d'évaluation	*Mesure de la différence avec une segmentation manuelle.*
	Contrôle	Critères de performances	*Temps de traitement d'une image inférieur à 1 seconde.*
		Critères de qualité	*Taux de satisfaction proportionnel à la variabilité inter-experts.*
Temporel	Archivage	Numéro de version	*1.0*
		Date de révision	*2000*

5. Le modèle du domaine

5.1. Définition

Le modèle du domaine identifie les informations qui définissent la classe d'images. L'approche sémiotique fournit les bases théoriques pour modéliser la classe d'images, en considérant qu'une image est un système de signes (le signal mesuré) mis pour représenter une chose réelle ou artificielle (la scène ou le phénomène mesuré) [Eco-92], [Joly-94]. Cette approche conduit à modéliser une classe d'images, réduite à sa dénotation, en s'intéressant à trois niveaux de description (Fig. 4) qui couvrent : le signal mesuré (le niveau physique), le rendu visuel (le niveau perceptif) et les objets de la scène ou le phénomène à mesurer (le niveau sémantique) [Clouard-94], [vandenElst-96], [Aubry-01]. La modélisation d'une dénotation consiste en une simple énumération ou description des éléments présentés sur ces trois niveaux à l'aide de descripteurs symboliques ou numériques.

5.1.1. Le niveau physique

Le *niveau physique* décrit toute la chaîne d'acquisition des images, de la mesure du signal à son stockage (*e.g.*, Table 2). Contrairement aux informations composant les objectifs de l'acquisition, les informations collectées ici ne sont pas destinées à reproduire la même classe d'images mais à orienter la conception des solutions notamment dans les étapes de pré-traitements. Pour cela, le schéma de principe d'un système d'acquisition fournit la liste des catégories d'information à renseigner (Fig. 4). Toutes les catégories ne sont pas forcément présentes pour une acquisition donnée (*e.g.*, il n'y pas d'optique pour une acquisition d'images en IRM) et une même catégorie peut cumuler plusieurs dispositifs réels (*e.g.*, dans le cas d'une photographie scannée, le bruit du capteur est la résultante des bruits des deux capteurs : scanner et caméra). Les descripteurs de niveau physique synthétisent donc les conditions d'acquisition (*e.g.*, éclairage, milieu ambiant), le système optique (*e.g.*, zoom, filtre, focalisation, distorsions géométriques), les caractéristiques techniques du capteur (*e.g.*, photométrie, colorimétrie, bruit), le type de numérisation (*e.g.*, quantification, échantillonnage) et le format de stockage (*e.g.*, taille, type de codage).

5.1.2. Le niveau perceptif

Le *niveau perceptif* s'intéresse à la description du rendu visuel des images d'un point de vue global, c'est-à-dire indépendamment des concepts métiers (*e.g.*, Table 3). La modélisation se fait par la caractérisation des primitives visuelles qui composent les images : les contours, les régions, les points d'intérêt, le fond d'image, l'histogramme, la texture ou les zones d'image. La notion de région en particulier est définie par des critères d'homogénéité et non par l'appartenance à un objet. Chaque primitive choisie est alors détaillée par ses caractéristiques géométriques, topologiques, spatiales, photométriques et colorimétriques.

Table 2. La liste des descripteurs ci-dessous donne un extrait d'une description physique de l'application de cytologie.

Catégorie	Descripteur	Valeur
Éclairage	*Illumination*	*Non-uniforme sur la lame ; mais constante entre les images.*
	Modèle d'éclairage	*Image de champ optique vide.*
Optique	*Grossissement*	*(Caméra + Microscope)* $= 20\times$.
	Résolution	*1 pixel représente* $0{,}349\mu\text{m}^2$ *réel.*
	Focalisation	*Flou possible dû à l'épaisseur variable du contenu des lames.*
Capteur	*Type*	*Caméra CCD.*
Numériseur	*Espace couleur*	*RVB.*
Stockage	*Taille de l'image*	512×512 *pixels.*
	Taille d'un pixel	*24 bits.*

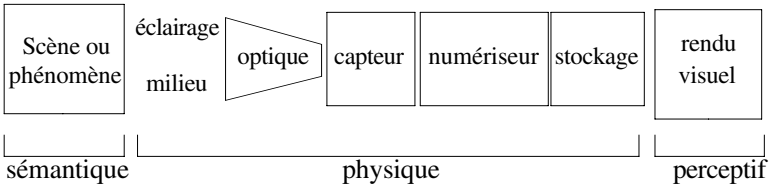

Figure 4. Le schéma de principe d'une acquisition d'images définit les différentes catégories d'informations à renseigner pour formuler un modèle du domaine.

Table 3. La liste des descripteurs ci-dessus donne un extrait d'une description perceptive de l'application de cytologie.

Catégorie	Descripteur	Valeur
Région	*Frontière*	*Contrastée.*
	Couleur	*Bleu et rouge.*
	Taille	*Faible* $< 40 \times 40$ *pixels.*
	Distribution	*Régions dissociées.*
Fond	*Texture*	*Faiblement texturé.*
	Couleur	*Claire.*

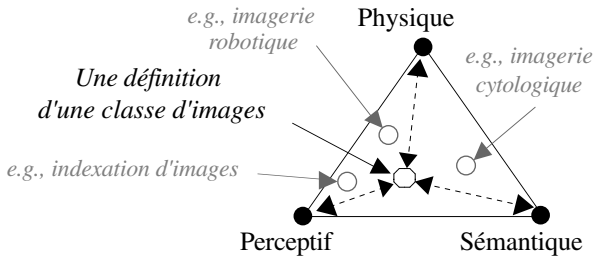

Figure 5. Selon le point de vue « modèle du domaine », on voit l'application au travers du triangle sémiotique qui définit le référentiel de modélisation d'une classe d'images

5.1.3. Le niveau sémantique

Le *niveau sémantique* supporte la définition des concepts du domaine d'application (*e.g.*, Table 4), en particulier ceux utilisés dans la formulation des objectifs. Il a pour but de fournir une description des objets réels et de redonner des informations perdues ou invisibles lors de l'acquisition (*e.g.*, les parties occultées, la troisième dimension). La description des objets est faite en terme de propriétés individuelles (*e.g.*, taille minimale, forme des objets) et de relations entre ces objets (*e.g.*, distribution spatiale, relation de composition, relation temporelle).

5.2. Référentiel de modélisation

L'écriture d'un modèle du domaine s'opère dans le référentiel défini par les trois pôles : physique, perceptif et sémantique (Fig. 5). Pour une dénotation particulière, on se situe quelque part dans le référentiel de modélisation, plus proche du pôle sémantique lorsque l'on est capable de prévoir la plupart des objets dans la scène (cas de l'imagerie cytologique), plus proche du pôle perceptif lorsque les objets sont inconnus (cas d'une banque d'images pour une application d'indexation d'images par le contenu) ou entre les pôles perceptif et physique lorsque les objets sont imprévisibles mais l'acquisition maîtrisée (cas de l'imagerie robotique).

5.3. Formalisme de représentation

Le modèle du domaine d'une application conduit à la rédaction d'un document qui rassemble les tableaux des trois niveaux de description, des fichiers de données, des représentations diagrammatiques et une galerie d'images.

Les tableaux sont composés de descripteurs majoritairement donnés sous forme quantitative ou qualitative (*e.g.*, *taille-image* $= 512 \times 512$, *bruit-acquisition = faible*). Mais quand cela est possible, il est avantageux de donner des descripteurs sous forme d'images (*e.g.*, *le modèle de l'illumination de la scène de cytologie est donné par une image de champ microscopique vide*), de fichiers de données (*e.g.*, la *table des couleurs utilisée pour la normalisation des couleurs*), de loi mathématique (*e.g.*, *la correction gamma =100 %*(intensité/100 %)$^{2.2}$)* voire de dessins faits directement sur les images (*e.g.*, *le pathologiste détoure à la main un exemple de noyau*).

Les diagrammes explicitent des réseaux sémantiques plus ou moins formels les relations d'héritage, de composition et de topologie entre concepts (*e.g.*, Fig. 6).

La galerie d'images regroupe les images représentatives de la classe ainsi que tous les cas particuliers répertoriés ou les contre-exemples instructifs.

Table 4. La liste des descripteurs ci-dessus donne un extrait d'une description sémantique de l'application de cytologie.

Catégorie	Descripteur	Valeur
Cellule de séreuse	*Composition*	*Composée d'un noyau entouré d'un cytoplasme.*
Cytoplasme de séreuse	*Frontière*	*Contrastée, mais certaines transitions ne sont pas assez nettes.*
Noyau de séreuse	*Couleur*	*Bleue : teinte entre* $[\pi + \pi/8, 3\pi/2]$.
	Forme	*Oblongue.*
	Taille	*Petite.*
Globule rouge	*Couleur*	*Couleur rouge : teinte entre* $[0, \pi/3] \cup [3\pi/2, 2\pi]$.
Relation	*Spatial*	*Les noyaux sont répartis sur le fond d'image. Ils peuvent se toucher voire se chevaucher.*

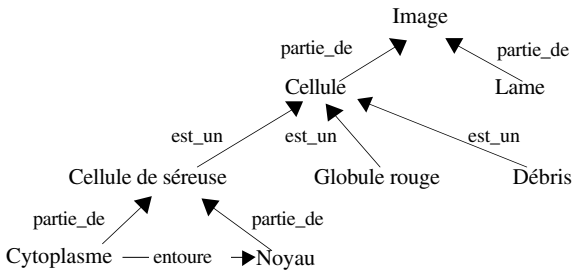

Figure 6. Ce réseau sémantique décrit l'organisation des scènes de cytologie. Les images visualisent des objets cellulaires reposant sur un fond. Les objets cellulaires sont soit des cellules de séreuse soit des globules rouges soit des débris. Une cellule de séreuse se compose d'un noyau entouré d'un cytoplasme.

6. Le modèle des tâches

6.1. Définition

Le modèle des tâches identifie les informations qui définissent les solutions conceptuelles. Ce modèle prend ses fondements d'une part dans les travaux portant sur la modélisation d'expertise par arbres de tâches pour formaliser les solutions [Chandrasekaran-92] et d'autre part dans la logique de conception (Design Rationale) pour capturer et tracer les raisons qui ont conduit à la conception de ces solutions [Buckingham-96]. En conséquence, ce modèle permet d'accéder à deux espaces [Lee-97] : l'espace des solutions qui permet de répondre aux questions sur le comment et l'espace des argumentations qui permet de répondre aux questions sur le pourquoi.

6.1.1. L'espace des solutions

Dans l'espace des solutions, une solution est représentée par un **arbre des tâches** organisé en niveaux d'abstraction. Une tâche de l'arbre encapsule, pour le niveau d'abstraction concerné, une décision de transformation d'images en terme de but de la transformation, contraintes sur le but et listes des images d'entrée et de sortie. L'arbre des tâches renferme éventuellement plusieurs versions de la solution. Quand plusieurs alternatives de décomposition d'une tâche sont possibles, elles sont conservées en parallèle et font l'objet d'un sous-arbre distinct.

L'intérêt d'une organisation de l'arbre en niveaux d'abstraction est qu'elle offre deux visions complémentaires des solutions selon l'axe considéré [Clouard-99] (Fig. 7) :

- Verticalement, l'arbre représente un plan de traitements. Ce plan décrit, en plusieurs niveaux de décision, l'enchaînement des traitements à opérer pour atteindre les objectifs initiaux.
- Horizontalement, l'arbre représente une hiérarchie de solutions de plus en plus détaillées. À chaque niveau, la solution décrit la chaîne complète des traitements pour le niveau d'abstraction.

Nous postulons que l'explicitation des solutions conceptuelles de traitement d'images nécessite exactement quatre niveaux d'abstraction successifs, partant de la définition du problème et débouchant sur son implémentation (Fig. 7). De façon décroissante, nous distinguons les niveaux **objectif**, **directive**, **fonctionnalité** et **algorithme**. De ce fait, le raffinement d'une solution d'un niveau d'abstraction au niveau suivant aborde successivement les décisions d'ordre **politique**, **stratégique**, **tactique** et enfin **technique** :

1. Un **objectif** est un sous-problème que le client a en tête à un instant donné de la résolution. Ce niveau est porteur des concepts et des finalités de l'application et partage ses tâches avec les objectifs de traitement d'images du modèle du systè-

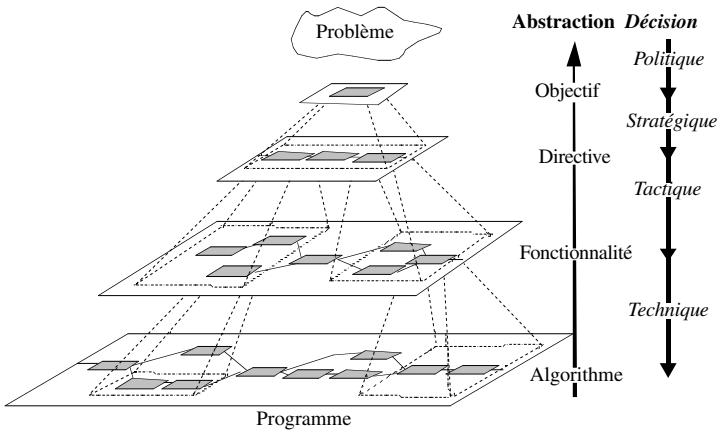

Figure 7. L'arbre de tâches se présente horizontalement comme une hiérarchie de solutions de plus en plus précises et verticalement comme un plan hiérarchique associant des objectifs à une chaîne d'algorithmes adaptée à leur accomplissement

me. Un objectif se formule à partir d'un vocabulaire intentionnel utilisant les concepts métiers : *e.g.*, *séparer les noyaux ou localiser les cytoplasmes*. Il définit les motivations et engendre les contraintes qui peuvent se propager aux niveaux inférieurs. La construction d'un graphe de tâches de ce niveau fait appel aux **politiques** métiers liées à la pratique experte du client sur sa façon de percevoir les informations pertinentes dans les images et d'analyser le contenu des images. *e.g.*, *la politique du pathologiste pour « Individualiser les cytoplasmes » se base sur le fait qu'un cytoplasme ne contient normalement qu'un noyau. La tâche « Séparer les noyaux » doit donc précéder la tâche « Séparer les cytoplasmes »*. Les tâches de ce niveau sont les tâches de l'application que le concepteur doit accomplir.

2. Une **directive** est la spécification d'une étape dans le processus de résolution d'un objectif et s'adresse uniquement au concepteur. Elle se formule à partir d'un vocabulaire intentionnel emprunté cette fois au traitement d'images : *e.g.*, « *Extraire les marqueurs des régions à segmenter* » ou « *Corriger l'illumination des images* ». Elle détient l'ensemble des sous-problèmes à résoudre pour atteindre un objectif. La construction d'une solution de ce niveau se fait par le déploiement de stratégies de traitement d'images qui sont des variations sur les classiques stratégies ascendantes, descendantes ou mixtes. *e.g.*, *la localisation des noyaux de séreuse procède d'une stratégie d'analyse morphologique ascendante adaptée de [Beucher-92] : « Extraire les marqueurs des régions » puis « Construire les régions à partir des marqueurs à l'aide de la Ligne de Partage des Eaux »*.
3. Une **fonctionnalité** est la description abstraite d'une classe d'opérations de traitement d'images telles que « Classification de pixels » ou « Détection de contours ». Les contraintes associées agissent comme des paramètres de la fonctionnalité pour en préciser la portée. *e.g.*, *pour la Classification de pixels, le nombre de classes = 2*. Une solution de ce niveau se construit par application de **tactiques** de traitement d'images. Une tactique est un engagement sur la façon de mettre en oeuvre une directive. *e.g.*, *la tactique de [Coster-85] pour « Séparer des régions convexes » consiste à faire un « Calcul de l'image de distance aux frontières des régions » puis « Extraction des extréma régionaux de la fonction distance » puis « Localisation par ligne de partage des eaux à partir des extréma détectés sur l'inverse de l'image de distance »*.
4. Un **algorithme** décrit la réalisation d'une opération de traitement d'images. Les contraintes associées donnent la façon de paramétrer et de contrôler l'exécution de l'algorithme. La détermination des algorithmes se fait par la mise en place de **techniques** de traitement d'images. *e.g.*, *pour corriger le défaut d'illumination des lames, une technique consiste à faire la division de l'image à traiter par l'image de champ optique sans cellule puis à recadrer les valeurs [Russ-95]*.

Le choix des quatre niveaux se justifie par le principe d'irréductibilité des niveaux [Pylyshyn-84], c'est-à-dire qu'aucun niveau ne peut être confondu dans un autre, et inversement qu'aucun niveau n'en contient d'autres :

- Chaque niveau possède ses propres connaissances qui ne peuvent pas être déterminées ou expliquées par des connaissances des niveaux inférieurs. Ainsi, le niveau objectif est porteur de concepts métiers qui disparaissent au niveau directive. Le niveau directive définit des contraintes sur les tâches qui disparaissent au niveau fonctionnalité parce qu'elles sont intégrées implicitement dans le choix d'une décomposition.
- À l'inverse, les connaissances d'un niveau ne peuvent être déterminées ou expliquées par des connaissances des niveaux supérieurs. D'abord parce que certaines informations deviennent invisibles en remontant dans les niveaux. C'est le cas par exemple des paramètres des algorithmes. Puis parce que le passage d'un niveau aux niveaux inférieurs relève de choix. *e.g.*, *il existe plusieurs fonctionnalités différentes pour réaliser la directive « Extraire les marqueurs des régions » entre autres en passant par « Classification des pixels » ou par « Détection des H-minima de l'image »*.

6.1.2. L'espace des argumentations

Dans l'espace des argumentations, une solution est représentée par le **raisonnement de conception** de la chaîne d'algorithmes de traitement. La logique de conception identifie les informations à représenter dans cet espace pour capitaliser les raisons qui ont conduit l'élaboration de la solution. Par exemple, la notation semi-formelle QOC [MacLean-91] est une notation de la logique de conception basée sur l'argumentation des solutions. Elle présente l'avantage d'être proche de la représentation des solutions par arbre de tâches. QOC manipule trois concepts : une Question (Q) est un but à atteindre, une Option (O) est une alternative de réponse et un Critère justifie (C+) ou infirme (C–) le choix de l'Option comme réponse. À l'aide de ces trois concepts, quatre types de relations sont exprimables qui permettent alors d'accéder au raisonnement de conception : Réponse(Q,O), Soulève(O,Q), Argument pour(O,C), Argument contre(O,C). Une question correspond typiquement au but d'une tâche. Une option correspond à une décomposition d'une tâche en sous-tâches. Par contre, les critères doivent être ajoutés dans la représentation pour avoir accès à l'espace des argumentations. En traitement d'images, ces critères concernent les informations collectées par le modèle du système et par le modèle du domaine :

- des caractéristiques sur les performances de la méthode en terme de coût - efficacité ;
- des contraintes associées au but d'une tâche ;
- des caractéristiques de la classe d'images ;
- des règles ou des procédures qui permettent l'évaluation des alternatives.

La représentation des solutions par arbres de tâches augmentée des critères pour et contre suffit pour accéder au raisonnement de conception et bénéficier de la puissance d'argumentation de la notation QOC.

6.1.3. Référentiel de modélisation

Compte tenu de l'organisation de l'arbre de tâches, l'écriture d'une solution s'aborde par triangulation entre les pôles informations, fonctions et décisions (Fig. 8), selon que l'on se focalise plus particulièrement sur l'analyse du flot d'informations échangées entre les tâches (les images), sur l'analyse des fonctions de traitement (les tâches) ou sur la prise de décisions concernant l'accomplissement des fonctions. Les pôles informations et fonctions envisagent l'arbre des tâches par son axe horizontal et le pôle décisions par son axe vertical.

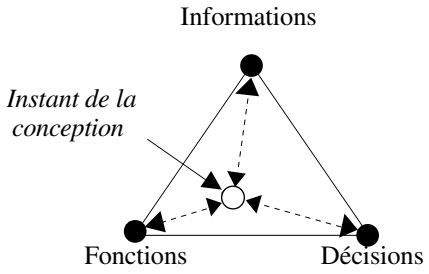

Figure 8. Par le côté « modèle de tâches » du quadriscope, on voit l'application au travers du triangle Informations-Fonctions-Décisions qui définit le référentiel de modélisation d'une solution conceptuelle.

6.1.4 Formalisme de représentation

La représentation d'un modèle de tâches couple l'espace des solutions et l'espace des argumentations dans des formulaires diagrammatiques unifiés où les solutions sont représentées par des arbres de type tâche/méthode (Fig. 9). Une méthode correspond à une alternative de décomposition d'une tâche en sous-tâches et regroupe, dans un formulaire de type IDEF-0 [Mayer-92], un graphe de tâches décrivant l'alternative avec les critères pour et contre argumentant cette alternative. La structure arborescente est réalisée par une numérotation arborescente des formulaires (A0, A1a, A1a1a...).

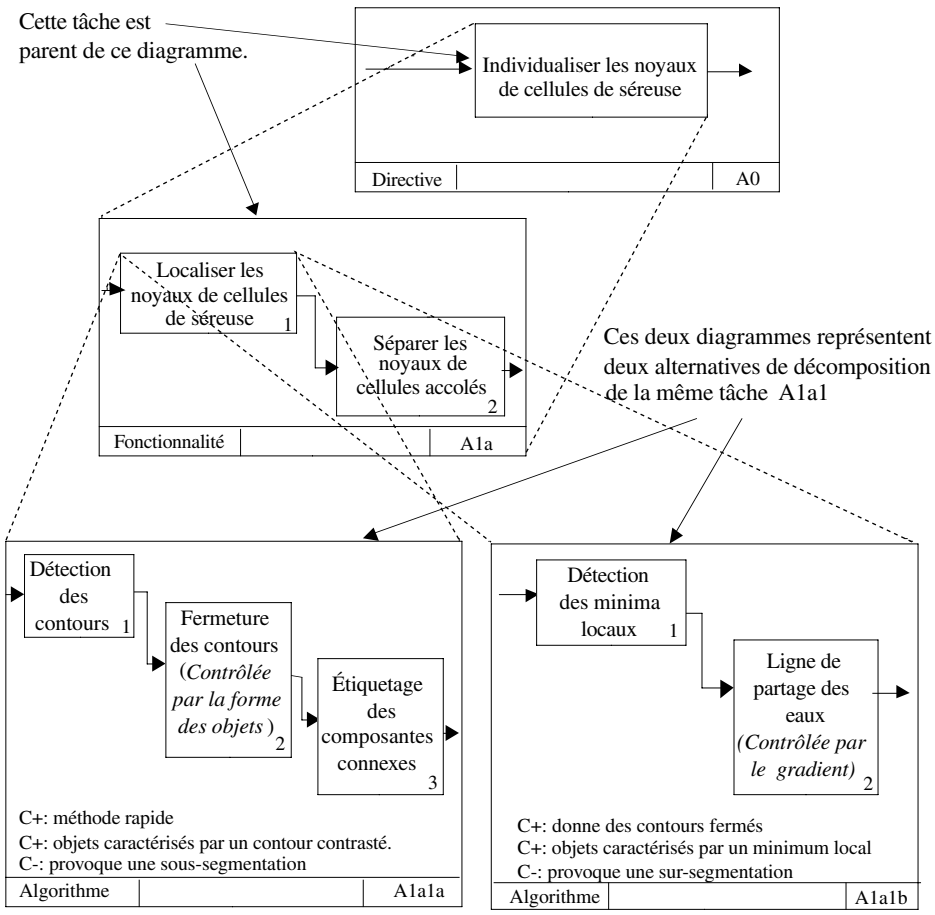

Figure 9. Ce plan propose deux façons de localiser les noyaux des cellules de séreuse. La première utilise la détection des contours puis leur fermeture pour obtenir les frontières des primitives régions. La seconde passe par la détection de germes à l'intérieur des régions par une localisation des régions basée sur une Ligne de Partage des Eaux.

7. Le modèle du programme

7.1. Définition

Ce modèle se fonde entièrement sur la programmation visuelle par flots de données [Blackwell-01], [Hartsough-95]. Un programme est alors représenté par un graphe d'opérateurs. Un opérateur réifie une commande exécutable (ou une fonction d'une API), qui prend en entrée des images et construit en sortie de nouvelles images. Il dispose de paramètres qui permettent d'ajuster son comportement. Les liens entre les opérateurs décrivent le réseau d'images et de valeurs de paramètres échangées entre ces opérateurs. *e.g.*, *la Fig. 11 présente la partie du programme qui permet de localiser les noyaux des cellules à l'intérieur de l'image*.

La validité de l'hypothèse de base selon laquelle n'importe quelle application de traitement d'images peut être représentée par un graphe d'opérateurs paramétrés est conditionnée d'une part à l'utilisation d'une bibliothèque d'opérateurs représentant un bon compromis entre posséder suffisamment d'opérateurs diversifiés et rester limitée à un nombre commensurable [Zamperoni-96], et d'autre part à l'intégration explicite dans le graphe de mécanismes de contrôle évolués pour réaliser l'adaptation à la variété des images [Matsuyama-89].

7.1.1. Opérateurs atomiques

Pour composer de telles bibliothèques, nous arguons qu'il faut restreindre le plus possible les bibliothèques à des **opérateurs atomiques** [Clouard-94]. Un opérateur atomique correspond à un programme sur lequel on peut disposer de toute la connaissance sémantique et syntaxique nécessaire à sa sélection, son paramétrage et à l'appréhension de ses effets sur les images. Il n'y a pas de granularité *a priori* sur le concept d'opérateur, simplement il faut que le problème du contrôle à l'intérieur de l'opérateur soit entièrement résolu. On peut espérer ainsi que plus un opérateur est atomique, plus il est générique et moins il est complexe à paramétrer. Les opérations plus spécialisées s'écrivent alors comme des chaînes d'opérateurs atomiques. *e.g.*, *dans le graphe d'opérateurs Fig. 11, le bloc « Marr-Hildreth edge operator » est une procédure de détection de contours qui s'écrit comme la soustraction de deux lissages d'intensité différente suivie d'une détection des passages par zéro*.

7.1.2. Mécanismes de contrôle

Pour gérer des processus de traitements complexes, T. Matsuyama [Matsuyama-89] propose d'intégrer trois mécanismes de contrôle dans les graphes :

1. **La focalisation d'attention sur des parties d'images**. Elle se réalise facilement par masquage à partir d'une image binaire indiquant les zones d'images à considérer.
2. **Le contrôle d'exécution de la chaîne d'opérateurs**. Il suffit d'ajouter les structures de contrôle classiques : itératives, répétitives et conditionnelles dans la représentation des graphes.
3. **La combinaison spatiale, logique et arithmétique de résultats**. Elle peut s'implanter par des opérateurs de la bibliothèque à part entière, prenant en entrée des images et retournant en sortie de nouvelles images.

Dans ces conditions, les graphes sont des compositions hétérogènes d'opérateurs distinguant différents types de données échangées entre opérateurs : les données image, les paramètres et les variables de contrôle.

7.2. Référentiel de modélisation

La programmation s'aborde par pondération entre les pôles données, opérations et contrôle (Fig. 10). Chaque instant de la programmation se situe dans le triangle défini par ses trois pôles selon que l'on s'intéresse plus particulièrement aux flots de données entre les opérateurs, aux chaînes d'opérations et leur paramétrage ou au contrôle d'exécution des chaînes. Ce référentiel reprend celui du modèle de tâches (Informations-Fonctions-Décisions) (Fig. 8) en considérant un niveau d'abstraction inférieur puisque l'on s'intéresse aux codes exécutables. Les informations se réduisent ici aux données images et masques. Les fonctions sont réalisées par des opérateurs paramétrables et les décisions se réalisent par l'enchaînement des opérateurs à l'aide de structures de contrôle.

7.3. Formalisme de représentation

Un programme se représente en utilisant la notation diagrammatique classique des environnements de programmation visuelle par flots de données (*e.g.*, Fig. 11). Un opérateur ou une structure de contrôle est représenté par un bloc et les flots par

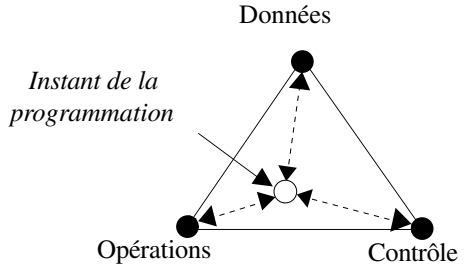

Figure 10. Selon le point de vue « modèle du programme », on voit l'application au travers du triangle Données-Opérations-Contrôle qui définit le référentiel de modélisation d'un programme.

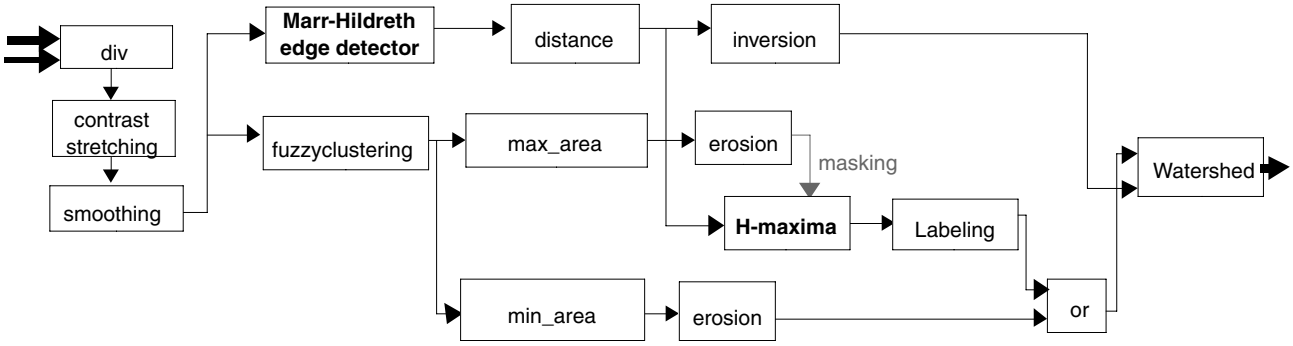

Figure 11. Un programme est un graphe d'exécution d'opérateurs codés comme des programmes exécutables. Le programme ci-dessus permet de localiser les novaux de cellules de séreuse. Il prend en entrée une image de séreuse et une image de champ optique.

des liens entre blocs. Toutefois, nous insistons ici sur la distinction qui doit être faite entre les trois types de liens entre opérateurs : de données, de paramètres et de variables de contrôle. Ceci peut se réaliser simplement par un codage couleur des liens. Le diagramme garde une stricte équivalence avec le code.

8. Conclusion

La modélisation d'applications présentée ici répond à un certain nombre d'objectifs pragmatiques dont le but est d'en faire une modélisation réellement utilisable par les spécialistes de traitement d'images, que ce soit pour l'ingénierie de nouvelles applications ou pour la re-ingénierie d'applications existantes. Mais, elle poursuit aussi des objectifs plus théoriques dont le but est de contribuer à rendre la connaissance du traitement d'images plus accessible, mieux structurée et bien maîtrisée. Dans cette conclusion, nous dressons un bilan en terme d'utilisabilité de la méthode puis nous discutons des perspectives de ce travail.

8.1. Bilan sur l'utilisabilité de la méthode

8.1.1. Ergonomie

L'utilisation de la modélisation est facilitée par le fait que :
- elle nécessite peu d'apprentissage puisqu'elle n'est composée que de quatre modèles qui n'exigent aucune rigueur dans l'écriture des informations ;
- elle est peu intrusive puisqu'elle n'exige du concepteur que de motiver plus ou moins ses choix par des caractéristiques issues de la modélisation des images et des contraintes issues de la modélisation du système. La spécification de l'argumentation associée à une solution est de plus répartie sur le cycle de vie et encouragée par le cycle auteur-lecteurs de la méthode ;
- elle propose plusieurs points d'entrée selon le degré d'investissement souhaité. Chaque modèle peut s'utiliser indépendamment des autres pour ne capitaliser qu'une partie de la sémantique de l'application.

8.1.2. Satisfaction d'utilisation

Par rapport à « pas de méthode du tout », ou l'utilisation d'une méthode classique du génie logiciel, notre modélisation apporte une valeur ajoutée incontestable puisque :
- elle sécurise la programmation du fait qu'il n'y a plus ici de réelle activité de programmation, ce qui accroît la robustesse des logiciels. Les erreurs de programmation sont repoussées au niveau des opérateurs. Mais là encore, la méthode apporte une solution parce qu'elle favorise la réutilisabilité des opérateurs. Ceci permet de les éprouver dans différents contextes ;
- elle prône la simulation comme démarche de développement, permettant d'obtenir rapidement des premiers résultats et de voir progresser l'application ;
- elle offre un support pour la comparaison, la critique, la révision et la réutilisation de solutions par l'encapsulation et l'abstraction de l'expertise de traitement dans un arbre de tâches. Une tâche représente un traitement par ses buts, ses contraintes et ses entrées-sorties tandis qu'elle cache sa mise en oeuvre et l'approche numérique qui le sous-tend ;
- elle capitalise les informations essentielles sur la nature du problème à résoudre et sur la construction des images autorisant ainsi la comparaison et la reproduction de solutions.

8.1.3. Performances pour l'ingénierie d'applications

Cette méthode est une aide au développement d'applications parce qu'elle est un guide formalisateur, complet et rigoureux. Elle répond aux difficultés présentées dans la section 2, parce que :
- elle permet le couplage entre la spécification du problème et la production de solutions mesurables, évitant ainsi la rupture du dialogue entre le concepteur et le client ;
- elle structure le développement en une séquence d'étapes parfaitement identifiées, définissant ainsi une démarche reproductible pour aboutir à des résultats plus fiables. Pour chacu-

ne d'elles, elle précise les informations à renseigner et la façon de les représenter grâce à un référentiel spécifique.
- Elle définit un référentiel d'évaluation en distinguant la validation, de la vérification et des tests logiciel et en organisant les connaissances en granules et en abstraction.

8.1.4. Performances pour la ré-ingénierie d'applications

L'autre apport de la méthode est sa capacité à formaliser *a posteriori* les applications sous une forme complète et explicite. Ainsi, pour le travail de ré-ingénierie d'applications :
- elle permet de faire émerger dans l'application les connaissances profondes, ayant traits à l'expertise de traitement d'images, des connaissances de surface ne relevant que du contrôle ;
- elle permet de dégager ce qui est spécifique de l'application de ce qui est générique du traitement d'images.

8.1.5. Les limites du champ d'application de la méthode

Malgré tout, il existe deux limites inhérentes à notre méthode :
1. Le paradigme de base induit une démarche particulière pour le développement d'applications, qui n'est pas nécessairement partagée par tous les spécialistes. Néanmoins, ce paradigme présente une forte plausibilité cognitive puisqu'il est le reflet de la littérature spécialisée où les techniques de traitement d'images y sont abordées sous l'angle de leur fonctionnalité et de leur enchaînement.
2. L'utilisation d'une représentation informelle des informations empêche la mise en place des mécanismes d'analyse du contenu des modèles qui permettrait de garantir la cohérence, la complétude ou la consistance des informations qui composent une modélisation. Le concepteur reste seul responsable des informations qu'il manipule à l'intérieur des modèles.

8.2. Perspectives

Sur la base de cette méthode, nous développons actuellement un atelier logiciel. Cet atelier se présente comme un éditeur de formulaires élaborés à partir des quatre modèles. À l'intérieur de cet atelier, nous étudions aussi la définition d'une représentation plus formelle des informations qui nous permettra de développer des mécanismes d'analyse du contenu des formulaires à des fins de vérification.

Nous cherchons aussi à travers la définition de cette modélisation à étudier la connaissance de traitement d'images. Nous souhaitons profiter de cette méthode pour contribuer à l'élaboration d'une théorie cognitive du traitement d'images, dont l'objectif serait de rendre les applications mesurables, c'est-à-dire qu'elle permettrait d'expliquer pourquoi et comment les traitements opèrent, de prévoir le comportement de tout ou partie d'une application et espérer ainsi agir sur son comportement de façon déterministe. Dans un premier temps, cela passe par la constitution d'un patrimoine des connaissances du traitement d'images sous la forme de livres des connaissances (*cf.* les travaux de [Ermine-00], [Moisan-00]) et d'ontologies (*cf.* les travaux de [Saidali-02], [Maillot-03]).

Références

[Aubry-01] F. AUBRY, A. TODD-POKROPEK, « Mimos: A description framework for exchanging medical image processing results », Proc. *MEDINFO 2001*, London, pp. 891-895, Sept. 2001.

[Baroth-94] E. BAROTH, C. HARTSOUGH, « Experience report: Visual programming in the real world », in *Visual Object-Oriented Programming: Concepts and Environments*, Burnett, Goldberg & Lewis Eds., Manning Publications Co., Greenwich, Connecticut, pp. 21-42, 1994.

[Beucher-92] S. BEUCHER, « The watershed transformation applied to image segmentation », *Scanning Microscopy International*, Vol. 6, No. , pp. 299-314, 1992.

[Blackwell-01] A.F. BLACKWELL, K. N. WHITLEY, J. GOOD, M. PETRE, « Cognitive factors in Programming with diagrams », *Artificial Intelligence Review, special issue on Thinking with Diagrams*, Vol. 15, No. 1, pp. 95-113, 2001.

[Buckingham-96] S. BUCKINGHAM SHUM, « Design Argumentation as Design Rationale », *The encyclopedia of Computer Science and Technology*, Vol. 35, No. 20, pp. 95-128, 1996.

[Chandrasekaran-92] B. CHANDRASEKARAN, T. R. JOHNSON, J. W. SMITH, « Task-structure analysis for knowledge modelling », *Communications of the ACM*, Vol. 35, No. 9, pp. 124-137, Sept. 1992.

[Chien-96] S.A. CHIEN, H.B. MORTENSEN, « Automating image processing for scientific data analysis of a large image database », *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 18, No. 8, pp. 854-859, Aug. 1996.

[Clément-93] V. CLÉMENT, M. THONNAT, « A knowledge-based approach to integration of image procedures processing », *Computer Vision, Graphics and Image Processing: Image Understanding*, Vol. 57, No. 2, pp. 166-184, Mar. 1993.

[Clouard-94] R. CLOUARD, « *Raisonnement incrémental et opportuniste appliqué à la construction dynamique de plans de traitement d'images* », PhD thesis, Caen, France, Feb. 1994.

[Clouard-99] R. CLOUARD, A. ELMOATAZ, C. PORQUET, M. REVENU, « Borg: A knowledge-based system for automatic generation of image processing programs », *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 21, No. 2, pp. 128-144, Feb. 1999.

[Cocquerez-95] J.P. COCQUEREZ *et al..*, « *Analyse d'images: filtrage et segmentation* », Masson, Paris, 1995.

[Coster-85] M. COSTER, J-L CHERMANT, « *Précis d'analyse d'images* », Presses du CNRS, Paris, 1985.

[Crevier-97] D. CREVIER, R. LEPAGE, « Knowledge-base image understanding systems: a survey », *Computer Vision and Image Understanding*, Vol. 67, No. 2, pp. 161-185, Aug. 1997.

[deRosnay-75] J. DE ROSNAY, « *Le macroscope: vers une vision globale* », Éditions du seuil, Paris, 1975.

[Eco-92] U. ECO, « *Le signe* », Le livre de poche, Paris, 1992.

[Ermine-00] J-L. ERMINE, « *Les systèmes de connaissances (2e edition)* », Hermès, Paris, 2000.

[Garbay-01] C. GARBAY, « Architectures logicielles et contrôle dans les systèmes de vision », in *Les systèmes de vision*, J.M Jolion ed., Hermès, Paris, pp. 197-251, 2001.

[Hartsough-95] C. HARTSOUGH, E. BAROTH, « Visual programming improve communication among the customer, developer and computer », Proc. *National Instruments User Symposium*, Austin, Texas, pp. 26-28, 1995.

[Jackson-83] M. JACKSON, « *System development* », Prentice Hall, Englewood Cliffs, New Jersey, 1983.

[Jolion-01] J-M. JOLION, « Sur la méthodologie de conception de systèmes de vision », in *Les systèmes de vision*, J.M. Jolion ed., Hermès, Paris, pp. 97-131, 2001.

[Joly-94] M. JOLY, « *Image et les signes: approche sémiologique de l'image fixe* », Nathan, Paris, 1994.

[Kruchten-96] P. KRUCHTEN, « A rational development process », *Crosstalk*, Vol. 9, No. 7, pp. 11-16, July 1996.

[Le Moigne-77] J-L. LE MOIGNE, « *La théorie du système général. Théorie de la modélisation* », Presses Universitaires de France, Paris, 1977.

[Lee-97] J. LEE, « Design rationale systems: Understanding the issues », *IEEE Expert*, Vol. 12, No. 3, pp. 78-85, May 1997.

[Lezoray-00] O. LEZORAY, H. CARDOT, « Cooperation of color pixel classification schemes and color watershed: a study for microscopic images », *IEEE Trans. on Image Processing*, Vol. 11, No. 7, pp. 783-789, 2000.

[Liedtke-92] C.E. LIEDTKE, A. BLÖMER, « Architecture of the knowledge-based configuration system for image analysis "Conny" », Proc. *IEEE Int. Conf. on Pattern Recognition*, The Hague, Netherlands, pp. 375-378, Aug. 1992.

[MacLean-91] A. MACLEAN, R.M. YOUNG, V.M.E, BELLOTTI, T.P. MORAN, « Questions, Options and Criteria: Element of Design Space Analysis », *Human-Computer Interaction*, Vol. 6, No. 3, pp. 201-250, 1991.

[Maillot-03] N. MAILLOT, M. THONNAT, A. BOUCHER, « Towards Ontology Based Cognitive Vision », Proc. *Computer Vision Systems (ICVS 2003)*, Graz, Austria, pp. 44-53, 2003.

[Marca-88] D. MARCA, C. MC GOWAN, « *SADT: Structured Analysis Design Technique* », Mc Graw Hill, New York, 1988.

[Matsuyama-89] T. MATSUYAMA, « Expert systems for image processing: Knowledge-based composition of image analysis processes », *Computer Vision, Graphics and Image Processing*, Vol. 48, No. 1, pp. 22-49, Oct. 1989.

[Mayer-92] R. J. MAYER, M. PAINTER, P. DEWITTE., « *IDEF Family of Methods for Concurrent Engineering and Business Re-engineering Applications* », Knowledge Based Systems Inc., College Station, TX, 1992.

[Moisan-00] S. MOISAN, J-L. ERMINE, « Gestion opérationnelle des connaissances sur les codes », Proc. *Journées francophones Ingéniérie des Connaissances (IC'2000)*, Toulouse, France, pp. 131-141, May 2000.

[OMG] Object Management Group, The standards organization that maintains the specification of UML, http://www.uml.org.

[Poggio-85] T. A. POGGIO, V. TORRE, C. KOCH, « Computational vision and regularization theory », *Nature*, Vol. 317, No. 1, pp. 314-319, Sept. 1985.

[Pylyshyn-84] Z.W. PYLYSHYN, « *Computation and Cognition: Towards a Foundation for Cognitive Science* », MIT Press, Cambridge, Mass., 1984.

[Reichgelt-91] H. REICHGELT, « *Knowledge representation: An AI perspective* », Ablex Publishing Corporation, Norwood, New Jersey, 1991.

[Ross-77] D. T. ROSS, «Structured Analysis (SA): a language for communicating ideas », *IEEE Transactions on Software Engineering*, Vol. Se-3, No. 1, pp. 16-34, Jan. 1977.

[Rumbaugh-91] J. E. RUMBAUGH, M. R. BLAHA, W. J. PREMERLANI, F. EDDY, W. E. LORENSEN, « *Object-Oriented Modeling and Design* », Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

[Russ-95] J. C. RUSS, « *The Image Processing Handbook (2nd edition)* », IEEE Press, Raleigh, North California, 1995.

[Saidali-02] Y. SAIDALI, « *Modélisation et acquisition de connaissances : Applications à une plate-forme de traitement d'images* », PhD thesis, Rouen, Dec. 2002.

[Shannon-49] C. E. SHANNON, W. WEAVER, « *A Mathematical theory of communication*», Urbana IL: University of Illinois Press, Princetown, New Jersey, 1949.

[Simon-69] H. A. SIMON, « *The sciences of the artificial* », The MIT Press, Cambridge, Mass., 1969.

[Thonnat-99] M. THONNAT, S. MOISAN, M. CRUBÉZY, « Experience in integrating image processing programs », *Proc. Int. Conf. on Vision Systems*, Las palmas, Spain, pp. 200-215, Jan. 1999.

[vandenElst-96] J. VAN DEN ELST, « *Knowledge modeling for program supervision in image processing* », PhD thesis, France, Oct. 1996.

[Whitley-01] K. N. WHITLEY, A.F. BLACKWELL, « Visual Programming in the Wild: A survey of LabVIEW programmers », *Journal of Visual Languages and Computing*, Vol. 12, No. 4, pp. 435-472, 2001.

[Zamperoni-96] P. ZAMPERONI, « Plus ça va, moins ça va », *Pattern Recognition Letters*, Vol. 17, No. 7, pp. 671-677, June 1996.

Régis **Clouard**

Régis Clouard est maître de conférences de l'école d'ingénieurs ENSICAEN depuis 1995. Ses travaux de recherche au sein de l'équipe Image du GREYC (UMR 6072) portent sur la modélisation des connaissances en traitement d'images en vue de leur capitalisation et de leur opérationnalisation dans des systèmes à base de connaissances.