MASTER EN SCIENCES INFORMATIQUES

Télémicroscopie virtuelle
application au contrôle externe de la qualité

De Nizza, Damien

Award date:
2009

FUNDP
Institut d'Informatique
Rue Grandgagnage 21,
5000 NAMUR (Belgique)

UCL Mont-Godinne
Cliniques Universitaires
Avenue Dr. Gaston Thérasse, 1
B-5530 Yvoir (Belgique)

Télémicroscopie Virtuelle : Application au contrôle externe de la qualité

De Nizza Damien

Promoteur : Jean-Paul Leclercq
Co-promoteur : Hubert Meurisse

Année Académique 2008-2009

Mémoire présenté en vue de l'obtention du grade de Maître en informatique

RÉSUMÉ

Depuis les années 90, les technologies de l'information et de la communication ont connu un développement sans cesse croissant qui les a porté dans la plupart des secteurs. La médecine a, elle aussi, surfé sur cette vague et a donné naissance à la discipline de télémédecine qui vise à appliquer les télécommunications aux pratiques médicales. La télémicroscopie peut être vue comme l'application de la télémédecine au domaine de la microscopie. Par la numérisation et la mise à disposition en ligne des images, elle incite à la décentralisation du processus d'élaboration du diagnostique et favorise la collaboration entre experts. L'approche *store and forward* de cette discipline coïncide avec les principes de l'évaluation externe de la qualité : la sélection d'images représentatives d'un échantillon, leur mise à disposition à destination de personnes extérieures et leur consultation asynchrone.

Ce mémoire propose d'envisager la télémicroscopie virtuelle à travers la modélisation d'une plate-forme générale de composition de contenus multimédias utilisables dans le cadre des évaluations externes de qualité. L'objectif de cette plate-forme étant de pouvoir être indépendante des domaines (hématologie, histologie, etc), elle doit reposer sur un certain niveau d'abstraction et pouvoir adapter les contenus en fonction des contextes de visualisation. Le modèle proposé utilise la technologie XML pour représenter logiquement les contenus de manière à séparer le fond des formes qu'il peut prendre. Ce modèle s'appuie sur une base abstraite générale extensible en fonction des besoins informationnels des domaines. Il intègre des mécanismes de transformation et d'adaptation de l'information en fonction des contextes de visualisation. Il inclut une gestion des résultats qui permet des configurations complexes de questions, utilise des *templates* pour masquer la complexité générale et fait recours à un réseau pour permettre la coopération entre experts.

MOTS-CLES : Télémédecine, télémicroscopie, microscope virtuel, étalement sanguin, étalement médullaire, xml, composition multimédia, évaluation externe de la qualité, EEQ

ABSTRACT

Since the 90's, the technologies of information and communication have known a constantly increasing development that brought them in most of the fields. The health cares have surfed on that wave too and gave birth to the discipline of telemedecine which aims at the application of telecommunications to the medecine practices. The telemicroscopy can be considered as an application of the telemedecine to the microscopy. Thanks to the scanning of samples and the availability of online images, that discipline incites to the decentralization of the diagnosis process and promote collaboration between experts. Its « store and forward » approach matches the principles of the external quality assessment : representative image selection, their provision online and their asynchroneous consultation.

This master thesis considers the virtual telemicroscopy throught the modelling of a general composition platform of multimedia contents for the external quality assessment. The objective of such a platform is to be domain independant (haematology, histology, etc), then it has to rely on a certain level of abstraction and must be abble to adapt the contents according to the viewing contexts. The suggested model proposes the use of the XML technology to represent logically the contents in order to separate the information from its possible forms. It relies on common abstract base which is extensible given the informational needs of the domains. It contains content transformation and adaptation mecanisms according to the viewing contexts. It includes a results managament which allows complex questions configurations, uses templates to hide the general complexity and rely on a network to allow cooperation between experts.

KEYWORDS : Telemedecine, telemicroscopy, virtual microscopy, blood smear, bone marrow, xml, multimedia composition, authoring tool, external quality assessment, EQA

REMERCIEMENTS

Par la présente, je tiens à remercier sincèrement les personnes qui ont contribué à la réalisation de ce mémoire.

Tout d'abord, je tiens à remercier monsieur **Jean-Paul Leclerq** qui a accepté d'être mon promoteur et m'a permis de travailler sur ce projet passionnant de microscopie virtuelle. Ses conseils avisés et sa disponibilité ont été une aide précieuse.

Je tiens également à remercier monsieur **Hubert Meurisse**, mon co-promoteur, qui m'a accompagné sur le terrain durant toute la durée du stage. Dès le départ, il m'a permis de prendre mes repères au sein de l'hôpital. Les nombreuses discussions que nous avons eu ont grandement contribué à la réalisation de ce mémoire.

Mes remerciements vont aussi au Docteur **Bernard Chatelain** qui a montré un intérêt pour mes réalisations et m'a accordé sa confiance en utilisant certains de mes slides dans des séminaires et en me demandant de réaliser une présentation de mes travaux pour un acteur extérieur.

Naturellement, je me dois de saluer mon collègue de bureau, monsieur **Yvan Cornet**. Bien que souvent occupé, il a toujours prit le temps nécessaire pour répondre à mes nombreuses questions. Je tiens aussi à le remercier pour son aide dans le domptage du microscope et sa bonne humeur.

Je suis également reconnaissant envers **François Mullier** pour sa disponibilité et son aide concernant les questions relatives à l'hématologie ou encore les contrôles externes (national et Européen).

Je remercie au passage messieurs **Cédric Peeters** et **Xavier Bocken** pour les réponses détaillées qu'ils ont apporté à mes interrogations. Leur participation m'a grandement aidé à saisir les subtilités du projet en début de stage.

Enfin, je souhaite remercier sincèrement les membres de ma famille pour m'avoir soutenu indéfailliblement lors de mes études. Leur présence a su être une source de réconfort et d'encouragement.

TABLE DES MATIÈRES

Index des illustrations

INTRODUCTION

INTRODUCTION

De manière générale, les technologies de l'information et de la communication mettent en œuvre des techniques qui favorisent le traitement à distance et la transmission des informations. Depuis les années 90, elles ont connu un développement sans cesse croissant qui les a porté dans la plupart des secteurs. La médecine a, elle aussi, surfé sur cette vague et a donné naissance à la discipline de télémédecine qui vise à appliquer les télécommunications aux pratiques médicales. De cette combinaison sont apparues un vaste ensemble d'applications centrées autour de la téléformation, la téléexpertise ou encore la télésurveillance. La télémicroscopie peut être vue comme l'application de la télémédecine au domaine de la microscopie. A travers la numérisation des échantillons et leur mise à disposition sur un réseau, elle favorise la collaboration entre expert en vue d'enrichir l'information liée aux diagnostiques et la pertinence de ceux-ci. La télémicroscopie virtuelle se décompose en deux approches classiques. L'une, dite « *real time* », vise à offrir le contrôle total d'un microscope électronique à un expert, ce qui lui permet une visualisation temps réel à distance. L'autre, dite « *store and forward* », consiste à sélectionner des images caractéristiques d'un échantillon et de les mettre ensuite à disposition via un réseau. Cette seconde approche trouve une application directe dans l'évaluation externe de la qualité qui étudie la pertinence des diagnostiques déterminés sur la base des analyses réalisées sur des échantillons numériques.

Ce travail s'inscrit dans le cadre de la télémicroscopie virtuelle appliquée à l'évaluation externe de la qualité. Les aspects sur lesquels il porte sont doubles. D'une part, il concerne un aspect technique lié à la microscopie virtuelle à travers l'acquisition et la visualisation d'échantillons numériques. D'autre part, il concerne un aspect centré sur l'organisation et la présentation de l'information ainsi que la coopération lors du processus de composition multimédia. Dans le cadre de ma dernière année de maîtrise en informatique aux Facultés Universitaires Notre Dame de la Paix à Namur (FUNDP), j'ai eu l'occasion de réaliser un stage sur ce sujet au sein du laboratoire d'hématologie de l'hôpital universitaires UCL Mont-Godinne. Cette affectation m'a permis de disposer de la plate-forme de microscopie virtuelle disponible sur place. Celle-ci est le fruit des développements réalisés par d'autres étudiants des FUNDP et de l'IESN.

Ce mémoire vise à étudier l'élaboration d'une plate-forme générale de composition de contenus multimédia utilisables dans le cadre d'évaluations externes de la qualité. Par « générale », il faut comprendre « non limitée à un seul domaine ». Il a donc fallu mener une réflexion sur un modèle de représentation logique de l'information qui soit adaptable et présentable différemment en fonction des contextes de visualisation. Durant mon stage, étant localisé dans un laboratoire

d'hématologie équipé d'une station de microscopie virtuelle de type *store and forward* (ce qui cadre bien avec les principes de l'évaluation externe), j'ai pu concrétiser la réflexion en appliquant le modèle au domaine de l'hématologie. Afin de pouvoir générer les images numériques et permettre la composition des contenus, il a été nécessaire de réaliser certains développements sur la station de microscopie virtuelle. Ceux-ci s'inscrivent dans une perspective d'amélioration du système en place qui fait suite aux développements réalisés par les précédents étudiants. Cette perspective est indépendante du principe de l'évaluation externe et concerne des améliorations propres à la station du laboratoire.

La première partie de ce mémoire pose le contexte et aborde des considérations générales concernant la télémédecine et la télémicroscopie. Enfin, elle présente deux cas concrets de contrôle externes : le contrôle national Belge de la qualité en hématologie et son équivalent européen.

La seconde partie présente, dans un premier temps, la méthode adoptée pour organiser le travail sensé mener à la réalisation de l'objectif général d'élaboration d'une plate-forme de composition multimédia pour l'évaluation externe de la qualité. Elle décrit ensuite l'ensemble des objectifs potentiels qui ont été évoqués durant le stage. Ceux-ci sont répartis en deux groupes : un objectif général de modélisation d'une plate-forme de composition multimédia et plusieurs objectifs techniques liés à l'amélioration et l'adaptation de la station de microscopie virtuelle à une nouvelle caméra numérique (pilotage de la caméra, pilotage du microscope, adaptation du système de coregistration et mise au point d'une solution d'*auto-focus*). Le choix de ces objectifs a été opéré sur base de l'importance des besoins qu'ils couvrent. S'ensuit une section dédiée à la présentation de l'infrastructure de microscopie disponible au sein du laboratoire et une description de l'historique du projet en mettant en lumière les apports des étudiants ayant participé au proje. Pour clore, elle propose une description des principes généraux du projet EMIM ([emim]) qui traite de la représentation logique de l'information et de la composition multimédia coopérative en milieu hospitalier.

La troisième partie présente les analyses effectuées et les résultats obtenus. Elle se divise en deux parties distinctes : l'une portant sur les objectifs techniques (liés à la microscopie virtuelle) et l'autre sur l'objectif de modélisation de la plate-forme de composition. Dans la première partie, elle décrit la nature et l'importance des problèmes techniques rencontrés, ce qui a incité au développement d'une nouvelle application de pilotage du microscope. Les sections qui suivent réalisent une description de cette application et terminent par une description de l'impact de celle-ci sur les objectifs fixés au départ. La seconde partie décrit la manière dont a été élaborée la modélisation de la plate-forme générale de

composition média en évoquant notamment la représentation logique des contenus, les spécifications auxquelles elle doit répondre, le niveau d'abstraction qu'elle doit prendre en compte, la manière selon laquelle elle peut intégrer les spécificités des différents domaines, la manière d'adapter la présentation des contenus selon les contextes de visualisation, la manière de gérer les résultats ou de masquer la complexité aux utilisateurs.

Enfin, la quatrième partie amène une discussion sur les résultats obtenus et présente une vue synthétisée des développements futurs qu'il est possible d'apporter au projet.

PREMIÈRE PARTIE : LE CONTEXTE

1. PREMIÈRE PARTIE : LE CONTEXTE

1.1. L'HÉMATOLOGIE

L'hématologie est une branche de la médecine qui a pour objet l'étude des cellules sanguines. Les échantillons de sang (ou de moelle) à observer sont généralement fixés sur une lame porte-objet par étalement. L'étude de ces échantillons est réalisée par un analyste dont l'objectif est de parcourir l'ensemble de l'étalement et de détecter les anomalies sur la base desquelles il pourra alors déterminer une éventuelle pathologie. Les analyses cyto-hématologiques s'inscrivent dans le processus de détermination du diagnostic général d'un patient.

Les composants qui sont analysés dans les échantillons sont des trois types :

- Les globules rouges (ou érythrocytes) sont des cellules dont le cytoplasme est riche en hémoglobine et dont le rôle est de transporter l'oxygène jusqu'aux cellules du corps. Un sujet sain contient approximativement 5 millions de globules rouges par mm^3 de sang
- Les globules blancs (ou lymphocytes) sont des cellules qui interviennent dans le fonctionnement du système immunitaire. Elles se décomposent en plusieurs classe ayant chacune un rôle bien précis dans la défense du corps. Un adulte sain comporte normalement entre 4 milliards et 11 milliards de globules blancs par litre de sang
- Les plaquettes (ou thrombocytes), sont des cellules du sang formés dans la moelle osseuse mais qui se fragmentent directement, ce ne sont pas des cellules complètes mais des petits amas qui servent à la coagulation du sang
- Le plasma est le liquide dans lequel les cellules sanguines sont en suspension et qui constitue 55% du volume total du sang.

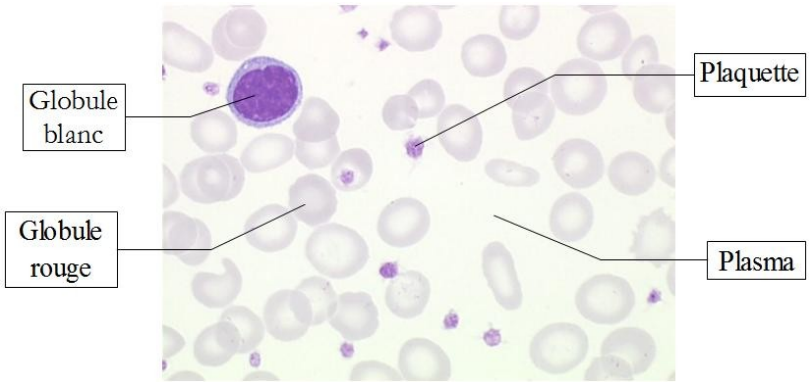

Illustration 1: Illustration des principaux éléments qui composent le sang

1.2. LE TÉLÉMÉDECINE

La télémédecine fait référence à l'ensemble des pratiques médicales qui peuvent être servies par les technologies de l'information et de la communication. L'utilisation de ces technologie amène la possibilité de consultations à distance et de transmissions des informations médicales via un réseaux. Le développement de la télémédecine a connu une croissance fulgurante depuis 10 ans. Cette croissance est liée au développement de NTIC et plus particulièrement à l'amélioration des technologies de stockage de l'information, à l'amélioration de la fiabilité du matériel, à l'amélioration des performances liées aux réseaux et à la démocratisation des coûts. La télémédecine représente donc un ensemble vaste qui peut se décomposer en plusieurs spécialités (tirées de [wik01]) :

- Télé-Consultation : consultation, diagnostic et suivi du patient à distance
- Télé-Expertise : demande d'un deuxième avis à un médecin référent (Médecin Expert)
- Télé-Assistance à domicile : télé-alarme pour personnes âgées, femmes enceintes, handicapés, etc.
- Télé-Assistance des voyageurs isolés : nautisme, montagne, trekking, etc.
- Télé-Surveillance : surveillance d'un patient à distance
- Télé-Formation : formation et enseignement médical à distance (*e-learning*)
- Télé-Médico-Social : encadrement d'un patient maintenu à domicile
- Télé-Transmission : transferts d'informations médicales entre professionnels de santé et patient (Réseaux de soins)
- Télé-Radiologie : interprétation d'examens radiologiques à distance (diagnostic et expertise)
- Télé-Chirurgie : opération chirurgicale à distance
- Télé-Psychiatrie : consultation, diagnostic et suivi d'un patient par un psychiatre
- Télé-Staff : réunion de professionnels de santé en visioconférence.

1.3. LA TÉLÉMICROSCOPIE

Selon [bge02], la télémicroscopie (ou microscopie virtuelle à distance) *est la pratique de la visualisation et du diagnostic, à distance, d'images en provenance d'un microscope. Cette pratique nécessite la visualisation d'images numérisées sur écran vidéo plutôt qu'au travers des oculaires du microscope et également un moyen de communication entre le poste de visualisation et le microscope afin de permettre la transmission des images.*

A travers cette définition, on retrouve à nouveau l'implication des NTIC dans les processus de visualisation et d'élaboration d'un diagnostic. Le fait de numériser les échantillons sous forme d'images permet leur transmission sur le réseau. Il est alors possible de faire intervenir un acteur extérieur pour enrichir les informations médicales déjà récoltées ou demander un avis supplémentaire spécialisé. En effet, grâce à cette mise à disposition des images en ligne, les distances ne sont plus un frein à une coopération plus large. Le but de la télémicroscopie est donc de favoriser les échanges et la collaboration entre experts dans le souci d'avoir des avis complémentaires. La conséquence de ces interactions aboutit sur l'élaboration de diagnostics de meilleure qualité enrichis par des informations médicales très précises et accélérés grâce à l'assistance mutuelle des experts.

En pratique, on distingue principalement deux types de systèmes de microscopie : les systèmes statiques et les systèmes dynamiques. Le premier type se base sur une approche dite « *store and forward* » qui consiste à numériser un échantillon sous forme d'images, à les stocker et à les analyser à postériori. Le second type correspond à une approche dite « *real time* » selon laquelle le pilotage du microscope se fait en temps réel directement par un expert à distance.

1.3.1. L'APPROCHE « STORE AND FORWARD »

Le principe de l'approche *store and forward* consiste, dans un premier temps, à sélectionner une série d'images numérisées à partir d'un échantillon. Cette opération est effectuée par un analyste. Celui-ci organise sa sélection de manière à mettre en évidence un jeu d'images qui, selon lui, représente le mieux la caractéristique à observer. Dans un second temps, ces images sont mise à disposition sur un réseaux via lequel un expert peut les récupérer en vue de poser son diagnostic. Une telle approche permet des consultations « asynchrones » car la visualisation est postérieure au travail de sélection des images.

Cette approche permet également de bénéficier des avantages procurées par la numérisation. En effet, un échantillon réel occupe un certain espace et peut se détériorer avec le temps, ce qui entraine sa perte. Son équivalent numérique est facilement stockable sur un disque ou un CDROM et son état est inaltérable. Un autre avantage de ce type de télémicroscopie est le faible coût de mise en œuvre. En effet, une caméra analogique standard connectée à une carte d'acquisition permet la capture des images et la transmission de celles-ci peut se faire vie un serveur *ftp* ou par mail ([zuy03]).

Cependant, cette technique comporte deux désavantages non négligeables. Premièrement, les images correspondant à la zone d'échantillon mise

en évidence représentent une très petite surface[1]. Dès lors, il convient à l'analyste en charge de la création des images de poser un choix quant à leur sélection. Il y a donc une intervention purement subjective qui est effectuée avant même qu'un expert ne puisse entamer ses analyses et déterminer un diagnostic. Cette subjectivité a un impact direct sur la prise de décision car ce que l'expert voit est ce que l'analyste veut bien lui montrer. Ainsi, pour un même cas, provenant d'un même échantillon, deux analystes différents peuvent donner des jeux d'images qui mènent éventuellement à des diagnostics sensiblement différents, ce qui influence leur fiabilité et leur précision.

1.3.2. L'APPROCHE « REAL TIME »

L'approche *real time* consiste à permettre à un expert de piloter entièrement un microscope électronique et de visualiser le résultat en temps réel à distance. L'objectif est de fournir une marge de manœuvre maximale à l'expert en lui accordant l'accès complet au système de microscopie. Ceci a pour effet d'influencer positivement la qualité de diagnostic.

Cependant, cette méthode connait, elle aussi, certains désavantages. Tout d'abord, avec cette approche, l'expert a directement accès au flux vidéo de la caméra et aucune image n'est donc sauvegardée. Dès lors, on perd les avantages qu'offre la méthode *store and forward* en terme de conservation des échantillons. De plus, il n'est plus possible de réaliser des analyses asynchrones. En effet, selon cette approche, chaque fois qu'un expert veut consulter une lame porte-objet contenant un échantillon, il faut qu'un analyste viennent préalablement la préparer et l'installer sur le plateau motorisé du microscope. Une fois l'installation terminée, l'expert doit faire ses observations sans délais. Ainsi, il dépend non seulement de l'analyste, mais également de la disponibilité du microscope (dont il a besoin pour chaque consultation des images). Enfin, le dernier désavantage de taille est le coût de mise en œuvre d'une telle solution. En effet, pour que le pilotage à distance soit possible, il faut disposer d'un microscope électronique entièrement automatisable. Or, ce genre d'appareil est généralement très onéreux. C'est un investissement qui dépasse largement le déploiement d'une solution *store and forward*. Remarquons qu'il est tout de même possible de mettre en œuvre une solution *real time* et éviter l'achat d'un tel microscope en assignant un analyste au pilotage manuel du microscope sous les ordres d'un expert. Néanmoins, cette solution reste assez contraignante, tant pour l'analyste, qui doit rester disponible pendant toute la durée de l'analyse, que pour l'expert qui assiste à des déplacements qui ne sont pas aussi précis qu'il le souhaiterait.

1 Un problème également rencontré avec le système de télémicroscopie développé au sein du laboratoire d'hématologie de Mont-Godinne. [dec04] a étudié la question et proposé une ébauche de solution

1.4. EVALUATION EXTERNE DE LA QUALITÉ EN HÉMATOLOGIE

Le processus d'évaluation externe de la qualité en hématologie utilise la télémicroscopie de type *store and forward* pour générer une série d'images d'un échantillon et les transmettre aux laboratoires pour analyses. Comme les observations sont asynchrones, les analystes peuvent consulter les image selon leur disponibilité (pour les besoins du contrôle, une échéance est quand même fixée). Les résultats de leurs analyses et le diagnostic qu'ils établissent sont ensuite retransmis pour être évalués. Cette section présente deux cas concrets de contrôle externe de la qualité en cyto-hématologie : l'un opère au niveau national, l'autre au niveau Européen.

1.4.1. LE CONTRÔLE NATIONAL

Les contrôles Belges de la qualité en hématologie sont pris en charge par l'Institut de Santé Publique (ISP). Celui ci organise quatre contrôles par an. Ces contrôles visent à évaluer la qualité des analyses réalisées par les différents laboratoires Belges (dont le nombre approxime les 352).

Les cas qui constituent les contenus sont déterminés par un collège de dix experts. Lorsque ceux-ci rencontrent une pathologie intéressante émanant de leur patients, ils soumettent le cas à l'ISP en fournissant les informations techniques associées ainsi qu'un jeu de 27[2] lames contenant un échantillon de la pathologie observée. Une fois réceptionnés par l'ISP, ces échantillons sont envoyés aux autres experts pour évaluation. Cette fois, aucune information n'est jointe aux échantillon. Le but étant de déterminer, parmi l'ensemble des cas soumis à l'ISP, un cas exploitable et intéressant.

Sur base de leurs évaluations, les experts se réunissent pour élire le cas le plus intéressant pour un contrôle efficace. Une fois l'élection effectuée, ils en rédigent ensemble le contenu. Les lames contenant les échantillons de sang et éventuellement de moelle[3] (si le contrôle spécifie des informations quant à la moelle osseuse) à destination des laboratoires cibles sont réalisées par l'expert dont le cas a été élu. Elles sont alors rapidement envoyées à l'hôpital de Mont- Godinne pour y être conditionnée et ensuite transmises à l'ISP pour stockage (dans l'attente de la réalisation des grands champs et du CDROM qui accompagneront ces échantillons)

2 C'est-à-dire 3 lames pour chacun des 9 autres experts

3 A cause des difficultés de prélèvement de moelle osseuse, seules quelques échantillons médullaires seront envoyées aux laboratoires qui en ont fait la demande explicite

Entre temps, les grands champs et les images des globules blancs de haute résolution sont réalisé à Mont-Godinne (le matériel disponible sur place est celui qui offre la meilleure qualité d'image). Les images générées et le contenu textuel du cas, rédigé en groupe par les experts, sont transmis au développeur qui se chargera de les intégrer à un logiciel *Flash* qui servira à présenter l'information et les images aux laboratoires. Cette application sera embarquée sur un CDROM qui sera joint aux échantillons. Le développeur va ensuite graver autant de CDROM que nécessaire et les transmettre directement à l'ISP.

Enfin, un *bundle* contenant ce CDROM ainsi que les lames est envoyé par colis postal par l'ISP aux différents laboratoires d'hématologie de Belgique. Ces derniers effectuent leurs analyses et transmettent leurs résultats via papier par la poste ou via un formulaire électronique disponible sur le site du contrôle national (si le laboratoire y est inscrit). La gestion des résultats se fait donc au niveau de l'ISP. L'application *flash* n'est pas capable de transmettre de l'information, elle sert de support à l'affichage du contenu des cas et de support à l'impression des résultats.

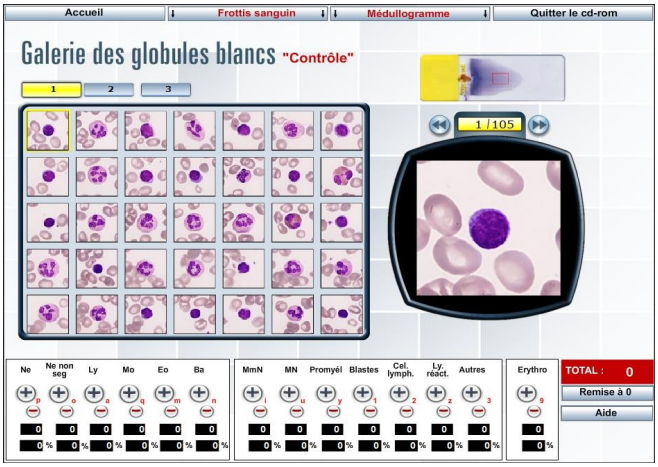

Illustration 2: GUI de l'application Flash du contrôle national (la galerie de globules blancs)

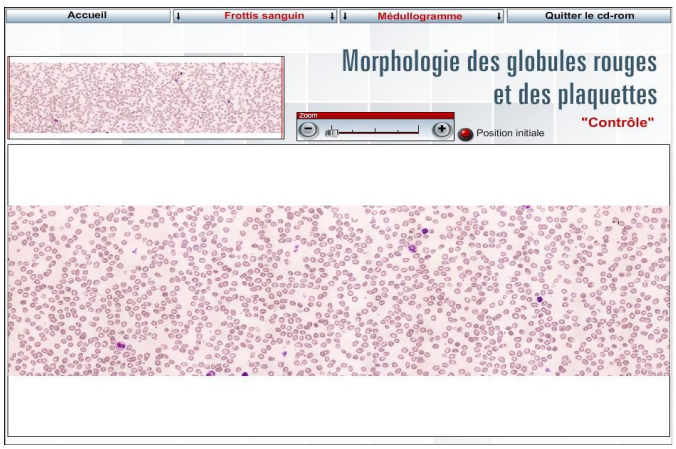

Illustration 3: GUI de l'application Flash du contrôle national (la méga-image de l'échantillon sanguin)

1.4.2. LE CONTRÔLE EUROPÉEN

Le contrôle externe de la qualité en cyto-hématologie au niveau Européen s'étend sur 9 pays : l'Allemagne, les Pays-Bas, la France, l'Italie, la Belgique, la Pologne, la Grande-Bretagne, le Portugal et l'Espagne (voir [hemat]).

C'est en France, et plus précisément à Toulouse[4], que sont coordonnés les différents contrôles à raison de neuf cas par an. Chaque expert d'un des pays participant doit soumettre un ou deux cas à des périodes déterminées réparties sur l'année. Ces cas sont transmis à Toulouse pour évaluation. Un *feedback* est retourné en visant les aspects éducatifs et exploitables du cas. S'il est accepté, il est transmis en Grande-Bretagne pour être traduit en Anglais, puis il est envoyé aux différents experts européen pour acceptation.

Lorsqu'un cas est accepté, l'expert qui en est l'auteur envoie à Toulouse une lame porte-objet contenant un échantillon de la pathologie. C'est là que sont capturées les images et que sont réalisés les *package* renfermant les contenus de contrôle.

Contrairement au contrôle national, ce contrôle Européen s'effectue en ligne à l'aide d'une application dédiés baptisée « *e-HematImage* ». Lorsque l'on exécute le programme, il se synchronise automatiquement avec le serveur Toulousain et rapatrie les contenus de contrôles. Remarquons que, même si plusieurs contrôles peuvent prendre cours en même temps, il n'est pas possible de les choisir, l'ordre est imposé : on ne peut procéder à l'analyse du contrôle *i+1* que lorsque le contrôle *i* a été traité. L'information associée au cas et les images sont présentées dans un jeu d'interface capable de recueillir les réponses de l'utilisateur. Ce denier pourra les transmettre au serveur lorsqu'il jugera avoir terminé son analyse.

L'examen des résultats se fait également à Toulouse. Un *feedback* est alors envoyé aux participants avec les réponses officielles accompagnées d'un commentaire. Depuis le second trimestre 2009, ce système a changé : dorénavant, ce n'est plus Toulouse qui se charge de la transmission des *feedback* à l'ensemble des participants, mais chaque expert qui prend en charge ces envois pour les participants de son pays. Ces *feedbacks* sont de deux types : ils sont accessibles via le programme *e-HematImage* et consultable dans un mail qui est envoyé aux participants. Dans le premier cas, les réponses seront plus succinctes, dans le second cas, les réponses seront plus étoffées et accompagnées de références

4 Ces contrôles sont organisés par l'association FCBM (Formation Continue en Biologie et Médecine)

bibliographiques sur la pathologie analysée.

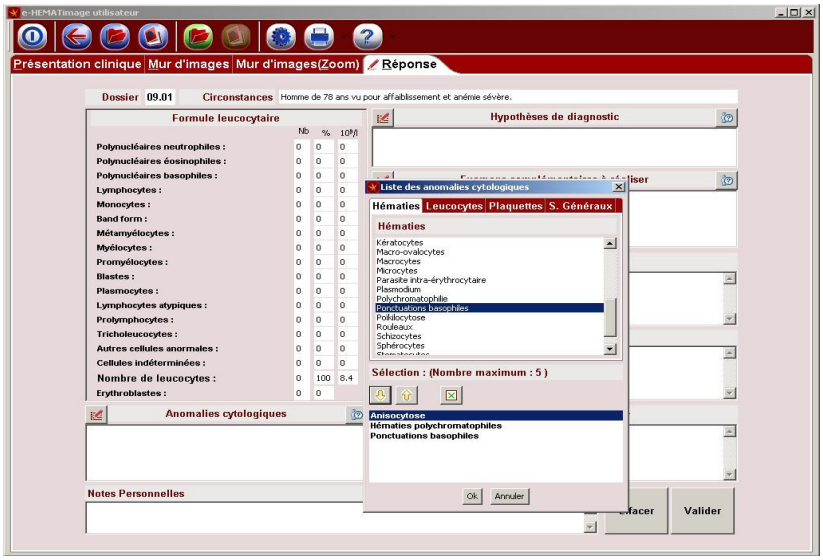

Illustration 4: GUI de l'application e-hematimage (sélection d'une réponse)

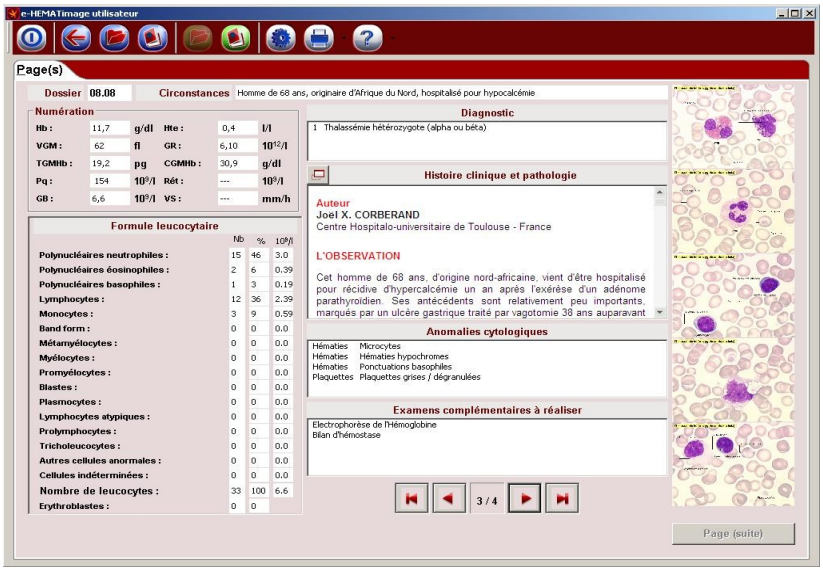

Illustration 5: GUI de l'application e-hematimage (affichage des réponses officielles et du feedback)

SECONDE PARTIE :
MATÉRIEL ET MÉTHODE

2. SECONDE PARTIE : MATÉRIEL ET MÉTHODE

2.1. INTRODUCTION

Cette courte introduction a pour but de résumer l'objectif général qui se dégage de ce mémoire. Dans un premier temps, celui ci s'est décomposé en une série de sous-objectifs dont certains sont apparus en cours de stage. Il a fallu prendre connaissance de chacun d'entre eux, analyser leur faisabilité endéans le temps imparti, tenir compte de leurs importances relatives et déterminer des perspectives de développements coordonnées qui visent un objectif général. La section 2.2 de ce chapitre aborde cette question.

Ce but général (dont fait référence ce mémoire en son titre) est d'envisager la microscopie virtuelle[5] dans le cadre du développement d'une plate-forme de composition de contenus multimétias destinés aux évaluations externes de la qualité d'analyse en cyto-hématologie. Pour ce faire, deux fronts de développements seront considérés : un front technique et un front formel.

Le premier s'appuie sur la microscopie virtuelle[6] qui permet la création des médias qui seront utilisés dans les évaluations en hématologie. Ceux-ci prennent la forme d'images de très grandes tailles[7] numérisées à partir d'un échantillon de sang ou de moelle. Le principe de réalisation des ces images consiste à capturer celle ci par morceaux qui sont ensuite assemblés. Pour parvenir à ces images, plusieurs besoins sont à couvrir et il faut disposer de moyens techniques et informatiques assurant :

- Le pilotage d'un microscope, de manière à pouvoir se déplacer dans une zone de capture
- Le pilotage d'une caméra numérique, afin de numériser un échantillon
- Un protocole de communication pour les interactions entre les appareils et un ordinateur
- Un système d'assemblage de méga-images (pour joindre les morceaux d'image capturés)
- Un moyen de correction des images pour une qualité optimale
- Un format de stockage adéquat (étant donné la taille des images, elles occupent un espace de stockage élevé)

5 Plus précisément, la télémicroscopie selon une approche *store and forward*

6 Qui relève de la numérisation et de la visualisation des échantillons

7 Appelées également « méga-images » qui sont des « *images dont les dimensions sont gigantesques, c'est-à-dire, des images qui nécessitent une grande quantité de ressources pour leur stockage et qui entraînent des délais d'accès importants* » ([zuy03])

Pour répondre à ces conditions et assurer la partie technique, nous utiliserons la station de microscopie virtuelle du laboratoire d'hématologie de l'hôpital UCL Mont-Godinne (voir section 2.3). Celle ci est en effet dédiée à ce genre de travail : elle interagit avec un microscope électronique (via le protocole *RS-232*[8]) sur lequel sont montés un module *d'autofocus* (permettant un réglage optimal de la netteté de l'image) et une caméra numérique, elle dispose d'un serveur de coregistration permettant l'assemblage d'images et stock celles-ci sous le format JPEG-2000. Cependant, certains développements doivent être réalisés sur cette station pour son amélioration et son adaptation au nouveau matériel. La première partie du chapitre 3 porte sur les résultats obtenus suite à ces travaux.

Le second front porte sur la question de la représentation des contenus multimédias utilisés dans les évaluations externes. L'objectif est d'imaginer un modèle qui puissent prendre en charge la représentation des contenus en hématologie, mais également ceux d'autres domaines de manière à rendre la plate-forme de composition la plus générale possible. Pour matérialiser ce objectif, il faut considérer plusieurs choses :

- Un certain niveau d'abstraction qui soit spécialisable aux spécificités des domaines
- Un modèle logique de représentation des contenus de contrôle
- Une séparation des concepts de fond et de forme, de manière à pouvoir spécialiser leur présentation en fonction des contextes de visualisation propres aux exigences des domaines
- Un masquage de la complexité et de l'abstraction à l'utilisateur, de manière à l'aider au maximum et ne pas le perturber dans son travail de composition

La seconde partie du chapitre 3 porte sur l'élaboration d'un modèle répondant à ces spécifications.

8 Norme standardisant un bus de communication de type série

2.2. OBJECTIFS ET CHOIX

Dans un premier temps, les objectifs initiaux qui ont été associés à ce mémoire ont porté principalement sur les développements inscrits dans la continuité directe de la perspective d'amélioration du système de microscopie virtuelle du laboratoire d'hématologie de Mont-Godinne. L'objectif concernant l'évaluation externe de la qualité est arrivé dans un second temps. Cette section commence par la présentation les objectifs d'amélioration du système en place à Mont-Godinne. S'ensuit une description de ce nouvel objectif de contrôle de la qualité d'analyse. Enfin, la section se clôt par une explication du choix qui a été réalisé parmi cette configuration d'objectifs.

Le laboratoire de Mont-Godinne ayant équipé le microscope d'une nouvelle caméra numérique, le premier objectif technique est de trouver un moyen de piloter cette caméra. Il faut savoir que parmi les développements antérieurs qui ont été effectués sur ce projet, une partie s'intègre sous forme de « modules[9] » à une application nommée AnalySIS. C'est à travers cette application que se font les interactions avec le microscope (réglages des objectifs, déplacements du plateau, etc) et avec la caméra. Or, il se fait que l'application ne permet plus de piloter cette nouvelle caméra. Ce problème, qui est en amont de tout le processus de création de méga-images, bloque le projet. Il était donc primordial de pouvoir y apporter une solution.

Le second objectif découle de l'utilisation de cette nouvelle caméra et concerne la coregistration des images. L'ancienne caméra capturait les images avec une résolution fixe de 768 * 573 pixels. La nouvelle caméra travaille avec des résolutions plus grandes. Or, les paramètres utilisés dans les algorithmes développés dans le cadre de la coregistration par L. Zuyderhoff ([zuy03]) sont spécialement conçus pour la résolution fixe de l'ancienne caméra. Des travaux d'adaptation ont été menés dans la perspective d'une amélioration du processus de coregistration pour des images de résolution quelconque. Cette tâche a été entamée par X. Bocken ([boc08]) qui a travaillé sur l'adaptation de ce système. AnalySIS ne reconnaissant plus la caméra[10], c'est grâce à un jeu d'images capturées manuellement à l'aide du *DP-Controller*[11] que cette mise à jour a pu être testée. Les résultats obtenus par X. Bocken au terme de son stage sont bons, mais il reste à intégrer cette nouvelle coregistration au système complet de création de méga-images.

Le troisième objectif assigné concerne un problème récurrent qui hante

9 Codé en *Imaging-C*
10 Le changement de caméra ayant été effectué avant l'arrivée de X. Bocken
11 Application de capture fournie avec la caméra

les mémoires des différents étudiants ayant participé à ce projet, à savoir : l'*aufo-focus*. Le problème vient du fait que, dans une capture automatique d'une série d'images, si l'une d'entre-elles ne représente pas une zone suffisamment contrastée, le module d'*auto-focus* ne pourra régler la netteté de l'image et s'arrêtera en retournant une erreur. Dans ce cas, il faut donc intervenir manuellement pour régler l'image et continuer la capture. Ainsi, dans l'optique d'éviter toute interruption du processus, l'idée est de trouver un moyen de gérer ou de contourner ce problème.

Enfin, le dernier objectif technique est liéà l'analyse d'image et à la classification des globules blancs. Le but est de poursuivre les recherches dans ce domaine sur base des premiers résultats établis par C. Peeters ([pee07]) et X. Bocken. Il s'agit donc de compléter les algorithmes de classification sur la base des paramètres déjà utilisés (principalement des paramètres de forme pour l'instant) et d'en intégrer de nouveaux sous forme de critères mathématiques (pour compléter le panel utilisés par les techniciens pour différencier les types de globules blancs).

Le nouvel objectif apparu dans le courant du stage[12] concerne l'élaboration d'une application permettant de créer des contenus de contrôle dans le cadre des évaluations externes de la qualité en cyto-hématologie organisées par l'Institut de Santé Publique (ISP). Actuellement, la partie technique de ces contrôles est prise en charge à l'hôpital de Mont-Godinne. Le système courant consiste à numériser des échantillons et des informations relatives à un patient virtuel, puis à les intégrer dans une application *standalone* réalisée en *Flash* et enfin, à embarquer cette application sur un CDROM auquel est jointe une lame[13] contenant un échantillon réel du cas. Cependant, plusieurs problèmes sont liés à cette application *Flash*. Premièrement, chaque fois qu'un nouveau contrôle doit être conçu, il faut modifier l'application et recompiler systématiquement un nouveau programme. Deuxièmement, la gestion de cette application est déléguée à un développeur indépendant, il faut donc lui expliquer le plus précisément possible quel est le contenu qui doit figurer dans l'application et comment il doit être présenté. Ainsi, dès qu'il faut apporter la moindre modification au contenu en cours de réalisation, l'hôpital doit passer par ce développeur et lui expliquer soit par mail ou par téléphone les modifications à effectuer. Cette méthode n'est pas efficace tant il est parfois difficile d'expliquer la correction à effectuer. De plus, elle implique une latence dans le traitement qui est due aux échanges entre l'hôpital et le développeur (qui a lui aussi ses contraintes de temps). L'idée est donc de réaliser une nouvelle application permettant aux experts de créer et de gérer eux-même les contenus. De plus, l'*ISP* envisagent, à terme, que la future application (et donc le contrôle) soit

12 Un autre objectif concernant la tomographie a également été mis en lumière, mais il n'a pas été pris en charge dans ce mémoire

13 Eventuellement deux lorsqu'un échantillon de moelle est demandé

également accessible en ligne. Le but étant de faciliter la transmission des contrôles et de se décharger de la lourde tâche de réalisation des lames porte-objet à joindre avec les contenus. Cependant, ce passage *online* devra se faire en douceur. En effet, les différents laboratoires étant évalués sur la pertinence de leurs analyses, ils exigent d'être dans les meilleures conditions pour effectuer leur travail. Or, tous ne sont pas encore habitués aux images numériques. C'est pour cette raison que ces lames sont également transmises avec le CDROM. La future application à réaliser devra donc permettre de créer des contenus embarquables sur CDROM ou disponibles en ligne.

Etant donné le nombre d'objectifs à viser et le temps limité, il n'était pas possible de tous les prendre en compte. La focalisation a été établie sur base de l'importance des besoins qu'ils couvrent individuellement. La couverture des besoins techniques est prioritaire car les opérations qui leurs sont associées sont à la base de la création de méga-images de bonnes qualité et, par conséquent, elles sont aussi à la base de la création des contenus de contrôle. Néanmoins, parmi ceux ci, certains sont plus prioritaires que d'autres. En effet, il est, par exemple, nécessaire de pouvoir à nouveau piloter la caméra pour réaliser des captures d'images composantes qui constitueront le futur grand champ. Cet objectif est donc prioritaire. Une fois ce problème résolu, il sera alors possible d'intégrer le nouveau système de coregistration (pour l'assemblage des images) et de travailler sur l'*auto-focus* (pour un rendu optimal). Ainsi, la maitrise du microscope et de la caméra est indispensable pour les futurs développements, la priorité a donc été accordée à ces objectifs. La classification des globules blancs, quant à elle, a été considérée comme secondaire car elle intervient au niveau du post-traitement qu'il est possible de faire sur les images générées (elle se situe donc en aval de la création des méga-images). Concernant l'évaluation externe de la qualité, la plate-forme de composition de contenus est vraiment nécessaire pour les experts en charge de la réalisation des cas. En effet, le système actuel montrant ses limites, à court ou moyen terme, il faudra mettre en place un système de remplacement. Ce mémoire était l'occasion de démarrer ce nouveau projet et de faire les analyses préalables nécessaires. La stratégie a donc été de considérer la réalisation de la plate-forme de composition comme étant un nouvel objectif général se décomposant en deux parties : l'une couvrant les besoins techniques nécessaires à la création des médias qu'elle utilise, l'autre portant sur les aspects de plus haut niveau liés à la représentation logique des contenus et à leur gestion. Ces deux composantes sont indispensables pour l'élaboration de la future plate-forme. Les travaux qui ont été réalisés les concernant font l'objet de ce mémoire.

2.3. INFRASTRUCTURE EN PLACE À MONT-GODINNE

2.3.1. PRÉSENTATION

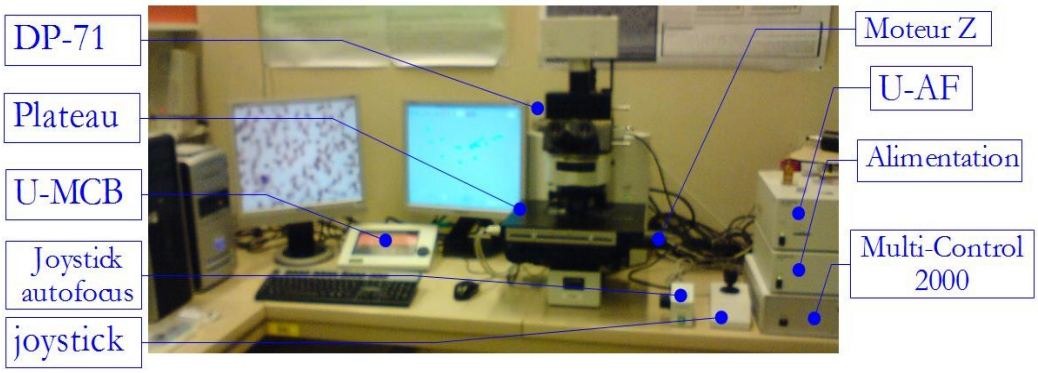

Illustration 6: Le système de microscopie électronique du laboratoire d'hématologie de l'hôpital UCL Mont-Godinne

L'illustration ci dessus montre le système de microscopie électronique utilisé au sein du laboratoire d'hématologie de l'hôpital UCL Mont-Godinne. Les éléments qui le composent sont les suivants :

- Une caméra DP-71 (nouvelle caméra)
- un pupitre de commande « *U-MCB* [14]» qui permet de régler les paramètres du microscope (choix de l'objectif oculaire, de l'intensité lumineuse, du voltage, etc)
- Un *joystick* manuel pour les déplacement du plateau motorisé
- Un autre *joystick* permettant de régler manuellement le *focus* (la netteté) de l'image
- Un moteur indépendant montable sur l'axe Z du microscope
- Un module « *U-AF* » gérant un *auto-focus hardware*
- Un boitier d'alimentation
- Un boitier central « *Multi-Control 2000* » sur lequel sont centralisées les connexions entre les appareils et gérant l'ensemble des requêtes qui sont transmises au microscope

Le système de télémicroscopie qui a été développé au laboratoire et qui utilise ce matériel est baptisé « Microscope Virtuel ». Il est constitué de deux composantes essentielles : l'acquisition et la visualisation. Toutes deux sont basées sur une architecture client-serveur. Cette architecture a été préférée pour répondre à des impératifs de stockage et de performances. En effet, lors de l'acquisition, le processus de coregistration, qui est une opération lourde, requiert beaucoup de ressources pour l'assemblage des images. Le fait de pouvoir déléguer ce genre de

14 Unit Multi Control Board

traitement à un serveur distant dédié plus robuste est préférable. De même pour le stockage et la visualisation des méga-images, il faut une grande capacité en mémoire pour leur conservation ou leur décompression et les clients de visualisation ou d'acquisition ne sont pas forcément équipés en conséquence. De plus, le système se basant sur l'approche « *store and forward* », il faut faire en sorte que les images soient consultables à distance depuis plusieurs postes différents.

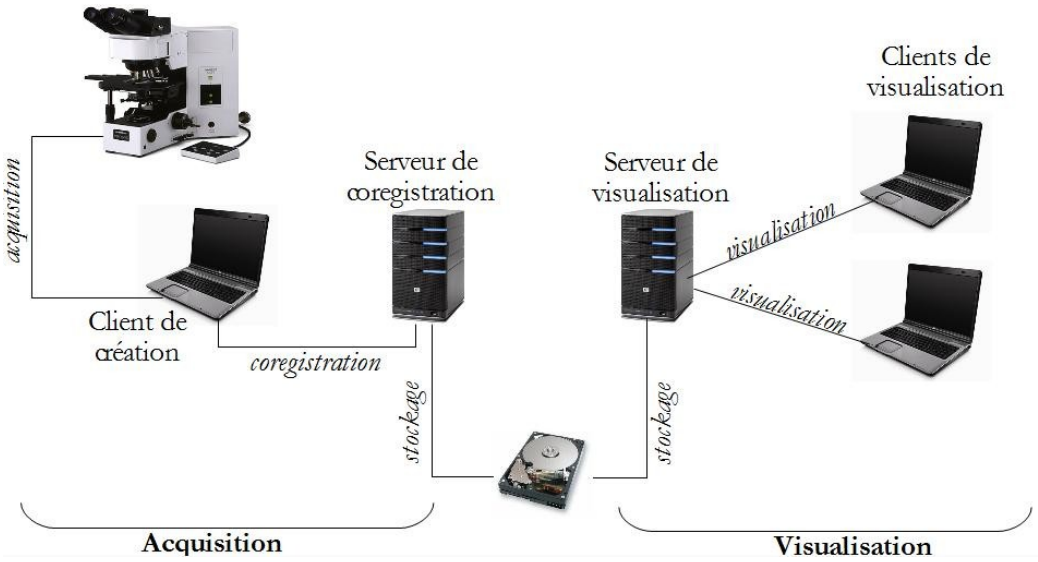

Illustration 7: Architecture générale du microscope virtuel

L'architecture actuelle du microscope virtuel est illustré à la figure 7. Le client de création, qui commande les opérations d'acquisition, interagit avec le microscope et la caméra par l'intermédiaire de modules codés en « *imaging*-C » intégrés au sein du logiciel AnalySIS. Ces modules permettent de commander des déplacements du plateau motorisé et des *snapshots* de la caméra. Les images obtenues sont alors transmises au serveur de coregistration qui va effectuer son travail d'assemblage au gré des réceptions. Etant donné la taille des méga-images composées, le format de stockage qui a été choisi est le standard *JPEG-2000*[15]. En effet, celui ci possède de nombreux avantages qui sont parfaitement adaptés aux besoins exprimés dans le cadre qui nous occupe : performance du processus de compression avec ou sans perte (utilisant la compression par ondelette), possibilité de travailler par régions d'intérêt et possibilité d'affichage des images selon différents niveaux de résolution en sont les avantages principaux. Une fois les images stockées sur un dossiers partagé, il est possible de les visualiser à distance à l'aide d'un client de visualisation qui interrogera le serveur de visualisation pour récupérer les morceaux d'image à afficher. C'est à travers ce client que l'utilisateur peut effectuer les opérations de base qu'un microscope classique offre : choix d'un échantillon (ici, sélection d'une méga-image disponible sur le serveur de

15 Ce standard est expliqué en détail dans [jad05]

visualisation), déplacement sur un échantillon (ici, numérique), changement d'objectif (zoom) ainsi que la sélection de quelques filtres[16] de base permettant d'améliorer les conditions de visualisation de l'image (luminosité, filtres *RGB*). La version la plus récente du client de visualisation est accompagnée de fonctionnalités avancées telles que la création automatique d'une galerie de globules blancs (voir [pee07]).

Ce microscope virtuel ne s'est pas fait d'une traite. Il est le résultat issu des développements de plusieurs mémorands qui se sont succédés au laboratoire. Le projet en lui-même a évolué au fil du temps. La section suivante lui consacre une description de l'évolution suivie.

2.3.2. *EVOLUTION DU PROJET AU FIL DES ANNÉES*

Les travaux de recherche effectués au sein de l'hôpital de Mont-Godinne, et plus précisément au laboratoire d'hématologie, ont débuté en 2001. A l'origine, ces travaux visaient la mise en place d'un système permettant de résumer en une galerie d'images les globules blancs observés sur un échantillon sanguin. Ceci ayant pour but d'offrir un gain de temps non négligeable aux techniciens en charge des analyses. Ces développements ont été pris en charge par B. Georges ([bge02]). Au fil du temps, cet objectif s'est enrichi et le projet a évolué. Des discussions ont amené à repenser le concept : certes, une galerie de globules blancs facilite le comptage ou la comparaison cellule à cellule, mais disposer d'une image numérique d'une surface maximale de la lame permet d'avoir une vue d'ensemble et de prendre connaissance de la distribution et de la morphologie des globules rouges et des plaquettes.

Les travaux qui ont suivi se sont focalisés sur la mise en place d'un système de capture de grand champs (ou méga-images)[17]. Cette création se décompose en deux phases : d'une part la capture[18] des images constituant la future méga-image et d'autre part, la coregistration[19] de ces images visant à les assembler. Comme nous l'avons vu dans la section précédente, cette architecture « client-serveur » a été préférée car les opérations de coregistration sont lourdes et la machine qui est en charge des captures n'est pas forcément très performante. Ceci permet donc de répartir la charge et d'optimiser le travail de création. Par la suite, des améliorations ont été portées à ce système afin de l'optimiser. En effet, la taille de la zone numérisée de l'échantillon était trop faible. Ceci était dû principalement au fait que l'opération de coregistration nécessitait de charger en mémoire toutes les

16 Du moins dans les versions antérieures du *viewer*
17 Voir [zuy03]
18 Effectuée par le client de création
19 Effectuée par le serveur de coregistration (aussi appelé « serveur de création ») qui reçoit les images du client de création

images composantes, ce qui réduisait considérablement la taille de la méga-image résultante. Une tentative d'amélioration a été étudiée par G. Decoq ([dec04]) et a consisté à ne charger que les bords d'images de manière à réduire les ressources nécessaires en mémoire centrale. Cependant, l'amélioration qui a été retenue et offrant les meilleurs résultats est due à l'intervention de L. Zuyderhoff dont les développements ont consisté à faire en sorte que le serveur de coregistration charge les images composantes au fur et à mesure et entame son travail de coregistration pendant que le client de création continue de capturer des images. D'autres optimisations concernant la qualité de la méga-image résultante ont également été apportées au système[20]. Enfin, dans l'optique de minimiser la taille des images finales sans perdre en qualité, le format *JPEG-2000* a été choisi. Au delà des performances que ce standard propose, nous l'avons vu, celui-ci offre notamment la possibilité de faire de la compression avec ou sans perte ainsi que la possibilité travailler par régions d'intérêt, c'est-à-dire qu'il permet de ne charger qu'une région bien précise de l'image et pas son entièreté.

Les opérations décrites ci-dessus permettent la numérisation d'une surface (que l'on souhaite) la plus grande possible d'un échantillon. Afin de pouvoir parcourir virtuellement les méga-images créées, un *viewer* a été développé. Celui ci est également basé sur une architecture « client-serveur ». Le serveur de visualisation[21] (ayant accès à l'espace de stockage où se trouvent les méga-images) traite des requêtes transmises par un client de visualisation et fournit les parties désirées de la méga-image. La décompression de la portion d'image commandée est laissée au client, ce qui améliore la rapidité de transfert (d'autant que plusieurs clients peuvent se connecter simultanément au serveur pour leur transmettre des requêtes). Ce *viewer* a lui aussi fait l'objet d'optimisations prises en charge par C. Jadoule (voir [jad05]), principalement au niveau de la gestion de la mémoire grâce à un système de *caching* et de *prefetching* qui permet d'anticiper les morceaux d'images à charger en fonction du déplacement de l'utilisateur sur la méga-image (typiquement, les régions de l'image adjacentes à la région courante visualisée).

Tous ces développements amènent à la manipulation des images : chargement, *cropping*, conversion de formats, application de filtres, coregistration, etc. Une dimension supplémentaire s'est ajouté à ces traitements : « l'analyse d'image ». Les récents travaux de recherches réalisés sur le projet ont, d'une part, assuré la continuité dans l'amélioration de l'implémentation mais, d'autre part, ont aussi ouvert un nouveau volet dédié à l'analyse des méga-images[22]. La raison

20 Voir [zuy03] pour plus d'informations sur la post-optimisation des méga-images

21 Typiquement une machine aveugle, sans interface, juste un terminal

22 Remarquons que de l'analyse d'image a également été effectuée au tout début du projet (2001) par B. Geroges car, à l'époque, il fallait détecter les globules blancs dans les images transmises par la caméra pour les mettre dans une galerie

principale est que, certes, le système courant permet de numériser une partie d'un échantillon sanguin ou médullaire mais, contrairement à l'idée originelle, il n'offre plus une vision résumée des globules blancs sous la forme d'une galerie d'images. Afin de combler ce manque, une fonctionnalité de « création automatique de galerie » a été ajoutée dans le *viewer* par C. Peeters (voir [pee07]), qui a notamment aussi travaillé sur l'homogénéisation des méga-images. Cette fonction a pour but de parcourir la méga-image, d'en détecter les globules blancs et de réaliser une galerie. Dans cette optique d'analyse d'image, des recherches sur la classification de ces globules blancs ont été menés par C. Peters et X. Bocken[23] (voir [boc08]). Les développements n'en sont encore qu'à leurs débuts mais de bonnes pistes ont été tracées et certains paramètres de classification (principalement la forme et la taille des cellules) ont fait l'objet d'implémentations pour une première classification de base. Ces implémentations sont intégrées au *viewer* et sont exécutées lorsque l'utilisateur lance une « création automatique de galerie ». Le résultat de cette classification se présente sous la forme d'un tableau *Excell* résumant les valeurs des paramètres ayant permis la classification ainsi que l'identification du type des cellules analysées. La galerie créée indique également le nom de certaines cellules ayant été identifiées. Ces résultats serviront de base aux discussions avec les experts afin d'établir des modèles mathématiques qui seront utilisés dans les futures améliorations. ces dimensions d'analyse de l'image et de classification ont permis d'ouvrir une nouvelle voie au projet : la génération d'informations de diagnostic. En effet, dès lors qu'il est possible de classifier les globules blancs, on peut alors imaginer en faire le comptage[24] automatiquement et proposer un premier pré-diagnostic.

Ainsi, le projet a évolué au fil du temps. Au début, son but était de résumer un échantillon sanguin en une galerie de globules blancs. Ensuite, la galerie a été remplacée par une méga-image représentant la numérisation d'une surface de l'échantillon. Pour permettre de parcourir ces grands champs à la manière d'un microscope classique, un *viewer* spécialisé a été développé. Les mémorands se succédant, des améliorations et optimisations ont été apportées aux implémentations. Enfin, l'analyse d'image permet de recréer une galerie de globules blancs à partir d'une méga-image (ce qui renoue avec le but originel du projet) et permet d'envisager une classification automatique des cellules en vue d'une génération de pré-diagnostics. Le microscope virtuel n'est donc plus pensé comme un « simple *viewer* », mais comme un outil capable de déduire de l'information.

23 Qui a également travaillé sur une amélioration des algorithmes de coregistration en adaptant leur traitements à des images de résolutions quelconques

24 Méthode utilisée par les laborantins qui, sur base des statistiques observées suite au comptage des différents types de cellules, permet de déduire des informations importantes sur le patient et de poser un diagnostic

2.4. EMIM2[25]

Le contexte qui a motivé le lancement du projet EMIM ([emim]) vient de l'opposition entre les capacités sans cesse croissantes de numérisation des données matérielles et les limites de l'approche diagnostic classique. Le développement des NTIC a favorisé la dématérialisation des données provenant des plateaux médico-techniques hospitaliers. Les raisons de cet engouement aux nouvelles technologies sont liées aux nombreux avantages que celles-ci apportent en terme de stockage, de conservation sur le long terme et de mise à disposition de l'information à distance. De nos jours, les possibilités du monde numérique permettent de générer une quantité considérable de données en tout genre à propos d'un patient. Bien que ces informations représentent une source riche et précieuse, le volume qu'elles représentent devient difficilement gérable dans son ensemble. Or, la démarche de prise de décision à travers le diagnostic est une opération qui se fait par fragments, à partir d'informations séparées en temps et en espace, ce qui fait obstacle aux prises de décision rapides qui peuvent être cruciales dans certains cas. C'est dans ce contexte qu'intervient la composition multimédia coopérative en support à la démarche diagnostic en se déclinant en trois axes : la composition de l'information multimédia, sa présentation adaptée et la coopération entre les acteurs intervenant dans la prise de décision. Ainsi, la composition de l'information multimédia vise à « enrichir *une collection hétérogène de médias d'un contexte logique et sémantique intégrateur. Elle y ajoute ensuite un ensemble de spécifications de présentation et d'interactivité* » ([arm04]). Il s'agit donc d'organiser les données multimédia selon une organisation logique adaptée dans le but d'enrichir leur expressivité. Pour permettre d'ajouter cette dimension d'expressivité au contenu et permettre les adaptations de présentation adéquates, il convient de séparer le fond des formes qu'il pourrait prendre.

La figure 8 illustre ces considérations à travers une organisation logique de l'information associée à un dossier médical d'un patient. Cette structuration renferme une description des données médicales générales sur le patient, des descriptions (sous forme de « pages ») des données spécifique provenant des différents plateaux hospitaliers, auxquelles sont joints des scénarios de transformation pour adapter leur présentation, et l'ensemble des données multimédia utilisées. Les adaptations de présentation (organisées selon les scénarios et réalisées à l'aide de feuilles de style *xsl*) portent sur la disposition des éléments, leur temporalité ou encore les liens hypermedias qui peuvent exister entre eux.

25 Emission Multimédia des données en provenance de plateaux techniques d'Imagerie Médicale

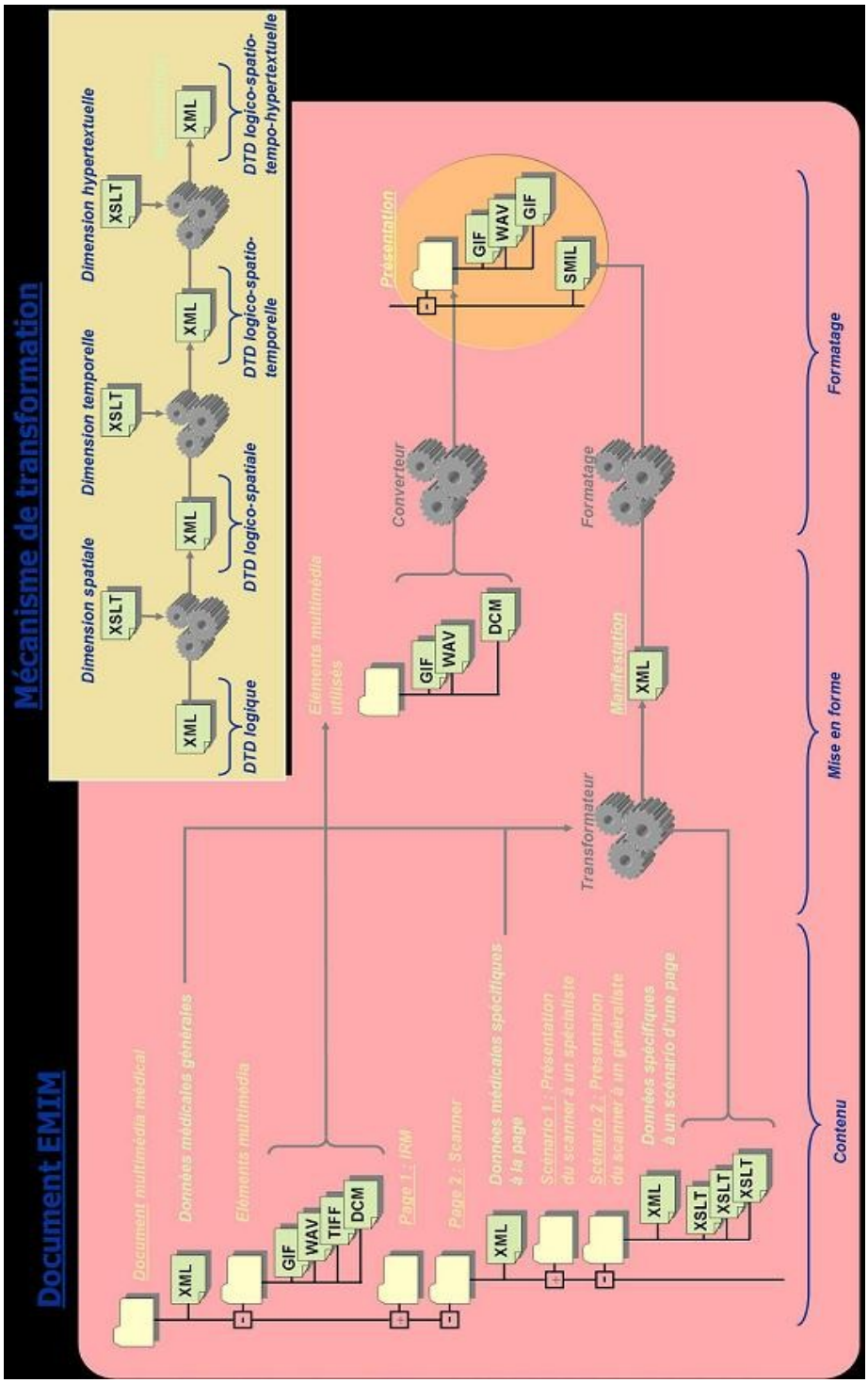

Illustration 8: Structure d'un document EMIM et mécanisme de transformation ([emim])

L'illustration 9 montre les éléments qui constituent l'architecture du système de composition et d'émission des données. La composition des dossiers médicaux organisés logiquement se fait au niveau des « Points d'Entrée Multimédia » (PEM). Ceux-ci sont équipés d'un outil faisant office d'éditeur et de navigateur qui permet la composition et l'édition des pages et des scénarios. Les « auto-PEM » prennent aussi en charge cette fonction de composition mais de

manière automatique : ils extraient les informations médicales contenues dans un un fichier DICOM, les places dans les fichiers *xml* et ajoute des scénarios prédéfinis en fonction du type d'examen médical. Les documents générés sont ensuite mis en partage au niveau du « Relai Multimédia » (RM) qui gère la conservation et la distribution des documents. Ce relai utilise un système de base de donnée relationnelle pour la gestion des données reçues, la gestion des utilisateurs et la gestion des ressources. Enfin, la visualisation des contenus se fait par l'intermédiaire d'une « Borne Multimédia » (BM) à l'aide d'un simple navigateur Internet. Les interactions avec l'utilisateur sont prises en charge par un *Applet* Java qui établit la communication entre le relai multimédia et la borne. Les différentes présentations d'un document sont alors obtenues par l'exécution des scénarios qu'ils contient (et qui sont choisis par l'utilisateur). L'application des transformations et des conversions est prise en charge par le relai multimédia qui placera ensuite les fichiers qui composent la présentation sur un serveur *http*. Le Relai Multimédia transmet ensuite à l'*applet* une *url* à laquelle récupérer la manifestation du document.

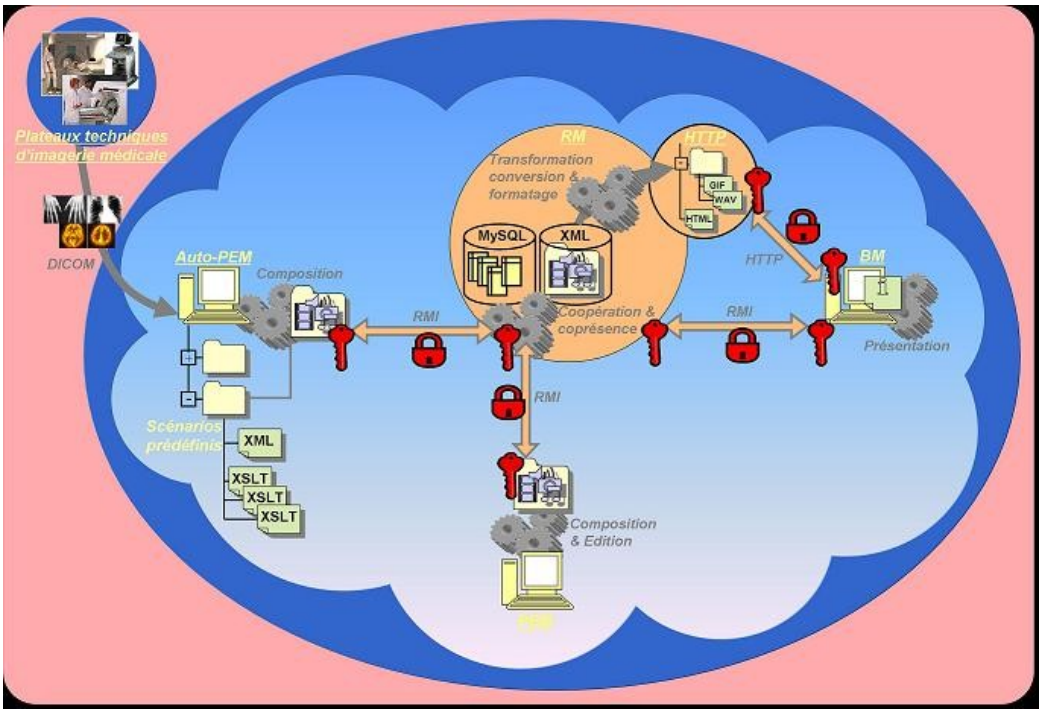

Illustration 9: Architecture du système EMIM mis en place ([emim])

TROISIÈME PARTIE :
ANALYSES ET RÉSULTATS

3. TROISIÈME PARTIE : ANALYSES ET RÉSULTATS

3.1. INTRODUCTION

Comme évoqué au chapitre 2, le principe de l'évaluation externe de la qualité renvoie à la composition de contenus regroupant de l'information multimédia et textuelle. D'une part, ceci implique d'avoir accès aux outils et techniques permettant la création des médias nécessaires. D'autre part, il s'agit également d'envisager les questions de regroupement, d'organisation, de structuration, de stockage et de présentation de l'information. En conséquence, ce chapitre se divise en deux parties principales correspondant aux analyses et résultats obtenus sur le volet technique et sur le volet formel (modélisation).

Le travail effectué sur la partie technique consistait principalement à porter des améliorations sur la station d'acquisition disponible au laboratoire d'hématologie de l'hôpital UCL Mont-Godinne. Plus précisément, cela revenait à envisager des solutions quant au pilotage de la nouvelle caméra à partir du logiciel AnalySis et à intégrer l'adaptation des algorithmes de coregistration au serveur de coregistration. Pour ce faire, dans un premier temps, nous avons procédé à une mise à jour d'AnalySIS qui, jusqu'ici, servait d'interface pour le pilotage du microscope. Malheureusement, cette opération n'a pas rencontré les attentes escomptées. La section 3.2 aborde premièrement les raisons qui ont incité le développement d'une nouvelle application indépendante de pilotage du microscope. S'ensuit une explication des analyses préalables qui ont été effectuées sur l'appareil. Ensuite, les principes du nouveau programme et les spécificités liées à son fonctionnement seront présentées. Enfin, les impacts sur les objectifs déterminés seront abordés. Remarquons que ces travaux ne sont pas directement liés aux principes de l'évaluation externe de la qualité. En effet, la station d'acquisition conserve son indépendance à ce sujet et reste considérée comme un outil général de numérisation d'échantillon. Il ne s'agit pas d'orienter son développement dans le but d'en faire un outil spécifique dédié au contrôle de qualité mais bien d'adapter le système au nouveau matériel et d'en améliorer les services.

La seconde partie de ce chapitre porte sur l'élaboration du modèle de gestion des contenus multimédias. La mise en place de ce modèle (et du système qui devra l'implémenter) correspond au démarrage d'un nouveau projet. Il a donc été nécessaire de procéder à une analyse des besoins (à travers la réalisation d'un cahier de charges) et une analyse de l'existant[26], suite à quoi les travaux de réalisation ont pu commencer. Le chapitre 2 est introduit en précisant que le

26 Voir la section 1.4 pour des cas de contrôle externe concrets

modèle se veut général et indépendant vis-à-vis des domaines de la biologie, dont la visualisation peut être adaptée en fonction du contexte et permettant la collaboration entre experts. Le présent chapitre expose les détails qui permettent au modèle de satisfaire ces caractéristiques en abordant des questions d'abstraction, de structuration logique, de transformation de fichiers, de gestion des résultats et de coopération.

3.2. APPLICATION DE PILOTAGE DU MICROSCOPE

3.2.1. MOTIVATIONS

Afin de rencontrer les objectifs d'amélioration de la plate-forme d'acquisition du laboratoire, plusieurs solutions ont été étudiées et toutes n'étaient pas simple à envisager. Suite à nos discussions avec la société Olympus, une mise à jour complète d'AnalySIS nous a été proposée, ce à quoi nous avons répondu par l'affirmative en commandant un devis ainsi que la possibilité d'évaluer le produit sur une période déterminée. Cet *update* concernait à la fois des mises à jour *software* et *hardware* (les *drivers* de certains composants du microscope électronique). Par cette mise à jour, nous espérions aussi pouvoir apporter une solution quant au problème récurrent de l'*auto-focus*. Ce problème vient du fait que lorsque la zone d'image dans laquelle s'effectue l'analyse de la netteté n'est pas suffisamment contrastée, l'*auto-focus* ne peut se faire et retourne une erreur. Dès lors, dans le cas d'une capture automatique d'images multiples, cette interruption pose problème. La solution idéale sur laquelle plusieurs mémorands ont travaillé consiste à intercepter l'erreur et envoyer une série de commande au module[27] gérant cette opération de mise au point pour récupérer des valeurs (typiquement, la hauteur) qui permettraient de continuer les captures suivantes en repartant sur la dernière hauteur valide (qui correspond à une hauteur à laquelle l'image précédente était nette).

Les résultats de cette mise à jour se sont révélés négatifs. Bien que la caméra soit à nouveau détectée par AnalySIS, le module d'*auto-focus* a continué à retourner des erreurs sur les commandes qui nous intéressaient (cette question sera abordée plus amplement au point 3.2.6.3.). De plus, un nouveau problème est apparu au niveau du pilotage du plateau du microscope. En effet, pour des raisons obscures, les déplacements de l'appareil sur l'axe des abscisses n'étaient plus réguliers et par moments, ils étaient trop courts. Par conséquent, il n'y avait plus de recouvrement systématique entre les images ce qui rendait toute coregistration automatique impossible. En composant la méga-image à la main avec le système intégré de AnalySIS, on voyait très clairement des « trous » apparaître. L'illustration

27 Module U-AF

suivante montre le résultat obtenu sur une méga-image de 4*3 images selon un recouvrement de 150 pixels.

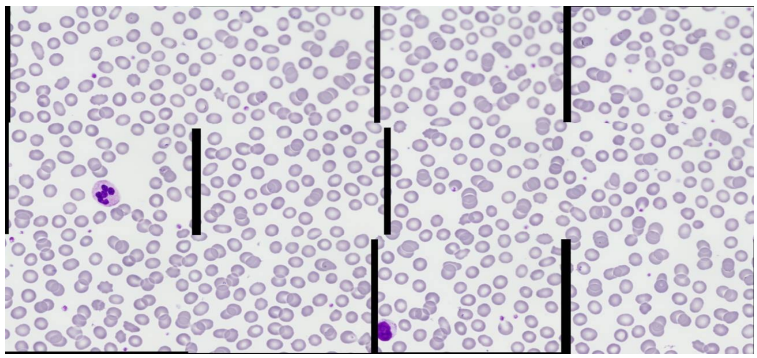

*Illustration 10: Le problème de déplacement du plateau rend impossible toute coregistration automatique (4*3 images, overlap de 150 pixels)*

On constate que seules les deux premières images de chaque rangée horizontale (en considérant le sens en « zig-zag » suivit lors de la capture) sont coregistrables. Toutes les autres présentent un *gap* de taille variable avec l'une de leurs voisines. Ces observations se sont répétées sur chacun des tests réalisés et il a été impossible de trouver la raison direct de ce problème (une défectuosité possible au niveau du *Multi-Control* nous a même été évoquée). Le seul moyen de compenser ces écarts était d'augmenter le recouvrement tel qu'il soit supérieur au *gap* de taille maximale (taille qui est variable est imprévisible). La même zone capturée avec un recouvrement de 300 pixels permettait d'absorber les interstices entre les images.

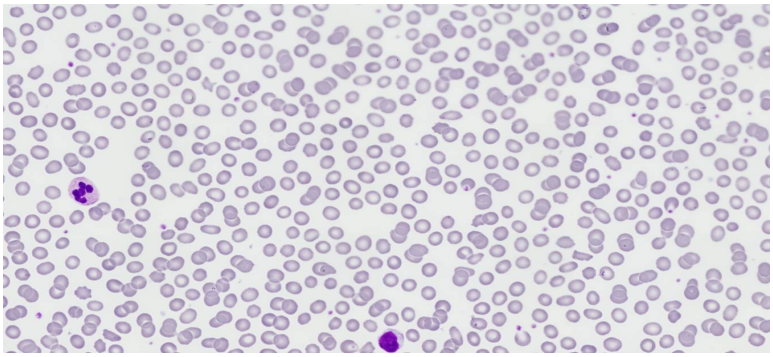

*Illustration 11: Méga-image de taille 4*3, overlap de 300 pixels : les écarts sont absorbés*

Naturellement, cette compensation ne représente pas une solution en soi. Elle ne règle pas le fondement du problème. Or, les déplacements du plateau sont à la base de tout le processus de création des méga-images et donc des traitements que l'on peut effectuer sur elles par la suite. L'impact de ce problème était non négligeable car il bloquait littéralement les avancées sur les autres objectifs.

C'est à la suite de ce constat que, parmi toutes les solutions discutées, la réalisation d'une application Java de pilotage du microscope a été envisagée. Effectivement, des tests de déplacements hors AnalySIS ont démontré que le plateau ou le *Multi-Control* n'étaient pas défectueux et qu'il était possible d'indexer précisément[28] des positions correspondant aux coordonnées des images composantes de la méga-image. Dans l'absolu, disposer d'une telle application pour la station d'acquisition du laboratoire n'est pas sans intérêt. En effet, plusieurs avantages apparaissent :

- Elle représente une alternative à AnalySIS et une nouvelle voie dans les développements futurs
- L'accès au code source est complet, toute la documentation est disponible
- L'application peut être directement modifiée à volonté selon les besoins
- Le fait qu'elle soit codée en Java offre une compatibilité directe avec les précédent développement, ce qui facilitera sont intégration à la station d'acquisition actuelle (en prenant le pas sur AnalySIS) ainsi que l'intégration de futurs modules

De plus, en complément du module d'*auto-focus* (qui, visiblement, posait toujours les mêmes problèmes), le laboratoire est équipé d'un moteur indépendant haute résolution que l'on peut monter sur l'axe Z du microscope. Dès lors, ce nouveau développement représenterait une occasion concrète d'effectuer des tests plus poussés avec ce moteur, notamment en vue de mettre en place un système *software* de réglage du *focus*.

Etant donné que la situation ne semblait pas se débloquer, nous avons validé cette option. Néanmoins, pour pouvoir mener ce nouveau projet à terme et disposer d'une application fiable, il a été nécessaire d'effectuer quelques analyses et recherches quant au fonctionnement « bas niveau » du microscope ainsi que certaines opérations indispensables telles que la calibration ou encore la ré-initialisation de l'appareil. Ces analyses préalables vont être abordées dans la section suivante.

3.2.2. ANALYSES PRÉALABLES

Avant d'entamer le développement même de l'application, il était primordial de prendre connaissance de manière plus profonde du fonctionnement du microscope et la manière dont il traite les requêtes qui lui sont transmises. En effet, il était obligatoire d'aborder ces questions afin d'assurer une fiabilité

28 Précision qui évite le sur-recouvrement ou l'apparition de *gaps* entre les images à coregistrer

suffisante ainsi que la prise en charge des opérations nécessaires à son bon fonctionnement. Les informations qui vont suivre portent principalement sur la gestion des *stacks* qui régissent le traitement des commandes et la calibration de l'appareil.

3.2.2.1. La gestion des stacks

La communication avec le système de microscopie s'effectue par le biais du standard *RS-232*. Ainsi, il est possible d'envoyer des commandes et de recevoir des réponses à partir de n'importe quelle machine pourvue d'un port *COM*. Cependant, cette communication ne se fait pas directement entre un ordinateur et le microscope lui-même. En effet, ces deux éléments sont reliés à un module central (le *Multi-Control 2000*) où sont gérées ces interactions.

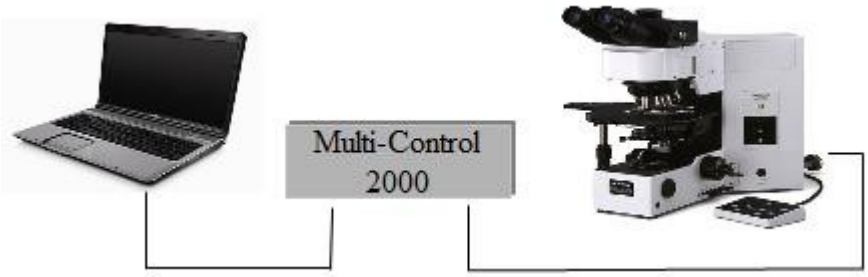

Illustration 12: Schéma simplifié du branchement entre l'ordinateur et le microscope

Toutes les commandes ainsi que les paramètres qui sont envoyés par l'ordinateur sont empilés au niveau du *Multi-Control* sur deux piles logiques distinctes. Les commandes sont placées sur une *stack* FIFO (*First In First Out*) alors que les paramètres sont placés sur une *stack* LIFO (*Last In First Out*) supportant jusqu'à 99 empilements. Si on prend l'exemple suivant « *18 4 82 23 4 89m* » qui fait appel à la commande de déplacement « *move* » (prenant jusqu'à 3 paramètres correspond aux 3 axes), l'état des piles évolue comme indiqué sur l'illustration 4. Les paramètres restant seront utilisés ultérieurement lorsqu'une autre commande sera placée en sommet de pile FIFO.

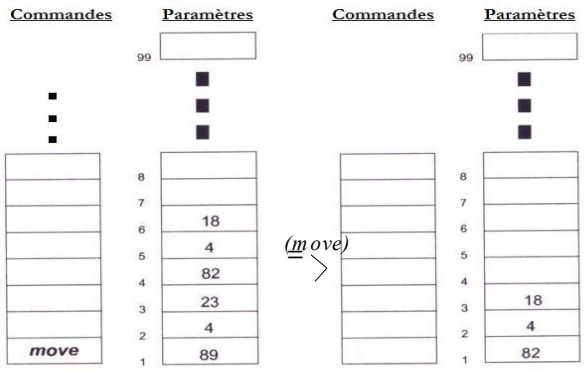

Illustration 13: Effet de l'application d'une commande "move" sur les stacks

Les tests réalisés sur ces *stacks* ont mis en évidence un problème de taille : pour des raisons particulières, il arrive qu'elles se désynchronisent. Il est possible que cela vienne du fonctionnement du *Multi-Control* même ou d'un enchaînement[29] incorrect de commandes pourtant valides. Dans ce cas, les commandes envoyées et les réponses reçues ne sont plus dans un format valide et prennent la forme d'une chaine de caractères inintelligible. Les ordres[30] qui sont transmis au microscope deviennent alors incertains et le comportement de l'appareil aléatoire (ce qui peut l'endommager). Lorsque cela se produit, le seul moyen de rétablir les *stacks* est de redémarrer le *Multi-Control* manuellement, ce qui pose problème lorsque l'on se place dans le cadre d'une application de capture automatique de frottis.

Pour trouver une solution à ce problème, l'idée a été d'analyser les interactions entre une application professionnelle existante et le microscope et de faire de la rétro-ingénierie sur base des messages échangés. Une telle application se doit d'être fiable et doit soit pouvoir éviter les dérèglements, soit être capable de rétablir les *stacks*. Dans un premier temps, le logiciel utilisé pour ces analyses a été WinPos qui est une application de base fournie avec le système de microscopie et permettant de commander des déplacements de l'appareil. Il a été utilisé car il a la particularité, dans son interface, d'afficher dans un champs de texte les commandes qu'il envoie. De plus, en cas de désynchronisation, le démarrage de cette application remettait systématiquement tout en ordre.

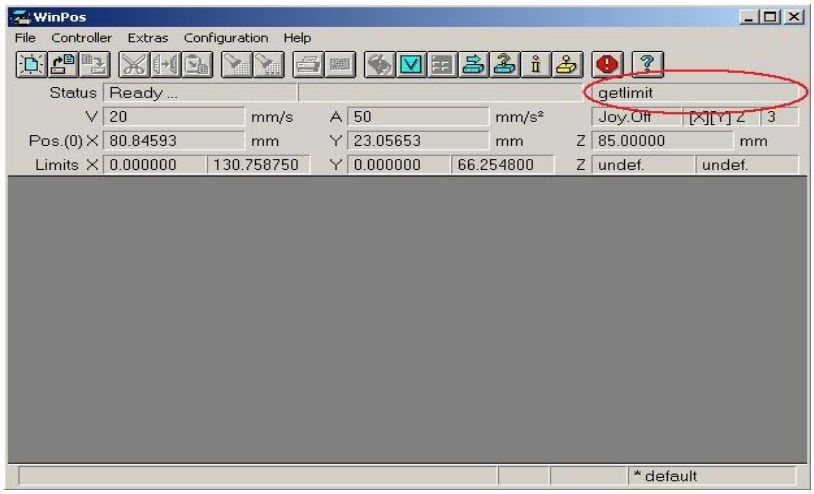

Illustration 14: L'interface de WinPos et l'affichage des commandes envoyées

Le test a consisté à désynchroniser[31] les piles manuellement, libérer le

29 La documentation de référence ne fait pas mention des inclusions et exclusions entre commandes et se borne simplement à les lister

30 Du moins, ceux qui sont syntaxiquement corrects (mais sémantiquement faux)

31 A l'aide d'un module de communication RS-232 développé en Java grâce auquel sont envoyées les commandes au *Multi-Control*

port de communication et ensuite y connecter WinPos. En se connectant, le programme effectue systématiquement un envoi groupé de commandes dans le but d'établir la connexion et d'initialiser une communication correcte. Le jeu de messages typiques transmis est repris à l'annexe 6.1. L'idée était donc de factoriser cet ensemble et de déterminer précisément quelles étaient les commandes qui rétablissaient effectivement la synchronisation des *stacks*. Cependant, dans ce jeu se trouvaient des *macros* internes à WinPos qui masquaient certains messages, notamment la *macro* « *CleanBuffers »* qui renfermait peut être la clé de la solution. En conséquence, sur base d'une factorisation des commandes visibles et d'autres provenant des manuels, il n'a pas été possible d'élaborer un message qui permette de rétablir systématiquement[32] la synchronisation.

Afin d'avoir accès à l'entièreté des commandes échangées et particulièrement aux commandes masquées derrière des *macros*, il fallait qu'une seconde application écoute le flux transmis par WinPos sur le port *COM* auquel il était connecté. Cependant, un tel port ne peut être réservé que par une seule application à la fois, sous peine de se voir lever une exception de type « *PortInUseException*[33] ». Le recours aux ports virtuels a donc été nécessaire. L'idée a été de connecter WinPos à un port virtuel[34] et de rediriger le flux vers le port physique sur lequel est branché le *Multi-Control*. Cette opération a été effectuée via le logiciel « *Bill Serial Monitor*[35] ». A gauche de l'interface se trouve le bloc de configuration du port virtuel et à droite, celui du port réel. En communication, les

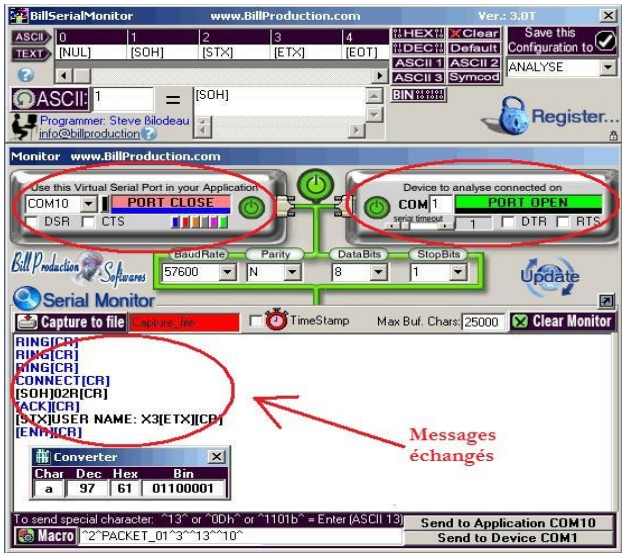

Illustration 15: L'interface de BillSerialMonitor, un programme de monitoring RS-232

32 Plusieurs variantes de messages capables de rétablir l'état des piles ont été composés, mais ils ne fonctionnaient pas de manière systématique, ce qui est une condition nécessaire pour la fiabilité de la future application

33 Dans le cas d'une application Java utilisant le *package* « *Comm* »

34 Dont l'identifiant ne peut être celui d'un port réel existant sur la l'ordinateur

35 Téléchargement de la démo : http://www.billproduction.com/serialmonitor/index_fr.html

échanges sont alors affichés dans la zone de texte au bas de l'application.

Cette solution a été déterminante car, dans l'ensemble des échanges réalisés entre les appareils, elle a permis de mettre en évidence une suite de commandes envoyées systématiquement lorsque les *stacks* du *Multi-Control* étaient déréglées. Cette suite se présente sous la forme des 4 instructions suivantes :

```
0 mode
clear
0 setioport
m ge ge ge
```

Illustration 16: Formule permettant une resynchronisation des stacks du Multi-Control

La première instruction sert à déterminer le mode d'opération du contrôleur : un paramètre « 0 » correspond au mode « *Host* » alors qu'un paramètre « 1 » correspond au mode « *Terminal* ». « *Clear* » a pour fonction d'enlever tous les éléments de la *stack*. Seule, elle n'amenait pas à systématiquement à une resynchronisation. La fonction « *setioport* » n'est renseignée dans aucun des manuels de référence. Son action est donc inconnue. Enfin, « *m* » correspond à la fonction « *move* » et « *ge* » à la commande « *getError* » qui retourne le code de la dernière erreur rencontrée. Leur enchaînement dans la formule est assez particulier car le mouvement n'a pas de paramètres[36] donc, physiquement, rien ne se passe et la fonction « ge » est exécutée 3 fois de suite. Les nombreux essais réalisés auparavant à partir du jeu de commandes obtenus de WinPos ont permis de constater que, dans la manipulation des *stacks*, le fait d'y placer ou d'en retirer un ou plusieurs éléments peut être déterminant pour une synchronisation. C'est peut-être pourquoi « *m* » n'a pas de paramètres et « *ge* » est répétée plusieurs fois. En tout cas, l'application de cette formule au moment d'une désynchronisation a **systématiquement** rétabli la situation. Ce problème a donc été considéré comme résolu. Il était alors temps d'envisager le traitement de la seconde condition préalable au bon fonctionnement de l'application, à savoir : la calibration du plateau du microscope.

3.2.2.2. La calibration

Conceptuellement, le plateau du microscope, sur lequel sont posées les lames porte-objet, peut être vu comme une grille sur laquelle il est possible d'indexer des positions auxquelles se déplacer. Il y a un point d'origine, le point (0,0), qui représente normalement la position minimale à laquelle il est possible de

36 Peut être l'appel à la commande « *setioport* » laisse des valeurs sur la pile LIFO qui sont alors pris en paramètres par la commande « *m* », mais sans documentation, il est impossible de le savoir

se déplacer. Dans le cas où l'appareil est bien calibré, ce point de repère se trouve en haut à gauche du plateau lorsque l'on est face au microscope.

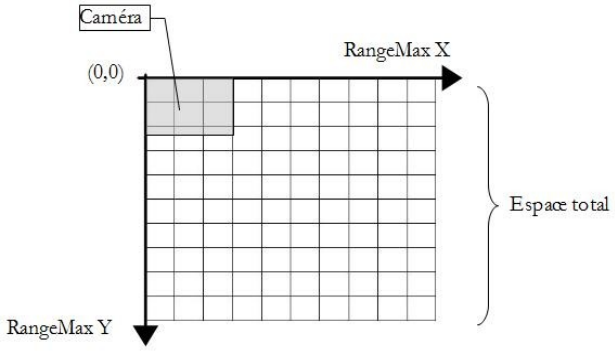

Illustration 17: Espace total indexable avec le plateau du microscope

Sur l'illustration 17, il faut considérer la caméra comme élément fixe : l'utilisateur a souhaité se rendre à la zone (0,0), il a déplacé le plateau de telle manière que cette zone se trouve sous l'objectif de la caméra[37]. Il en va de même pour n'importe quelle autre position.

Lorsque le microscope n'est pas calibré, le point d'origine ne correspond pas aux valeurs minimales des axes X et Y. Il se trouve alors à un endroit aléatoire dans l'espace couvert par le plateau. Si l'on se base sur l'exemple de l'illustration 18, l'utilisateur visionne toujours ce qui se trouve à la position (0,0), mais celle ci ne correspond plus au coin supérieur gauche (valeurs minimales indexables) de l'espace disponible.

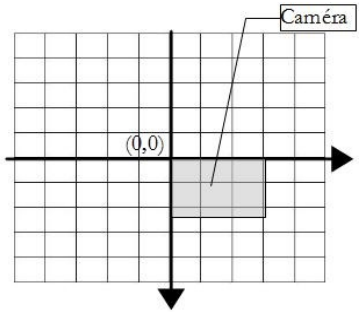

Illustration 18: Lorsque le microscope n'est pas calibré, le point d'origine peut se trouver n'importe où

La nécessité d'une calibration survient chaque fois que l'on doit démarrer le microscope. En effet, lors du démarrage, le point d'origine est attribué à la position à laquelle se trouve le plateau à ce moment là. L'illustration 18 représente cette situation : au moment du démarrage de l'appareil, le point d'origine

37 Remarquons qu'il n'y a en général rien à voir en ce point. En effet, la lame a observé est généralement placée au centre de l'espace couvert par le plateau et ne représente qu'une petite portion de cet espace

est affecté à la dernière position à laquelle se trouvait le plateau juste avant d'être éteint. Il est à noté que lorsque le microscope n'est pas calibré, il ne connait pas les valeurs maximales qu'il peut indexer sur chaque axe. Dès lors, lorsque l'on tente de dépasser ces limites, le plateau est violemment stoppé par les butées mécaniques. Il arrive parfois que l'impact soit suffisamment violent pour qu'il les dépasse (de peu). Dans ce cas, il faut éteindre l'appareil et déplacer le plateau à la main à l'aide de poulies spéciales prolongeant les axes. Dans le souci d'éviter d'endommager le matériel, il est recommandé de pratiquer une calibration à chaque démarrage du microscope. Voilà pourquoi il était nécessaire de faire des recherches pour maîtriser cette opération avant d'envisager le développement de la future application de pilotage.

Pour trouver les commandes nécessaires, il a également fallu analyser les messages échangés entre WinPos et le *Multi-Control* lorsque l'on commande cette calibration. Cela a permis de mettre en évidence deux commandes qui doivent inévitablement être exécutées ensemble. Il s'agit de « *cal* » (*calibrate*) et « *rm* » (*range mesure)*. La première commande un déplacement automatique du plateau du microscope jusqu'à l'extrémité supérieure gauche. La seconde commande le renvoie ensuite à l'extrémité inférieure droite jusqu'aux butées mécaniques, ce qui va permettre de définir l'espace maximal[38] qu'il est possible de couvrir. Cela garanti que le point d'origine (0,0) correspond bien aux valeurs minimales qu'il est possible d'indexer et que les valeurs maximales sont bien déterminées.

Ce dernier résultat clôt les analyses préalables qu'il était nécessaire de réaliser afin de s'assurer de :

- la fiabilité du *Multi-Control* et la précision d'indexation des positions (tests réalisés suite à la mise à jour d'AnalySIS évoqués au point 3.2.1), sans quoi il aurait été impossible de pouvoir effectuer des coregistrations d'images par la suite
- la maîtrise de la gestion des *stacks*, particulièrement lorsque celles-ci se désynchronisent
- la maîtrise de l'opération de calibration, nécessaire à chaque redémarrage de l'appareil

Ces trois conditions ayant été remplies, le développement de l'application de pilotage du microscope a pu commencer. Les sections suivantes vont en décrire les principes, l'IHM, l'architecture et le fonctionnement.

38 Ces cales d'arrêt sont des visses coulissante que l'on peut régler manuellement

3.2.3. PRINCIPES

Les deux sections suivantes vont aborder les deux grands principes qui sont implémentés dans l'application de pilotage, à savoir : la détermination d'une zone de capture au sein d'un échantillon et la possibilité de déterminer des parcours prédéfinis dans la zone créée.

3.2.3.1. Détermination de la zone de capture

Une zone de capture représente un sous espace d'un échantillon que l'on souhaite numériser. Cette zone est observée à l'aide d'une caméra numérique qui transmet les images à la carte *PCI* de l'ordinateur sur lequel elle est connectée. Les images reçues sont ensuite affichées dans une fenêtre d'aperçu qui a une taille (en pixels) connue[39]. On peut donc décomposer la zone de capture en blocs contigus de taille équivalente à celle de la fenêtre d'aperçu. Ces blocs représentent les images composantes de la future méga-image. La zone de capture peut donc être assimilée à un mur dont les briques sont des images. Pour construire ce mur, il faut déterminer le nombre d'images que l'on souhaite en largeur et en hauteur.

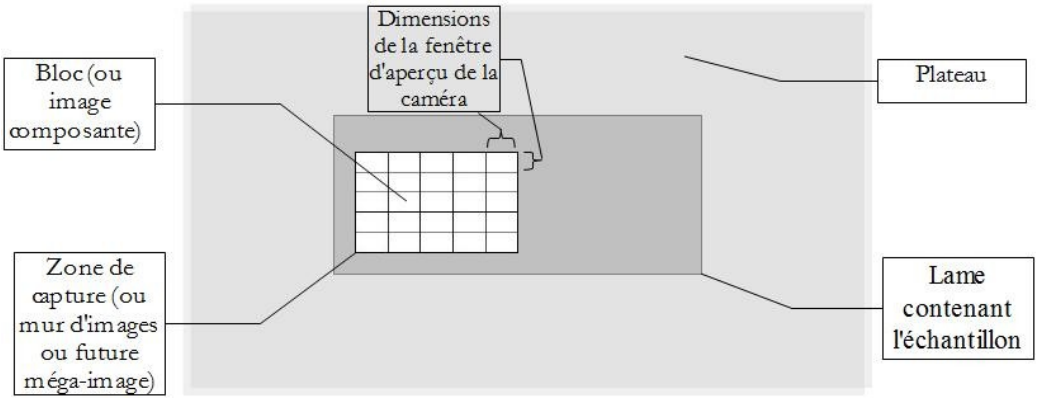

Illustration 19: Représentation de la zone de capture (le schéma n'est pas à l'échelle)

Afin de pouvoir reporter en millimètres[40] les tailles exprimées en pixels, il faut connaitre le *spacing* de chaque objectif du microscope. Le *spacing* est l'association d'une valeur de distance à un pixel. Le microscope étant équipé de 4 objectifs permettant 4 niveaux de grossissements[41], il a été nécessaire de mesurer 4 *spacings* différents à l'aide d'une lame témoin[42]. En effet, il est logique que plus le grossissement est élevé, plus le pixel représente une petite unité de surface.

A l'aide de la valeur de *spacing* pour l'objectif utilisé et du nombre d'images qui composent la zone de capture, on peut facilement construire le mur. Il s'agit de récupérer une seule fois les coordonnées de la position courante à laquelle

39 Dans notre cas, une hauteur de 512 pixels et une largeur de 680 pixels
40 Le microscope peut travailler en millimètres ou en micromètres, au choix
41 10x, 20x, 40x et 100x. Il peut également être équipé d'un objectif 60x
42 Lame spéciale sur laquelle sont imprimées des graduations de 10µm(0,01mm)

se trouve le plateau (qui correspond également à la position du bloc visé par la caméra, à partir duquel on veut construire le mur) et d'ajouter à ces coordonnées X et Y les valeurs de largeur et de hauteur (respectivement) de la fenêtre d'aperçu autant de fois qu'il y a d'images désirées en ligne et en colonne. Les valeurs des positions ne sont donc pas calculées relativement de bloc en bloc, mais à partir de la position du bloc de départ. Ceci afin d'éviter le report d'éventuelles erreurs d'imprécision du microscope. Effectivement, compte tenu des faiblesses[43] dont avait fait preuve l'appareil suite à la mise à jour d'AnalySIS, cette option a été choisie par sécurité. Ainsi, les positions calculées correspondent aux emplacements exacts auxquels les blocs doivent se trouver. Le pseudo-code de construction du mur est le suivant (en considérant que le bloc de départ représente le coin supérieur gauche du mur) :

```
récupérer position blocDépart;
j=0;
i=0;
tant que (i<blocs par colonne)
{
    tant que (j<blocs par ligne)
    {
        blocCourant.x=blocDépart.x+(j*fenêtreAperçu.largeur);
        blocCourant.y=blocDépart.y+(i*fenêtreAperçu.hauteur);
      j=j+1;
    }
    j=0;
   i=i+1;
}
```

Illustration 20: Pseudo-code de construction de la zone de capture

Dans le but de pouvoir permettre ultérieurement une coregistration des images composantes, il faut également considérer un recouvrement. La gestion de ce recouvrement est assez simple à mettre en œuvre car il s'agit de le soustraire des valeurs de largeur et de hauteur de la fenêtre d'aperçu. Ainsi, lors de la construction du mur d'images, dans les calculs de position des blocs dans la double boucle, les pas sont inférieurs à la taille de la fenêtre d'aperçu ce qui provoque le chevauchement de ces blocs et donc des images composantes de la future méga-image.

Jusqu'ici, le mur est toujours construit de la même façon en partant du même coin de départ : le bloc de départ correspond au coin supérieur gauche. Dans la pratique, il peut être utile de pouvoir construire la zone de capture à partir d'un autre coin. Dans ce cas, le recouvrement provoque une compression de la zone de capture dans la direction du coin de départ à partir duquel est construit le mur.

43 Même si, hors AnalySIS, le microscope semblait fonctionner normalement

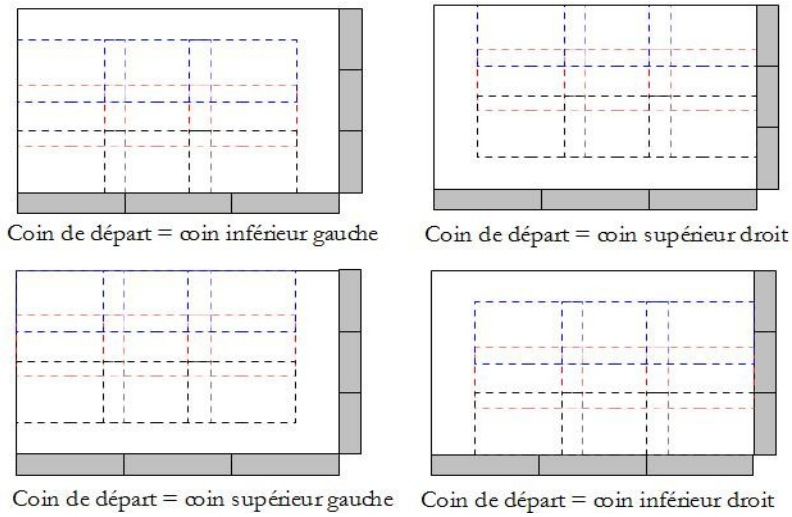

Illustration 21: Effet de compression causé par le recouvrement

Dans l'illustration 21, les rectangles gris représentent la largeur et la hauteur des blocs sans recouvrement, ce qui permet de voir la taille maximale qu'a la zone de capture dans ces conditions[44]. Un moyen simple d'adapter le calcul des positions des blocs lorsque le mur peut être construit à partir de coins différents consiste à ramener le bloc de départ choisi au coin supérieur gauche (en tenant compte de la compression). Dans le pseudo-code de l'illustration 20, cette opération se place juste après la première ligne qui récupère la position du bloc à partir duquel l'utilisateur souhaiterait construire son mur d'images. Par exemple, dans le cas d'une construction à partir du coin inférieur droit, les opérations nécessaires pour calculer les coordonnées du bloc supérieur gauche, en tenant compte de la compression, sont les suivantes :

```
newX= positionCourante.x-(maxColumns-1)*fenêtreAperçu.width_mm_overlapped;
newY = positionCourante.y-(maxRows-1)*fenêtreAperçu .height_mm_overlapped;
blocDépart = new Coordinate(newX,newY);
```

Où « *width_mm_overlapped* » et « *height_mm_overlapped* » représentent la largeur et la hauteur de la fenêtre d'aperçu de la caméra moins le recouvrement. Il suffit ensuite, d'appliquer la double boucle de la figure 20 et le mur d'image est construit.

3.2.3.2. Parcours dans la zone de capture

Comme la future application vise des captures automatiques d'images, il faut pouvoir déterminer des parcours (ou routes) prédéfinis au sein de la zone à numériser selon lesquels le programme pourra progresser seul. En tout, huit[45]

44 Ainsi, du fait du recouvrement, le mur construit a une taille inférieure ou égale à la taille réelle de la zone de capture

45 En fait, il y a un neuvième type de route qui correspond à un parcours libre selon lequel l'utilisateur peut sélectionner manuellement les blocs au fur et à mesure (voir section 3.2.5.4.)

parcours possibles ont été considérés. C'est donc à ce stade que les blocs vont pourvoir être indexés (numérotés, ordonnés). On passe donc d'une vue statique du mur (qui recueille les positions de toutes les images composant la zone de capture) à une vue dynamique (il possède en plus un sens et une direction).

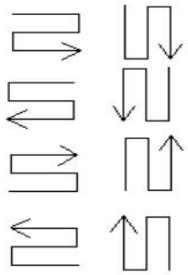

Illustration 22: Huit possibilités de parcourir un mur correspondant à une zone de capture (à gauche les directions horizontales et à droite les directions verticales)

L'algorithme qui prend en charge la gestion des parcours se base sur 4 variables principales : «*scan*», «*progression*», « *increment* » et « *reverse* ». La première variable correspond au mouvement en « *zig-zag* » du parcours, son sens alterne au cours des déplacements. La seconde variable fait référence à la progression générale du parcours, son sens ne varie pas. Les deux dernières variables sont en fait des paramètres techniques utilisées par les deux premières. « *reverse* » est une variable booléenne qui indique le sens de « *scan* ». Si « *reverse* » est égale à la valeur «*false*», alors « *scan* » sera incrémentée d'une unité (et inversément). « *increment* » est une variable entière dont le domaine est {-1,1}. Sa valeur dépend du coin à partir duquel démarre la route et de la direction du parcours : si la direction est horizontale et que l'on démarre d'un coin supérieur (gauche ou droite) alors « *increment* » vaut 1 (-1 si l'on démarre d'un coin inférieur), si la direction est verticale et que l'on démarre d'un coin gauche alors « *increment* » vaut 1 (-1 si l'on démarre d'un coin droit). Pour exemple, l'initialisation de ces variables est indiquée dans le tableau suivant (dont les cellules correspondent aux parcours présentés sur l'illustration 22).

progression=0 *scan=0* *increment=1* *reverse=false*	*progression=0* *scan=0* *increment=1* *reverse=false*
progression=0 *scan=maxBlocsParLigne* *increment=1* *reverse=true*	*progression=maxBlocsParLigne-1* *scan=0* *increment=-1* *reverse=false*
progression=maxBlocsParColonne-1 *scan=0* *increment=-1*	*progression=0* *scan=maxBlocsParColonne-1* *increment=1*

reverse=false	*reverse=true*
progression=maxBlocsParColonne-1 *scan=maxBlocParLigne-1* *increment=-1* *reverse=true*	*progression=maxBlocsParLigne-1* *scan=maxBlocParColonne-1* *increment=-1* *reverse=true*

La Variable « *scan* » changera de sens chaque fois qu'une borne maximale ou minimale est atteinte et, dans le même temps, « *reverse* » changera d'état. « *progression* », quant à elle, sera invariablement incrémentée de la valeur contenue dans « *increment* » (qui est soit positive ou négative) jusqu'à la fin du parcours :

```
/*size est la taille maximum d'une ligne ou d'une
colonne en fonction de la direction du parcours*/

Si (scan==size-1 && !reverse)
Alors reverse=true;
      progression=progression+increment;

Sinon
      Si (scan==0 && reverse)
      Alors reverse=false;
            progression=progression+increment;
      Sinon
            Si(reverse) scan--;
            Sinon scan++;
```

Enfin, en fonction de la direction du parcours, les variables « *progression* » et « *scan* » concernent soit des lignes, soit des colonnes. Il suffit donc de tester la direction du parcours pour commander le déplacement adéquat au bloc suivant :

```
/*avancer(ligne, colonne)*/

Si directionRoute == horizontale
Alors avancer(progression,scan);
Sinon avancer(scan,progression);
```

Ceci clôture la présentation des principes de base nécessaires à la réalisation de captures automatiques d'échantillons. Les sections suivantes vont se focaliser sur l'application développée qui implémente ces principes.

3.2.4. L'IHM DE LA NOUVELLE APPLICATION (R.I.S.C.)

L'application développé a pour vocation d'être utilisée directement par les experts. Il est donc nécessaire de prévoir une interface utilisateur qui soit compréhensible et donne accès aux fonctionnalités nécessaires à la réalisation d'une capture d'images multiples. Voici la *GUI*[46] de l'application de pilotage baptisée *R.I.S.C*[47] :

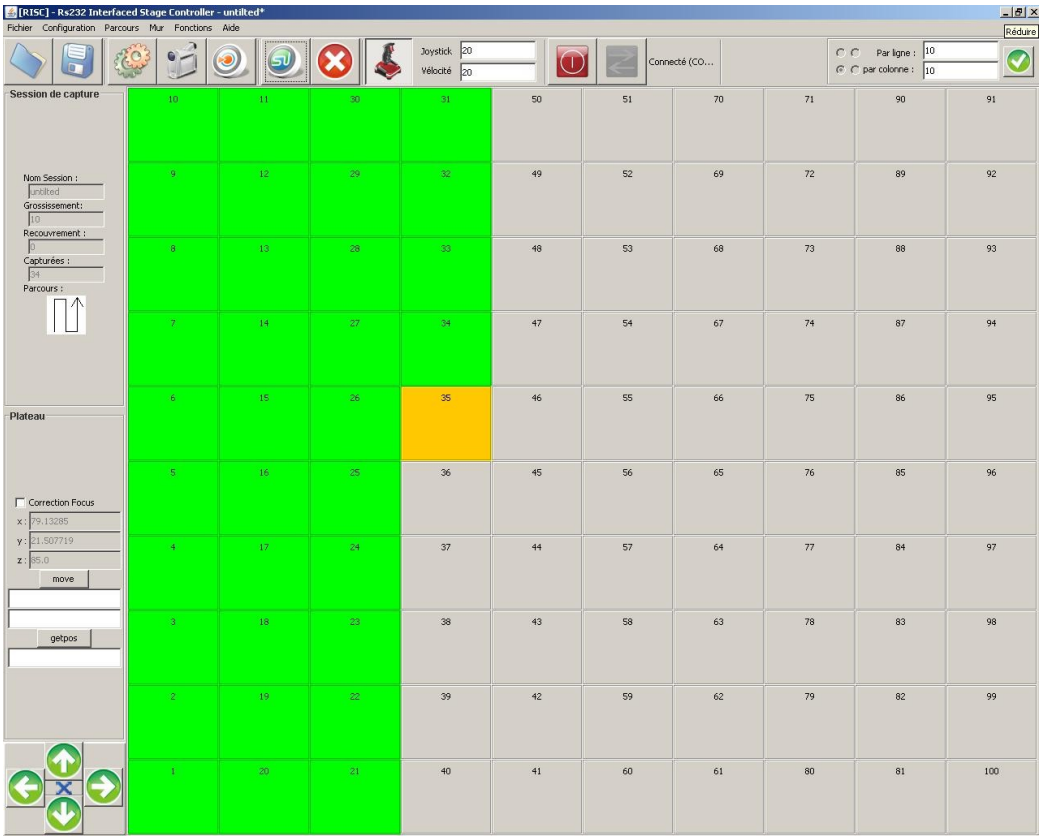

Illustration 23: Interface utilisateur de l'application de pilotage R.I.S.C.

On retrouve visuellement les deux principes présentés dans la section précédente. Au centre de l'interface se trouve le mur de blocs correspondant au découpage de la zone de capture. Les blocs en vert représentent les éléments déjà capturés selon le type de parcours choisi (dont on voit l'icône dans la colonne à gauche). On constate que les blocs sont numérotés en fonction de ce parcours.

Au sommet de la fenêtre se trouve le nom de la session de capture courante (en l'occurrence, dans ce cas ci, la session n'a pas encore été sauvée car elle est nommée « *untitled** »). Juste en dessous se trouvent les menus « Fichier » (qui permet de sauvegarder/charger des sessions et quitter l'application), « Configuration » (permettant de régler les paramètres d'application tels que ceux

46 *Graphical User Interface*
47 *RS-232 Interfaced Stage Controller*

du port, de la caméra ou ou d'effectuer une calibration), « Parcours » (qui offre la possibilité de définir un parcours, de lancer un parcours automatique ou de réinitialiser un parcours), « Fonctions » (qui offre certains services additionnels tels que l'exécution d'un *auto-focus software*[48]) et « aide ». Les icônes juste en dessous à gauche correspondent à des raccourcis vers ces fonctionnalités. La croix rouge permet de fermer la session courante, l'icône juste à droite active ou désactive le *joystick*, les deux champs de texte qui suivent permettent de régler les paramètres de vitesse de déplacement du plateau et la vitesse du *joystick*, ensuite arrivent les boutons de déconnexion et de connexion au port. Tout à droite se trouvent les éléments d'interface permettant de définir certains[49] paramètres nécessaires à la construction d'un mur, à savoir : le coin de départ, le nombre d'images par ligne et par colonne. Enfin, à gauche du mur de blocs se trouvent les informations sur la session courante (nom de session, grossissement utilisé, recouvrement, nombre d'images capturées et le type de route choisie) et sur le plateau (affichage des positions réelles, en millimètres, du bloc courant dans le mur et possibilité d'interactions de base avec le plateau).

L'image, quant à elle, est obtenue à l'aide du logiciel de pilotage du *driver* de la caméra numérique baptisé « *DP-Controller* ». L'ordinateur dédié à la capture d'images a été équipé de deux moniteurs afin d'en consacrer un à l'affichage du flux vidéo transmis par la caméra et de consacrer l'autre au pilotage du microscope. *DP-Controller* et *R.I.S.C.* sont donc utilisés côte à côte. Remarquons qu'il n'a pas été possible de prendre en charge la capture de l'image car les développements concernant une solution basée sur *TWAIN*[50] n'ont pas porté leurs fruits pour des raisons techniques propres à la carte d'acquisition. Néanmoins, il est prévu que cette prise en charge soit effective car des solutions existent pour les futurs développements[51].

48 Voir section 3.2.6.3.

49 Les autres paramètres ajustables sont : l'objectif (et son *spacing*), la taille de la fenêtre d'aperçu et le recouvrement. Tous se trouvent dans le menu « configuration ».

50 Protocole informatique standard destiné au contrôle logiciel des scanners de documents ou des appareils photos numériques

51 Voir la section 3.2.6.2

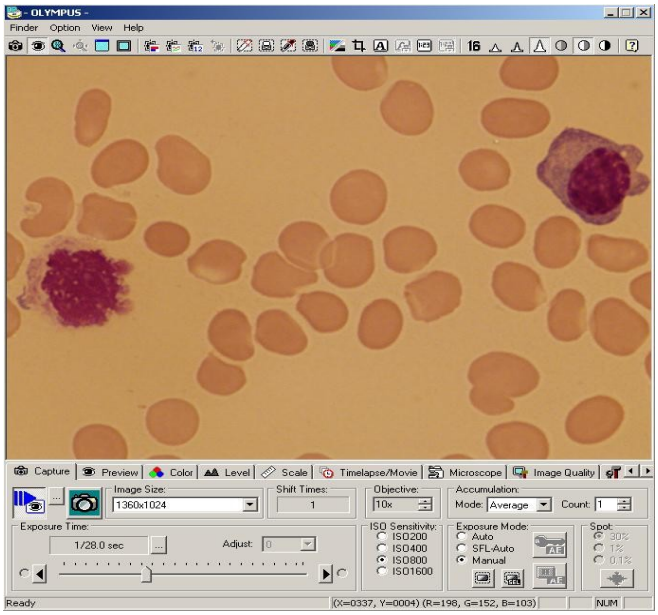

Illustration 24: L'interface utilisateur du logiciel DP-Controller

3.2.5. ARCHITECTURE ET FONCTIONNEMENT

3.2.5.1. La découpe en modules

Cette nouvelle application a pour vocation d'offrir une alternative à AnalySIS. Nous avons vu à la section 3.2.1 que cette réalisation a été motivée d'une part, par l'apparition de problèmes techniques majeurs et d'autre part, par les bénéfices que cela engendre. Parmi ces bénéfices, l'un fait référence au problème de l'aspect hétérogène de la station d'acquisition présentée à la section 2.3. En effet, les opérations de post-traitement telles que la coregistration sont codées en Java alors que les opérations de pilotage et de capture se présentent sous la forme de modules codés en *imaging-C* intégrés au sein d'AnalySIS qui se charge alors de la gestion (bas niveau) de la communication, du microscope et de la caméra. Le « monde » Java et le « monde » *imaging-C* étant connectés à l'aide d'appels système. L'idée est donc de proposer une solution de remplacement aux ensembles « modules *imaging-C* » et AnalySIS. Ce qui implique de prendre en charge à la fois le travail de captures multiples d'images d'un échantillon et le travail de gestion des interactions avec les différents appareils. Dans ce but, l'architecture de la nouvelle application s'articule autour de neuf modules qui se répartissent ces différentes tâches :

Nom du module	Rôle
application	Gère le démarrage du programme et le chargement des paramètres d'application

datas	Contient certaines donnéees et constantes globales
functions	Module destiné à accueillir des fonctionnalités additionnelles(ex: focus *software*)
gui	Contient les classes gérant l'interface et les *listeners* associés
iniFiles	Gère la création, le chargement, la sauvegarde et l'écriture dans des fichiers *.ini. Permet de sauvegarder les paramètres d'application
portsCom	Gère l'initialisation, la configuration, l'envoi et la réception de données sur un port RS-232
risc	Contient les classes permettant la création d'une session de capture, l'élaboration mur d'image et les parcours automatiques
twain	Module destiné à interagir avec la caméra (*deprecated*)
utils	Contient diverses fonctions statiques utiles

3.2.5.2. Les classes principales

Le module « *risc* » représente le cœur de l'application. En effet, c'est à l'intérieur de celui-ci que se trouvent les structures de données et les traitements nécessaires à la numérisation des échantillons. L'illustration 25 montre les classes principales qui le constituent et la manière dont elles sont associées entre elles. Pour des raisons de lisibilité, tous les attributs et méthodes n'ont pas été repris. Seul certains figurent afin de montrer le genre de traitement que prend en charge la classe. Le but de se schéma est de montrer le fonctionnement général de ce *package*.

Comme on peut le voir, le module « *risc* » est composé de plusieurs *managers* dont un nommé « *SessionManager* ». Celui ci prend en charge des sessions (une à la fois) et se charge des opérations de gestion qui leur sont inhérentes. Une session représente l'état d'une séance de capture. Elle regroupe donc les informations indispensables qui permettent de sauvegarder le travail accompli. A une session de capture sont associés un mur d'images (la classe « *Wall* ») et une route (la classe « *AbstractRoute* »). Le mur représente la zone de la lame qui va être effectivement capturée. Il est composé de blocs qui sont définis par la classe « *Tile* » qui possède plusieurs informations d'état et notamment la position réelle sur la lame (« *Coordinate* »).La session contient également une route. Comme son nom l'indique, « *AbstractRoute* » est une classe abstraite implémentée par deux types de routes différents : d'une part, « *FreeRoute* » qui correspond à un parcours manuel et libre du mur et d'autre part, « *FollowedRoute* » qui fait référence à un parcours prédéfini. A une route est associé un traitement défini par la classe « *Process* » qui est aussi abstraite. Pour l'instant, seuls deux traitements implémentent cette classe : « *ValidateTile* » qui réalise le traitement (typiquement un capture) et la validation d'un bloc, et « *computeTileIndex* » qui est utilisé pour numéroter les blocs selon le parcours prédéfini choisi.

Notons qu'au sein de ce module, d'autres classes sont nécessaires pour réaliser l'ensemble des opérations requises. A cette fin, une série d'autres *managers* composés de méthodes spécifiques sont utilisés par les classes décrites ci-dessus. On trouve donc « *CamProperties* » (qui gère les informations de la caméra), *MovementManager* (qui prend en charge la gestion de mouvements), *PortProperties* (qui gère les informations sur le port), *PreviewWindowProperties* (qui régit les données concernant la fenêtre d'aperçu) et « *StageProperties* » (qui gère les informations et traitements – exemple : la calibration – relatifs au plateau du microscope).

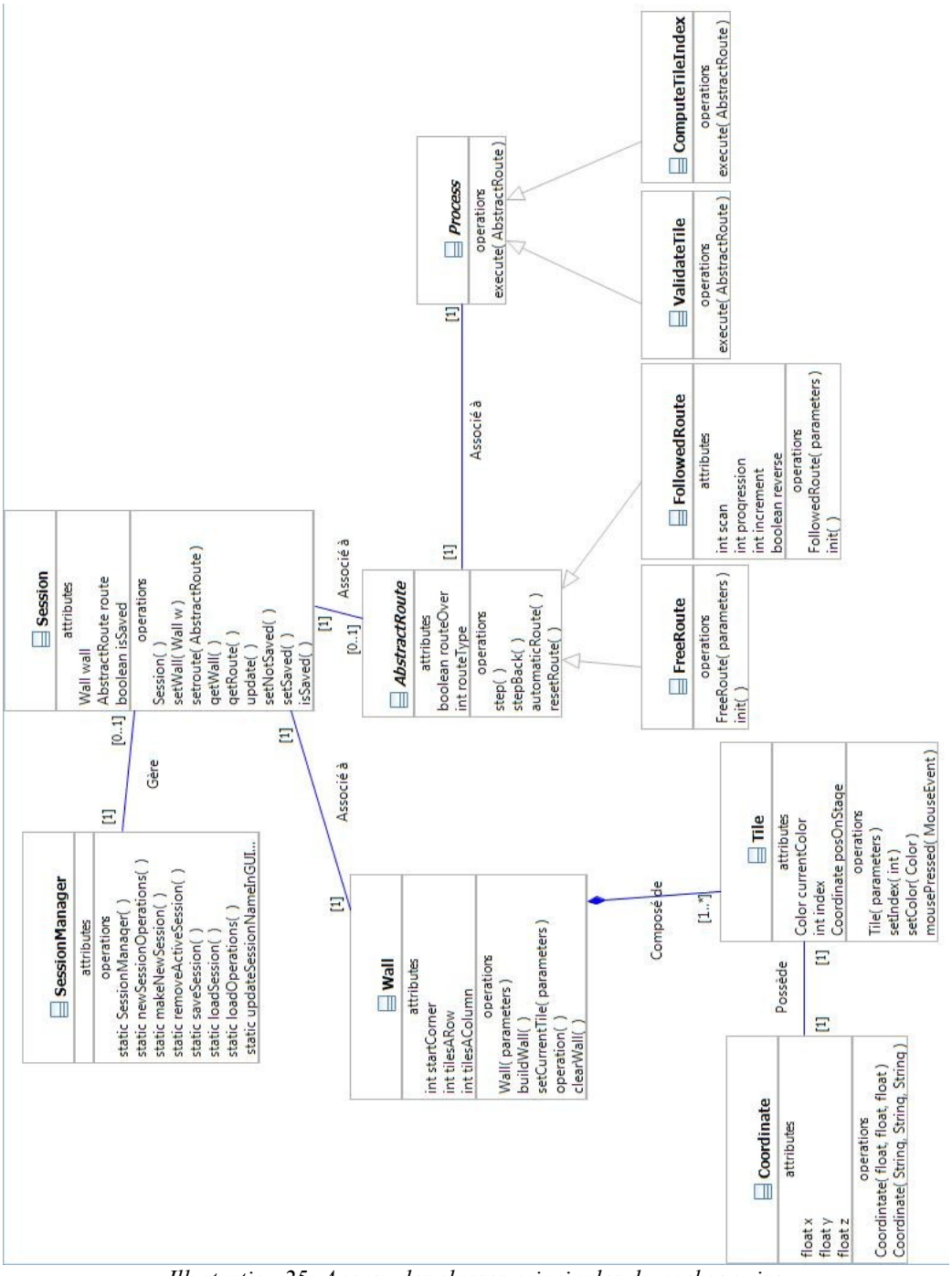

Illustration 25: Aperçu des classes principales du package risc

3.2.5.3. Les sessions

La numérisation d'une zone d'un échantillon est une opération lourde qui demande un certain temps d'exécution. Or, le système étant dépendant d'appareils et d'un réseau, une défaillance extérieur peut survenir. Dans le souci de ne pas perdre le travail qui aurait déjà été accompli, l'idée a été d'organiser les séances de captures sous la forme de sessions qu'il est possible de sauvegarder et charger. Il y a lieu de sauver de l'information de capture à partir du moment où l'utilisateur crée un mur. Ainsi, une session se crée à partir de ce moment là. En effet, ce mur représente des données importantes qui correspondent à une zone bien précise sur la lame porte-objet. Le mur sans parcours représente la condition de départ de la capture. Si à ce moment survenait un problème, il suffirait à l'utilisateur de sauver et de recharger la session pour retrouver la configuration de zone qu'il avait établie. Cependant, la classe « *SessionManager* » ne se contente pas de sauver uniquement l'objet « *Session* ». En effet, même si ce dernier contient les informations essentielles sur le mur, l'éventuelle route et la progression générale de la capture, il manque les paramètres techniques sous-jacents tels que la taille de la fenêtre d'aperçu, le recouvrement choisi, le grossissement, les paramètres de connexion, etc. Ainsi, lorsqu'une session a besoin d'être sauvée, l'objet « *SessionManager* » crée un objet « *ProgramState* » qui reprend l'ensemble des données nécessaires pour recharger ultérieurement l'état du programme à l'identique. Les objets qui constituent cet état sont : « Session », « *PortProperties* », « *CamProperties* », « *previewWindowProperties* » et « *StageProperties* ». L'objet « *ProgramState* » créé est ensuite sérialisé[52] en un fichier « **.rsc* ». Au delà du souci de sécurité, cela représente un moyen de sauvegarder des informations numériques quant à un échantillon, de la même manière que l'on conserve la méga-image d'un frottis.

3.2.5.4. Les parcours dans le mur d'image

Comme l'illustre le diagramme de classes de la figure 25, il existe deux types de route : libre et prédéfini. Ce dernier correspond directement à l'idée d'automatisation de numérisation d'échantillon. En effet, comme le point de départ, le sens et la direction sont définis, le programme sait exactement comment se déplacer. Dès lors, à quoi peut servir le type de route « libre » ? En fait, il n'est pas rare qu'a la suite d'une série de captures, certaines se révèlent être de qualité insuffisante. Dans ce cas, il est judicieux d'avoir la possibilité de ne re traiter à nouveau que celles-ci. Cependant, avec un système entièrement automatisé, ce serait impossible. D'où l'idée de ce parcours qui permet alors de numériser au choix les images manuellement. Ainsi, si après une série de captures d'images (destinée à une coregistration), certaines d'entre elles sont à recommencer, il est possible sans problème de permuter d'un parcours prédéfini à un parcours libre et de viser les

52 Opération qui consiste à écrire des données présentes en mémoire centrale vers un flux de données binaires, ce qui les rend alors persistantes

images concernées. Les classes « *FreeRoute* » et « *FollowedRoute* » prennent en charge ces fonctionnalités. Elles sont quelque peu différentes mais reposent globalement sur le même principe illustré par le pseudo-code suivant :

```
Initialisation de la route;
Aller sur la première tile;

Tant que la route n'est pas finie
{
      ProcessTile();
      VirtualMove();
      RealMove();
}
```

Globalement, en considérant le déplacement initial[53], il s'agit de se déplacer à la *tile* à traiter (automatiquement ou manuellement) et d'opérer le travail qu'elle doit subir. Le pseudo-code fait référence à deux types de déplacement : l'un virtuel et l'autre réel. Pourquoi avoir besoin de cette distinction ? Comme nous l'avons évoqué à la section précédente, le système base son traitement sur du *hardware* et de la connectique réseau. Or, rien n'est infaillible et il se pourrait que l'un des éléments devienne défaillant. Le mouvement virtuel correspond au calcul de la position de la *tile* suivante à traiter et à la mise à jour de la progression. Le mouvement réel se base sur ces nouvelles informations. Ainsi, même si le matériel fait défaut, au moins l'état du système correspond exactement à la situation à laquelle il devrait se trouver. Donc, après rétablissement du matériel, le programme pourra reprendre son fonctionnement là où il y a eu panne. Cette question sera abordée plus précisément dans la section suivante.

Remarquons que le traitement qui est effectué dans une route correspond typiquement à une capture. Mais qu'en serait-il s'il y avait d'autres traitements[54] ? Cela voudrait dire qu'il faudrait réécrire le code du traitement (« *ProcessTiles()* » selon le pseudo-code ci-dessus) et donc il y aurait autant de routes que de traitement. C'est là que la classe « *Process* » entre en jeu. Lorsqu'une route est créée, on lui associe un traitement (un objet « *Process* »). Ainsi, lorsque la route (libre ou prédéfinie) doit exécuter le traitement prévu sur la *tile*, elle ne fait qu'appeler l'exécution du « *Process* » prévu. De ce fait, avec ces deux seules classes (qui, rappelons le, représentent un type de parcours du mur, indépendamment du traitement qui est appliqué), il est possible d'effectuer n'importe quel traitement.

53 Dans le cas d'un parcours manuel, la première *tile* est considérée comme celle dans le coin supérieure gauche. Dans un parcours prédéfini, la première tile dépend du type de parcours prédéfini choisi (pour rappel, voir l'illustration 22)

54 Comme évoqué au point 3.2.5.2, c'est le cas car un autre traitement existe : il s'agit de la numérotation des *tiles*

Pour terminer cette section, il faut préciser que les routes prédéfinies (celles qui ne sont donc pas libres) peuvent être exécutées de deux manières : automatiquement et semi-automatiquement. Lorsqu'une route prédéfinie est exécutée automatiquement, l'enchaînement des étapes (et des captures) est pris en charge par une boucle qui ne s'interrompt pas (à moins qu'une panne ne survienne, voir la section suivante à ce sujet). Lorsqu'une route est exécutée semi-automatiquement, il revient à l'utilisateur de commander l'exécution d'un pas manuellement (dans l'application, via le menu « Parcours » ou avec un raccourci « F8 »). Le programme traite alors la *tile* courante, passe à la suivante et attend. La classe qui gère ces parcours prédéfinis reste la même quelque soit le type d'exécution. Il est d'ailleurs possible dans le programme de passer d'un type à l'autre dynamiquement car les paramètres concernant l'état de la route sont gérés au niveau de la classe elle même et non pas au niveau d'une fonction ou d'une boucle. Alors que l'on envisage d'automatiser au maximum la numérisation des échantillons, cette idée de parcours semi-automatique peut paraître inutile. En fait, il a été nécessaire de prévoir cette fonctionnalité[55] dans le programme car il n'a pas été possible d'apporter une solution logicielle en Java au pilotage de la caméra. En effet, la capture reste prise en charge par une application extérieure « *DP-Controller* » qui nécessite une intervention humaine pour la capture des images. Bien que des recherches et des essais dans ce domaine étaient en cours, le microscope devait quand même rester opérationnel pour les tâches du laboratoire (rappelons que AnalySIS était inutilisable, il n'y avait plus rien pour piloter convenablement le microscope informatiquement). Les détails concernant le pilotage de la caméra seront abordés à la section 3.2.6.2.

3.2.5.5. Fiabilité

Les deux sections précédentes expliquent plus en détails les concepts de session et de parcours et chacune fait référence aux problèmes qui peuvent survenir lors de l'utilisation du réseau ou du matériel. Cette section va faire le lien entre ces deux discussions. En effet, la fiabilité du système repose sur les sessions et la distinction entre les mouvements virtuels et réels.

Lorsqu'une panne[56] se produit pendant un processus de capture dans une route (qu'elle soit libre ou prédéfinie), il faut pouvoir détecter ce problème le plus tôt possible de manière à ne pas exécuter un traitement sur une mauvaise *tile*. En effet, le genre de problèmes qui peuvent survenir entraînent une incapacité du matériel à réagir aux commandes, ce qui rend impossible les déplacements réels. Sans vérification de cet avancement réel, le programme pourrait traiter ce qu'il considère comme la *tile* suivante alors que le microscope n'a pas bougé. Le

55 Qui peut tout-à-fait ne plus être utilisée par la suite

56 Typiquement une rupture de la connexion ou une défaillance du microscope. En réalité, il s'agira surtout d'un blocage du microscope

problème de la commande « *move* » vient du fait qu'elle est « *one-way* » : le microscope ne renvoie aucune information en retour et il est impossible de savoir si elle s'est bien exécutée. Dès lors, il est nécessaire de sonder régulièrement la connexion *R2-232* pour voir si tout est opérationnel. La question est alors : « où sonder pour avoir la fiabilité maximale ? ». La réponse logique est : « au moment de déplacer le plateau ». Plus précisément, lorsque l'on s'apprête à exécuter un mouvement réel. Comme évoqué au point 3.2.5.2., c'est le « *MovementManager* » qui prend en charge les mouvements. Lorsqu'un déplacement doit être effectué, le « *MovementManager* » demande à l'objet « *StageProperties*[57] » d'effectuer un sondage du réseau et du matériel. Si un problème est détecté, le « *StageProperties* » met sa variable booléenne interne « *unavailable* » à *true*, ce qui indique que le matériel est indisponible. De son coté, le « *MovementManager* » n'effectue aucune tentative de mouvement et retourne *false*. Comme ce dernier est utilisé dans un parcours pour commander un mouvement vers une *tile*, la route appelante se rend compte qu'il y a un problème et s'interrompt en mettant sa variable booléenne interne « *routeInterrupted* » à *true*. La figure 26 montre les objets principaux qui interviennent lors de l'exécution d'un pas selon une route prédéfinie. Si un problème est détecté par le « *StagePropertie* » un retour négatif est propagé jusqu'à la route et aucun déplacement n'est commandé. Si l'on est dans une route prédéfinie exécutée automatiquement, le pas suivant arrêtera la boucle, ce qui interrompra la progression du parcours. Si la route est exécutée semi-automatiquement, chaque fois que l'utilisateur commandera l'avancement d'un pas, rien ne sera effectué (et un message l'avertira du problème technique) car le test sur la variable « *routeInterrupted* » (qui est à *false* dans ce cas) empêchera tout traitement. Dans tous les cas, le programme informera l'utilisateur qu'il doit sauver la session courante et la recharger après avoir vérifier la connexion réseau et le microscope. Les erreurs de communication *RS-232* sont rares. Aucune erreur de ce type n'est d'ailleurs survenue pendant toute la durée du développement (à moins de débrancher un câble). Les erreurs viennent surtout du microscope, principalement lorsque celui ci exécute une opération étrange qui le bloque complètement. Selon nous, cette opération s'apparente à une mise en veille de l'appareil car elle ne semble exécutée qu'après un certain temps d'inactivité[58]. Dans ce cas, il faut redémarrer[59] le microscope et faire une calibration. C'est au rechargement de la session que le programme va effectuer un sondage pour tester la disponibilité du microscope. S'il répond bien, la variable « *unavailable* » dans « *StageProperties* » sera remise à *false*. Dans le même temps, si le programme détecte une route interrompue, il lèvera cette

57 Le « *MovementManager* » exécute des déplacements de base, il gère des positions. Le « *StageProperties* » contient les informations sur la manière de sonder le microscope, son état courant et prend en charge cette opération

58 Au final, c'est le seul problème matériel auquel nous avons été confrontés. Aucun autre genre de panne possible n'a été remarquée

59 Aucune commande permettant de lever ce blocage sans avoir à redémarrer n'a été trouvée dans les manuels du microscope

interruption et invitera l'utilisateur à reprendre le cours de cette route. Ainsi, on ne peut lever le statut d'indisponibilité du matériel et réactiver la route qu'en rechargeant une session. C'est indispensable car pour rétablir le microscope, il convient de le redémarrer. Donc, il faut suivre cette opération d'une calibration et une calibration s'exécute hors session[60], ce qui implique de sauver la session courante.

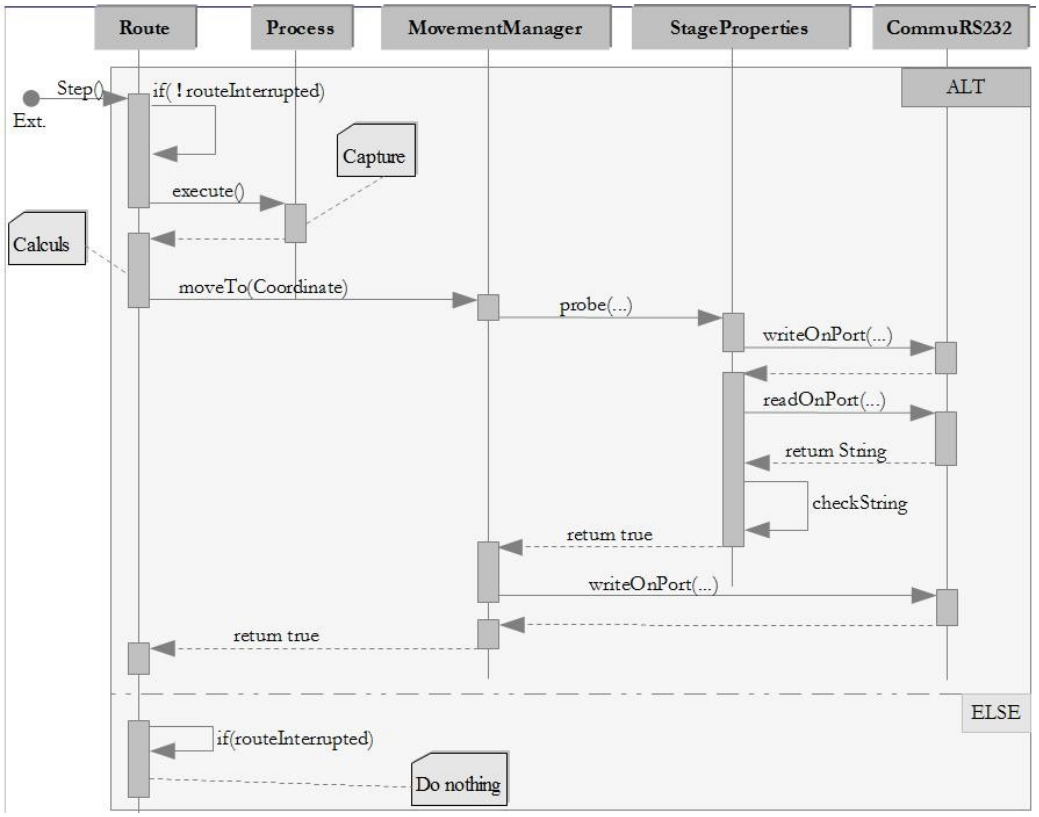

Illustration 26: Diagramme de séquence de l'exécution d'une étape dans une route prédéfinie

3.2.6. IMPACT SUR LES OBJECTIFS

Dans ce chapitre, nous avons évoqué le fait qu'il était nécessaire de développer une nouvelle application afin de prendre en charge des opérations indispensables à la composition automatique d'échantillons numériques. Nous avons vu les principes et le fonctionnement de cette nouvelle application. Mais qu'en est – il des objectifs fixés au chapitre 2 ? Dans la prochaine section, nous allons voir l'influence directe qu'ont eu ces développements sur la réalisation de ces objectifs de pilotage du microscope, de pilotage de la caméra, de coregistration et de gestion de l'*auto-focus*.

60 Pour des raisons de sécurité quant au matériel

3.2.6.1. Pilotage du microscope et coregistration

Le problème majeur rencontré suite à la mise à jour d'AnalySIS a été l'apparition de trous de tailles aléatoires dans les méga-images. Les causes de ce problème se situent au niveau du pilotage du plateau du microscope qui rencontrait des difficultés à indexer précisément les positions des images dans la zone de capture. Aucune raison valide n'a pu être donnée à ce sujet, ce qui remettait également en cause la fiabilité du matériel (ce qui a, par ailleurs, fortement motivé la recherche de solutions alternatives).

Ces problèmes ont été résolus par la nouvelle application. En effet, lorsqu'un mur d'images est construit, les déplacements aux *tiles* se font précisément et plus aucun trou n'apparaît entre celles ci. Cependant, il y a quand même un léger décalage qui se produit systématiquement au niveau de l'axe X entre les deux premières *tiles* d'une ligne (le caractère non aléatoire de ce phénomène est quelque peu rassurant). En effet, à chaque rangée, le déplacement de la première *tile* à la suivante est un peu trop court. Ce phénomène a été observé en testant la précision des déplacements à haut grossissement sans recouvrement. Dans ce cas, en observant l'image à travers la fenêtre d'aperçu et en considérant un déplacement de la gauche vers la droite, après mouvement, le bord droit de la première *tile* devrait correspond au bord gauche de la seconde. Ce qui n'était pas le cas, le déplacement est trop court de quelques pixels. La société *Olympus* avait évoqué un problème possible de résolution insuffisante du moteur (*linear encoding*), mais deux observations mettaient à mal cette hypothèse : le décalage ne se produisait pas sur tous les déplacements et la taille de ce décalage est bien supérieur à la capacité de résolution du moteur de l'axe X du microscope (qui reste élevée et suffisante). Après de nombreux autres tests, nous nous sommes aperçus que ce phénomène se produisait uniquement lorsque le moteur changeait de direction. Sachant que les parcours dans les murs d'images s'effectue en « *zig-zag* », ceci expliquerait pourquoi le décalage ne survient qu'au début de chaque ligne : c'est en effet à ce moment que le moteur contrôlant l'axe X change de sens. Les raisons de ce constat peuvent être multiples, il peut s'agir d'un problème de *firmware*, d'une conséquence malheureuse de la mise à jour ou encore d'un besoin de maintenance du moteur. Il est aussi possible que ce problème ait toujours été là sans qu'il soit remarqué (le décalage ne mesure que quelques pixels visibles au grossissement maximum). Au final, ce décalage ne pose aucun problème[61] pour la construction du mur ainsi que la conservation du recouvrement et le seul impact qu'il a sur le mur est que ce dernier est très légèrement (cela reste invisible à l'œil nu) compressé horizontalement.

L'absence de ces « trous » entre les images permet d'envisager à

61 Etant donné que le décalage vient du fait que le déplacement est légèrement trop court, des trous ne peuvent apparaître entre les images et le recouvrement est maintenu (et il sera légèrement plus grand)

nouveau une coregistration efficace des images. Les tests réalisés nous ont permis de constater que le problème de décalage évoqué ci-dessus n'a pas eu d'influence sur la réalisation des méga-images car, au final, elles forment un bloc compact homogène.

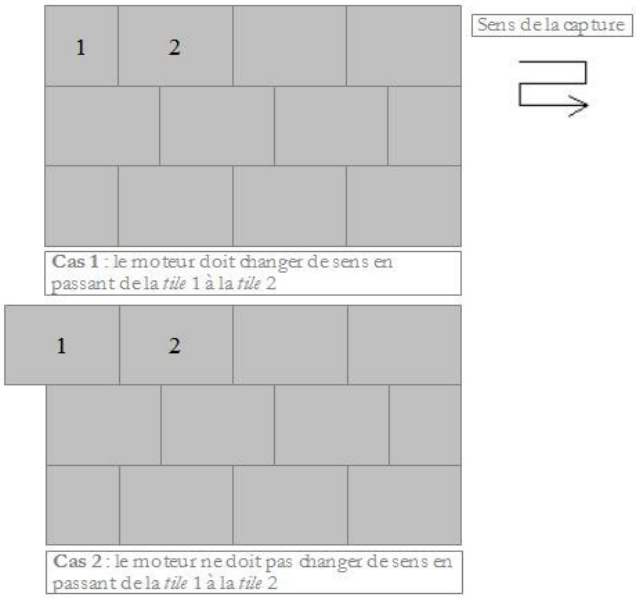

Illustration 27: Illustration de la structure d'une méga-image après coregistration

L'illustration 27 montre les deux types de structures que peuvent avoir les méga-images finales (selon le parcours de capture indiqué) en fonction du changement de rotation (ou pas) du moteur de l'axe X. Par exemple, le cas numéro un se produit lorsque le dernier mouvement effectué par le moteur, avant le démarrage de la session de capture, est un déplacement à gauche. De ce fait, dès le premier mouvement de la *tile* un à la *tile* deux, le moteur doit changer de sens ce qui produit ce décalage trop court et chevauche davantage ces deux images. Au final, on obtient quand même des blocs homogènes et compact qu'il est possible de coregistrer. X. Bocken ([boc08]), le dernier mémorand a avoir travaillé sur ce projet, avait mis à jour les algorithmes de coregistration développés par L. Zuyderhoff ([zuy03]) afin que ceux-ci puissent fonctionner pour des images de résolution quelconque. L'étape suivant cette mise à jour est l'intégration de ses travaux au sein de la station d'acquisition (nouvelle version). Cependant, je n'ai pas eu l'occasion d'effectuer cette tâche car l'importance de certains problèmes techniques rencontrés au cours de ce stage ont redéfini les priorités. Néanmoins, il est clair que cette intégration fera partie des priorités des prochains développements[62]. En tout cas, les essais de coregistration réalisés à l'aide d'une application[63] indépendante sur base des images capturées par la nouvelle application ont tous donné des méga-images de bonne qualité parfaitement

62 Voir la partie « Discussions »

exploitable. C'est donc un bon point de départ pour la future intégration de la mise à jour des algorithmes de coregistration.

3.2.6.2. Pilotage de la caméra

De manière générale, pour pouvoir prendre en charge l'entièreté du processus de numérisation d'un échantillon, il convient de disposer de moyens informatiques permettant le pilotage du microscope, la capture via la caméra numérique et la coregistration des images obtenues (auparavant, AnalySIS prenait en charge les deux premiers points). Pour répondre à ces besoins, nous disposons maintenant de la nouvelle application *R.I.S.C.* (une alternative à AnalySIS) et du serveur de coregistration[64]. Bien qu'actuellement ces deux parties ne soient pas encore fusionnées, elles couvrent les opérations indispensables de pilotage du microscope et de coregistration. Reste le pilotage de la caméra. Des recherches et développements ont été réalisés dans le souci de pouvoir insérer cette prise en charge directement dans l'application *R.I.S.C.* La tentative la plus prometteuse a consisté à utiliser le standard « *TWAIN* ». Cependant, lors des premiers tests, il était impossible de détecter la caméra. Pourtant, la documentation mentionnait cette possibilité. En fait, pour que l'utilisation du standard soit fonctionnelle, il faut prendre le soin d'activer la prise en charge « *TWAIN* » lors de l'installation du *driver* de la caméra. Nous avons donc réalisé cette opération et continué les tests. Malheureusement, le pilotage par ce moyen n'a pas rencontré nos espérances. La caméra était bien reconnue, mais lorsque l'on tentait de réaliser une capture, cela ne provoquait pas un « *snapshot* » mais l'ouverture de l'application « *DP-Controller* » (sur laquelle il est alors nécessaire d'appuyer sur un bouton déclenchant la capture d'écran). Ce constat constituait une pierre d'achoppement à l'objectif de capture automatique. Pour rappel, c'est notamment pour cette raison que l'application *R.I.S.C.* autorise des parcours semi-automatiques, de manière à permettre à l'utilisateur de faire un « *snapshot* » manuellement avec le « *DP-Controller* ». De plus, cela nous écartait de la nouvelle tendance de développement qui vise à intégrer l'entièreté du système en une application homogène codée en Java. Etant donné que les deux seuls moyens[65] de piloter la caméra étaient « TWAIN » (qui ne permet pas un pilotage direct) ou « *DP-controller* » (qui nécessite une intervention humaine), nous avons orienté les recherches vers l'utilisation d'un robot[66] qu'il serait possible de programmer pour appuyer automatiquement sur le bouton du programme « *DP-Controller* ». Ceci représentait l'ultime possibilité d'une automatisation complète du système de capture (même si moins élégante). Cependant, un rebondissement a tout remis en cause. En effet, bien après la fin du stage, nous

63 L'application utilisée est un logiciel de coregistration d'images photo baptisé « *Stitcher* » (http://usa.autodesk.com)

64 Dont les algorithmes ont été mis à jour et devrons y être intégrés

65 Moyen prévus par le constructeur

66 Nous envisagions d'utiliser « *AutoIt* » (http://www.autoitscript.com)

avons reçu les librairies (fichiers « **.dll* ») permettant un contrôle direct de la caméra ainsi que la documentation technique sur le sujet. Grâce à ces fichiers, il sera possible de se passer du programme « *DP-Controller* » et d'interagir directement avec la caméra. Cette prise en charge pourra alors être intégrée à l'application « *R.I.S.C.* » à l'aide de « *JNI*[67] » ou en travaillant directement avec les librairies *dll.* (comme le fait actuellement le module de communication *RS-232* de la nouvelle application). En conclusion, le développement d'une solution entièrement Java de numérisation prenant en charge à la fois les gestions de pilotage du microscope, de pilotage de la caméra et de coregistration reste possible.

3.2.6.3. Auto-focus

Depuis plusieurs années, un obstacle fait toujours face aux avancées dans la numérisation automatique des échantillons. Il concerne le module *U-AF* gérant un *auto-focus hardware* et qui est connecté au microscope. Ce module a pour rôle de permettre l'obtention d'images les plus nettes possibles pour une qualité optimale de méga-images. Cependant, il y a un problème face auquel cet appareil est désarmé. En effet, lorsque la zone[68] de mise au point, dans laquelle il fait ses calculs de netteté, n'est pas suffisamment contrastée, il ne peut trouver un niveau de hauteur adéquat pour le plateau et retourne une erreur.

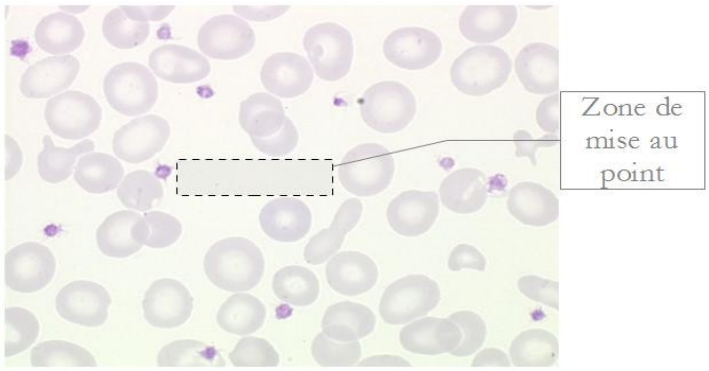

Illustration 28: La zone de mise au point de l'auto-focus n'est pas assez contrastée

Plusieurs pistes de solutions, tant *hardware* que *software*, ont déjà été testées à ce sujet. Pour rappel, C. Peeters (dans [pee07]) avait tenté deux approches différentes. La première visait à utiliser la fonction d'*auto-focus software* disponible dans l'ancienne version d'AnalySIS. Son idée était d'enclencher cet *auto-focus* pour prendre le pas sur le module *hardware* lorsque celui ci retournait une erreur. Mais les résultats obtenus n'ont pas été satisfaisant car l'*auto-focus software* était bridés à des pas de 0,2 µm, ce qui est apparemment trop grand pour un objectif 100x. La seconde solution ambitionnait de piloter directement en *RS-232* le module *U-AF* à

67 Java Native Interface

68 L'*auto-focus* ne travaille pas sur l'entièreté de l'image mais dans une zone limitée au centre de celle ci

l'aide des commandes *I-UFRS* qui lui sont propres. Le but était de pouvoir récupérer la valeur de la hauteur du plateau lors de la dernière mise au point valide (typiquement, celle de l'image précédente). Malheureusement, les commandes utiles à l'élaboration de cette solution n'étaient pas fonctionnelles et retournaient des erreurs.

X. Bocken ([boc08]) a été le dernier a tester une solution pour ce problème en se basant sur une utilisation conjointe du module *U-AF* (contrôlant l'*auto-focus hardware*) et le moteur indépendant amovible montable sur l'axe Z. Comme il n'était pas possible de récupérer la hauteur courante en interrogeant directement l'*U-AF*, son objectif était de sauvegarder les hauteurs successives du plateau en interrogeant le *Multi-Control* (sur lequel est branché le microscope) et de déplacer le plateau à l'aide du moteur amovible (branché sur le *Multi-Control*) lorsque l'*U-AF* retourne une erreur. Il n'a pu tester cette solution car cela requiert de monter à la fois le moteur de l'*U-AF* et le moteur amovible sur l'axe Z, ce qui entraine un inter-blocage (le moteur amovible exerce une pression physique sur l'axe, ce qui bloque le travail du module *U-AF*). De plus, cette solution qui consiste à déplacer le plateau avec le module d'*auto-focus* et interroger le *Multi-Control* pour récupérer la hauteur n'aurait pas pu fonctionner. En effet, lorsque l'on interroge ce dernier, il ne retourne pas la valeur réelle du plateau mais la valeur de la dernière position qui a été commandée (soit manuellement à l'aide du *joystick*, soit par commande). Cette nuance fait toute la différence. Il est donc impossible de récupérer la valeur réelle de la hauteur à laquelle se trouve le plateau si tous les mouvements ne passent pas par le M*ulti-Control*. Or, il se fait que le module *U-AF* court-circuite ce chemin en étant directement connecté au microscope (illustration 29). Dans ce cas, le *multi-control* n'a pas conscience que le plateau à bougé (vu qu'il n'a pas intercepté de commande de déplacement) et la valeur de la hauteur retournée est fausse.

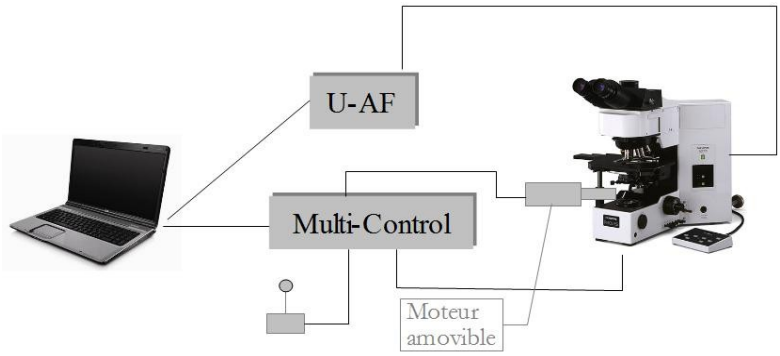

Illustration 29: Schéma simplifié du branchement entre l'ordinateur de contrôle, l'U-AF, le Multi-Control, le joystick et le microscope

Dès lors, sur base de ces constats, pour tenter d'amener un épilogue à ce problème, nous devons utiliser soit l'*U-AF* (pour une *auto-focus hardware*), soit le moteur indépendant (pour un *auto-focus software*). Comme les développements

réalisés pour la nouvelle application de pilotage impliquaient de prendre en charge les déplacements du plateau du microscope, ce fut une bonne occasion pour tester une solution de mise au point *software*. Cette solution revient à contrôler la netteté des images entièrement à l'aide du *Multi-Control* sans passer par le module *U-AF*. Elle est basée sur un principe de réglage du *focus* déterminé selon des points de références. Typiquement, la solution implémentée implique de régler la netteté de trois images dans la zone de capture à partir desquelles la netteté de l'ensemble des autres images vont être calculées par interpolation linéaire[69]. Ce principe repose sur une hypothèse fondamentale de linéarité de la lame porte-objet. Hypothèse confirmé par les experts qui utilisent des lames de bonne qualité certifiée comme étant bien plane. Les trois images qui servent de référence correspondent à trois coins du mur d'images (illustration 30).

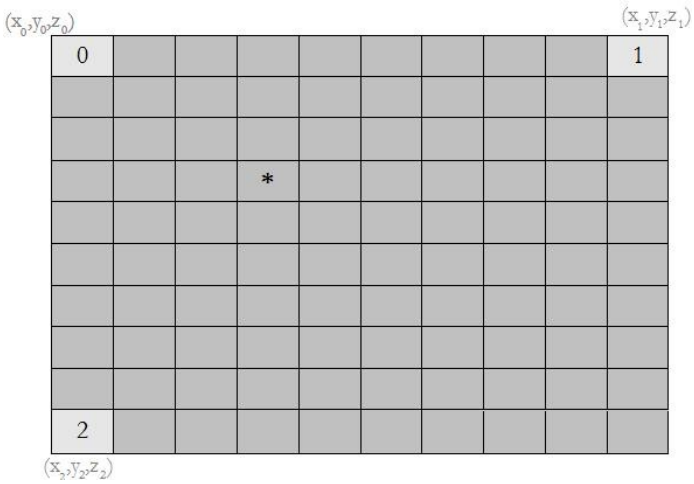

Illustration 30: Les trois images de références qui vont servir aux calculs d'interpolation linéaire

Ainsi, pour calculer la hauteur d'une image quelconque en (x_i,y_i,z_i), il convient d'appliquer la formule suivante :

$$z_i = z_0 + \left(\frac{x_i - x_0}{x_1 - x_0}\right) * (z_1 - z0) + \left(\frac{y_i - y_0}{y_2 - y_0}\right) * (z_2 - z0)$$

Une telle méthode présente des avantages et des inconvénients. Les points positifs viennent du fait que la méthode est simple, que le procédé est rapide, qu'il n'a besoin d'être effectué qu'une seule fois en début de session de capture et qu'il apporte une solution concrète au problème persistant de l'*auto-focus*. Néanmoins, cette méthode comporte certains désavantages. Premièrement, elle dépend de la qualité de mise au point manuelle réalisée sur les trois images de référence. Ensuite, elle dépend de l'homogénéité de l'échantillon. En effet, même si

69 Ces trois références vont permettre de déterminer la pente du plan suivi par la lame

la lame est parfaitement plane, l'échantillon qu'elle supporte est constitué de cellules et de plasma qui constituent une épaisseur qui peut varier en plusieurs endroits. La surface peut donc contenir des aspérités qui correspondent à des endroits marqués par une absence d'éléments ou un amas de matière[70]. Plus la taille de la zone est grande, plus ces variations peuvent être observées. Ainsi, la méthode est adaptée à des murs d'images de taille raisonnable (jusqu'à une centaine d'images). Remarquons que plusieurs tests réalisés sur des murs de 10 000 images ont également donné de bons résultats, mais moins régulièrement. Mais encore une fois, plus la taille est grande, plus les résultats deviennent dépendants de l'échantillon. Néanmoins, des améliorations du procédé sont possibles. Ces considérations seront abordées dans la partie « Discussion ».

3.3. EVALUATION EXTERNE DE LA QUALITÉ : MODÈLE

Si l'on considère l'objectif global qui vise à étudier et envisager l'élaboration d'un système de composition média dans le cadre de l'évaluation externe de la qualité, à ce stade, nous avons rempli la première moitié du travail. En effet, pour pouvoir générer du contenu de contrôle relatif au domaine de l'hématologie, il est nécessaire de pouvoir générer des méga-images. L'ensemble des considérations techniques à ce sujet ont été abordées dans la section 3.2 qui abouti sur un système semi-automatique (mais entièrement automatisable à court ou moyen terme) capable de numériser des échantillons.

La question suivante est : « Sur base des médias générés et de l'information textuelle associée, comment composer des contenus de contrôle qui puissent être portables[71], capables de recueillir les résultats saisis par les utilisateurs soumis aux évaluations et permettant un traitement efficace de ces données ? ». C'est à cette question que vont répondre les sections suivantes en présentant un modèle de représentation des contenus auquel sera associé une architecture de système adaptée à la manipulation de ce modèle.

3.3.1. IDÉE DU MODÈLE À CONCEVOIR

La demande quant à ce système de composition de contenu émane du laboratoire d'hématologie de l'hôpital de Mont-Godinne. C'est là qu'est prise en charge la réalisation technique des contenus de contrôles pour l'évaluation externe de la qualité en cyto-hématologie au niveau national. Le support informatique actuel prévu à cet effet montre ses limites et une refonte du processus de

70 Bien que la méthode de l'étalement (qui constitue le procédé par lequel un échantillon sanguin est « étalé » sur la lame porte-objet) vise une homogénéité la plus grande, ces irrégularités peuvent apparaître

71 Sous la forme de *pacakge* transportables en ligne ou physiquement, indépendamment du système d'exploitation utilisé par les acteurs soumis aux évaluations externes

composition est envisagée. Il est donc logiquement prévu que le futur système de composition dont nous discutons média soit adapté à ce domaine de la médecine.

Le support technique est fortement lié au domaine dans lequel il est utilisé. C'est pour cela que la première partie de ce chapitre concerne une station d'acquisition permettant la numérisation d'étalements sanguins ou médullaires. Cependant, une fois les médias générés et l'information qui l'accompagne rédigée, la structuration de ces données et la manière de les traiter peuvent rester générales. Sachant que l'Institut de Santé Publique (ISP), qui ordonne régulièrement les contrôles nationaux en hématologie, prévoit d'instaurer d'autres contrôles dans d'autres domaines, cette remarque prend tout son sens. En effet, dans le cas qui nous occupe, il s'agit d'imaginer un système qui prenne en charge les besoins de l'hématologie, mais si de nouveaux domaines doivent être pris en charge, comment peut-on tous les gérer ? Dans un premier temps, on peut considérer que le modèle de contenu (qui organise et structure l'information des contrôles) soit attaché à un seul domaine (en l'occurrence,l'hématologie), tout comme le système informatique dans lequel il est utilisé. Cela signifierait que le système que nous visons servirait de plate-forme pilote dédiée à l'hématologie. Sur base de celle ci, d'autres plate-formes pourraient être créées pour prendre en charge des domaines différents. Cependant, cette approche (qui consiste à développer une plate-forme différente pour chaque domaine) est accompagnée de plusieurs inconvénients.

Le fait de considérer une plate-forme de gestion des contrôles par domaine amène certains aspects moins désirables. Si l'ISP met en place plusieurs évaluations externes de qualité concernant plusieurs domaines différentes, il faut que la gestion informatique de ces évaluations toutes semblables et offrent la même qualité de service, afin de ne pas défavoriser certains domaines par rapport à d'autres. Dans ce cas, une standardisation des plate-formes de gestion des contenus de contrôle serait nécessaire, ce qui implique un consensus général entre les différents acteurs. Or, les difficultés de conciliation sur des domaines complexes peuvent rendre fastidieuse cette standardisation. De plus, dans le souci de garder une cohérence entre les plate-formes et une équivalence dans le traitement des évaluations, il faudrait contrôler les adaptations et mises à jour qui sont faites sur les différentes plate-formes et s'assurer du respect des spécifications générales (auxquelles ces plate-formes devraient se soumettre). Enfin, un aspect non négligeable est la nécessité de coder entièrement une plate-forme chaque fois qu'un nouveau domaine apparaît.

Du fait de ces difficultés, la solution qui consiste à réaliser une plate-forme par domaine fait place à une réflexion sur l'élaboration d'une plate-forme générale qui puisse accueillir n'importe quel domaine. Pour ce faire, il est nécessaire

de penser la chose avec un certain niveau d'abstraction, de manière à ne pas se borner aux spécificités d'un domaine en particulier.

3.3.2. BESOIN D'ABSTRACTION

L'abstraction en question se situe principalement autour du contenu de contrôle en lui-même. Basiquement, lorsque l'on réfléchi à ce qu'est un tel contenu (indépendamment du domaine qu'il concerne), c'est un document qui regroupe des informations sur différents thèmes et qui comporte des illustrations à l'aide de médias. Il convient donc de déterminer une structure logique portable qui puisse prendre en compte ces considérations de manière à séparer ce contenu des formes qu'il pourrait prendre selon les contextes[72]. Les travaux effectués dans le cadre du projet « *EMIM* » (voir [emim]) ont abordé cette question en se référant à différentes recherches sur la composition multimédia. La structuration logique qui va suivre et qui détermine la nature des contenus en est inspirée.

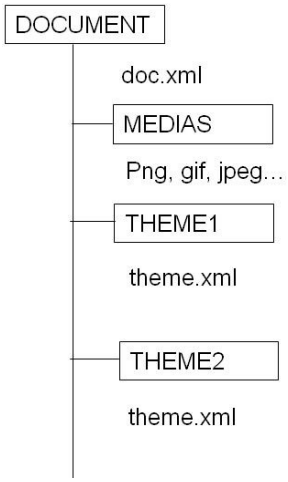

Illustration 31: Présentation de l'arborescence du « package » représentant un contenu (sans scénarios)

L'illustration 31 montre l'arborescence de base d'un contenu de contrôle. Cette structure accueille des dossiers et des fichiers de de type « *xml* » ainsi que des fichiers de type « multimédia ». Cette arborescence est la même pour n'importe quel domaine. Le point d'entrée est le dossier « Document » qui regroupe l'ensemble des sous-dossiers contenant l'information et les médias constituant un contenu de contrôle. Ce dossier contient un premier fichier « *xml* » qui va contenir toutes les informations techniques relatives au contrôle lui-même (titre, date d'échéance, langues supportées, etc). S'ensuivent une série de dossiers dont « *Medias* » qui intègre l'ensemble des fichiers multimédias utilisés et « $Theme_i$ » qui

72 Par exemple, la présentation d'images de globules blancs sous forme de galerie est typique au domaine de l'hématologie

incorpore en son fichier « *theme.xml* » l'information relative à l'un des thèmes[73] abordés par le contenu de contrôle. C'est au sein de ces derniers que va se trouver l'information propre à un domaine sur laquelle l'évaluation porte. En pratique, ce *package* qui sera transmis (sous forme de fichier compressé, par exemple au format **.zip*) aux différents laboratoires soumis aux évaluations externes de qualité. L'utilisateur le complètera par ses réponses, après quoi il sera renvoyé pour analyses.

La structure (physique) étant établie, il faut maintenant déterminer une organisation logique de l'information qui se trouve dans les différents fichiers *xml*. En effet, si l'ossature générale de la structure est invariable, la spécialisation aux domaines doit pouvoir se faire à travers les fichiers qu'elle contient. La section suivante examine cette question et présente la manière dont cette spécialisation est rélisée.

3.3.3. GESTION DES DOMAINES

Lorsque l'on parle de « spécialiser » les contenus de contrôle en fonction des domaines, cela revient à pouvoir intégrer dans ces contenus les concepts relatifs à un domaine. En d'autres termes, les fichiers *xml*[74] doivent permettre de manipuler de l'information propre à un domaine particulier. Dès lors, comment représenter tous les domaines ? Ou comment déterminer une organisation logique au sein des fichiers *xml* qui permette cela ?

Dans un premier temps, on peut considérer deux approches : une générale et une spécifique. La première consiste à mettre en place une organisation logique interne des fichiers *xml* qui puisse représenter n'importe quel domaine. Cette approche est d'entrée de jeu trop ambitieuse tant il faudrait connaître les spécificités de tous les domaines à venir. La seconde, elle, consiste à prévoir une organisation logique par domaine. Le problème de cette seconde approche est que l'on retombe dans les travers du point 3.3.1 car on perd alors la vue globale qui permet d'offrir une égalité de traitements (et de services) pour les contrôles indépendamment des domaines. En effet, la manière de traiter les contrôles sera différente en fonction du domaine et il y aurait alors autant de gestions que de domaines (à moins d'une standardisation générale, ce qui peut être fastidieux à mettre en place).

La solution vient de la fusion de ces deux principes : une organisation logique semi-générale. D'une part, il y a une base générale (contenant des types de

73 Par exemple, en hématologie, les thèmes d'une évaluation externe de la qualité peuvent être : l'informations sur un étalement sanguin ou médullaire, une galerie de globules blancs, un examen complémentaire, etc.

74 Ces fichiers sont, en quelque sorte, une représentation logique du domaine

données techniques communes à tous les domaines) sur laquelle la plate-forme base les services qu'elle est en charge d'offrir (communs, eux aussi, à tous les domaines) . D'autre part, il y des extensions qui viennent se greffer sur cette base et qui couvrent les besoins informationnels spécifiques aux domaines.

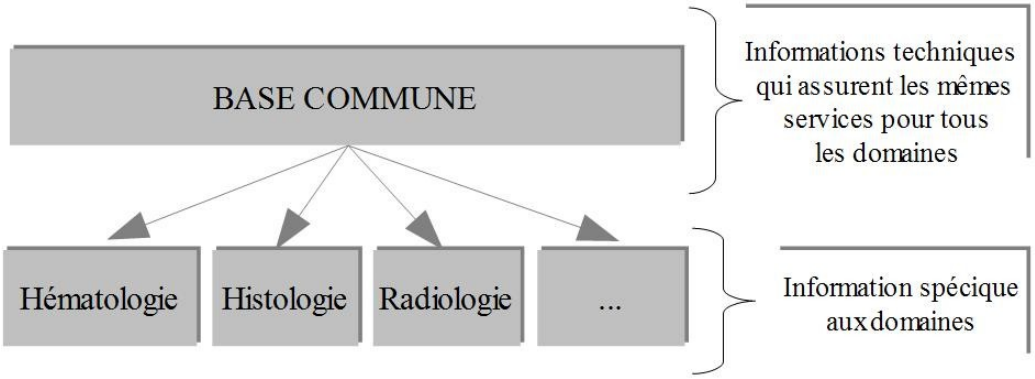

Illustration 32: Extension de la base commune du modèle de contenu

D'un point de vue technique, un moyen de décrire logiquement des fichiers *xml* est d'utiliser des fichiers de description *dtd*. Ainsi, la description logique de la base commune évoquée ci-dessus sera représentée par deux fichiers *dtd* : « *doc.dtd* » (pour décrire la structure logique contenue dans *doc.xml*) et « *theme.dtd* » (pour décrire la structure logique contenue dans *theme.xml*). De ces deux fichiers *dtd's*, on pourra étendre[75] n'importe lequel des éléments qu'ils contiennent en fonction des besoins du domaine.

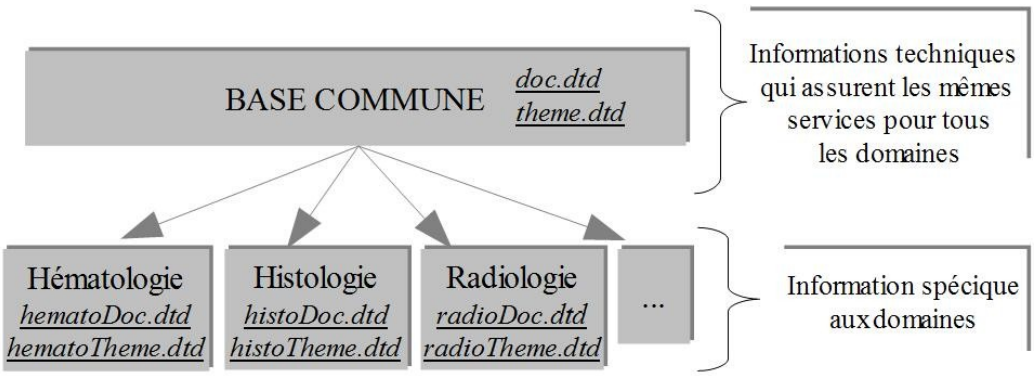

Illustration 33: Extension de la base commune du modèle de contenu, avec indication des dtd's (sans scénarios)

Dans l'illustration 33, si nous prenons l'exemple du domaine « Hématologie », les *dtd's* « *hématoDoc.dtd* » et « *hematoTheme.dtd* » étendent la structure décrite dans les fichier *dtd's* « *doc.dtd* » et « *theme.dtd* ». Il est important d'indiquer que cette extension ne peut se faire qu'en ajoutant des éléments. Il est interdit de retirer ou altérer des éléments qui existent dans les *dtd* de la base

75 En faisant de l'extension pure sans altérer les attributs initiaux des éléments étendus. Nous allons revenir sur ce point

commune[76]. C'est la seule contrainte au niveau de l'extension. Cependant, il n'est pas possible d'exprimer cette condition avec des fichiers de type *dtd*. Dès lors, dans notre exemple de l'hématologie, le fichier « *hematoDoc.dtd* » comprend le contenu du fichier « *doc.dtd* » ainsi que les éléments supplémentaires qui sont nécessaire pour les besoins d'un domaine. C'est pourquoi, il peut être intéressant d'utiliser des fichiers de type *xsd* qui incluent dans leur syntaxe des opérateurs d'extension grâce auxquels il est possible de préciser le type d'extension[77] à réaliser (dans notre cas, de l'extension pure sans restriction des éléments provenant du fichier *xsd* que l'on étend).

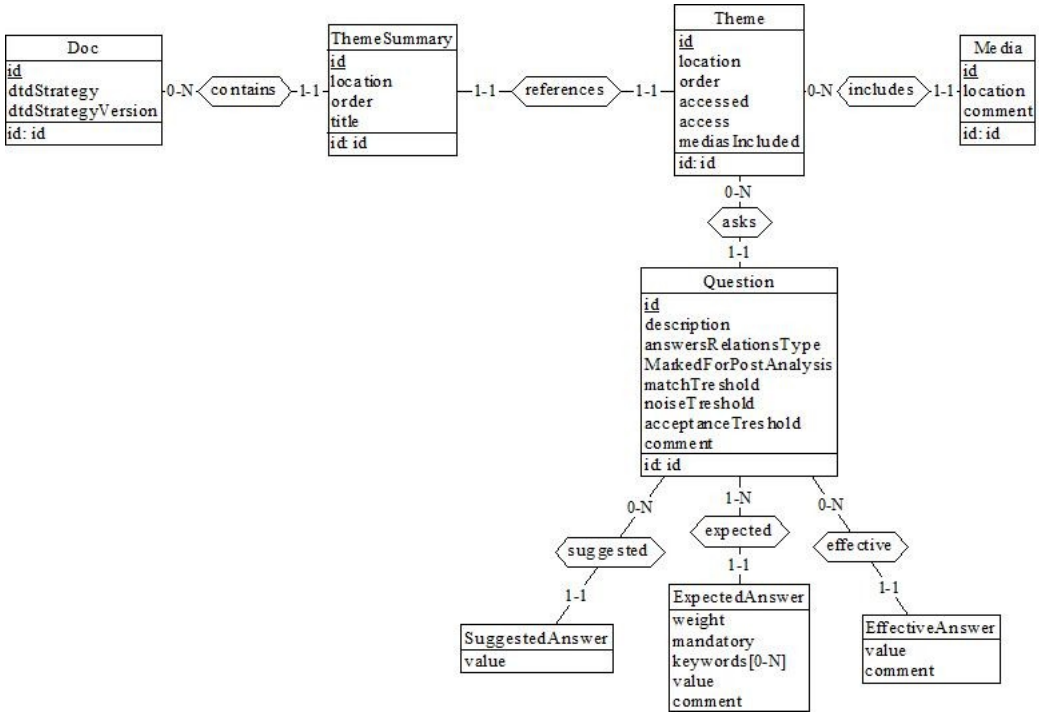

Illustration 34: Schéma ERA de la base commune du modèle de contenu (sans scénarios)

L'illustration 34 montre conceptuellement ce que contiennent les fichiers *doc.dtd* (« *Doc* », « *ThemeSummary* ») et *theme.dtd* (« *Theme* », « *Media* », « *Question* », « *SuggestedAnswer* », « *ExpectedAnswer* », « *EffectiveAnswer* »). Cette organisation est très épurée et pour cause, elle se doit d'être la plus abstraite possible. Elle représente l'information minimale dont a besoin le système pour fournir des services[78] équivalents pour tous les domaines (étant donné que tous les domaines auront cette même base).

Ainsi, dans l'absolu, on apprend qu'un document (qui représente un contenu) contient au moins un identifiant et un nom pour la stratégie (le domaine)

76 De manière à éviter les problèmes en cas d'évolution d'une *dtd*

77 Il existe deux types d'extensions : « *restriction* » qui permet de mettre des restrictions sur des types simples ou complexes existants et « *extension* » qui permet d'ajouter des éléments à des types simples ou complexes. Voir la partie « Discussion »

78 Par exemple, la génération de statistiques, la correction semi-automatique, etc

qu'il représente. On considère qu'un domaine représente une stratégie car l'implémentation déjà réalisée se base sur un *design pattern* spécifique appelé « *Strategy* [79]». En effet, le système doit pouvoir s'adapter dynamiquement en fonction du domaine choisi (ce qui représente une stratégie du *design pattern*). Un document contient également la liste des thèmes qu'il contient. Cette liste fait référence aux différents thèmes qui correspondent aux matières abordées par le contrôle de qualité.

Un thème contient un identifiant, une localisation (dans l'arborescence du contenu illustrée à la figure 31), une variable indiquant s'il a été consulté ou pas, une variable indiquant s'il y a une restriction sur la consultation du thème (« *free* » ou « *restricted* ») et une dernière variable indiquant si les médias que le thème utilise sont inclus dans le *package* du contrôle (ce qui permet d'en avoir une version *lightweight* lors de sa récupération après que l'utilisateur ait indiqué ses réponses car il n'est pas nécessaire de transmettre systématiquement les médias, d'autant que ceux-ci peuvent être nombreux et lourds). Un thème utilise des médias qui sont identifiés et possèdent une localisation dans l'arborescence. Enfin, un thème pose une série de questions relative à une des matières sur lesquelles sont évalués les utilisateurs. A ces questions sont associés plusieurs types de réponses : des réponses suggérées (qui constituent une liste[80] de réponses parmi lesquelles choisir), des réponses attendues (les réponses correctes auxquelles l'examinateur s'attend) et des réponses effectives (celle rédigées par l'utilisateur). Les questions et réponses contiennent une série de paramètres techniques utilisés dans le cadre d'une correction semi-automatique dont les principes seront expliqués plus loin dans la section 3.3.6. (« Gestion des résultats »).

Prise telle quelle, cette organisation n'est pas vraiment utilisable. En effet, comme nous l'avons vu, il faut étendre ces éléments pour en dériver une structure qui soit exploitable en fonction du domaine qui intéresse. Ainsi, si l'on considère le domaine de l'hématologie, la *dtd* qui lui est associée inclut les éléments présentés dans l'illustration 35. En reprenant la nomenclature de la figure 33, la partie supérieure du schéma conceptuel correspond au fichier *hematoDoc.dtd* et la partie inférieure (à partir de l'objet « *Theme* ») représente le fichier *hematoTheme.dtd.* On voit de nouveaux paramètres et de nouveaux objets apparaître. Par exemple, le document a maintenant un titre, des dates, il peut posséder un logo, etc. On voit également que des personnes gravitent autour de ce document : au moins un auteur, des collaborateurs (ayant aidé à sa réalisation) ainsi que des éditeurs. Ces personnes peuvent être attachées à une institution dans laquelle ils évoluent. Un thème peut maintenant contenir des illustrations qui servent à sa présentation. Ces

79 Voir annexe 6.2 pour des détails d'implémentation

80 Un sous ensemble de cette liste est considéré comme correcte et fait référence à des réponses attendues

illustrations permettent également d'agrémenter les questions qui sont posées aux utilisateurs. On retrouve donc tout ce qui existait dans la base commune avec, en plus, toute l'information requise pour les besoins de la gestion du domaine de l'hématologie. Remarquons enfin que la *dtd* de ce domaine peut sembler abstraite. C'est effectivement le cas et cela a été fait volontairement. Le but a été d'imaginer une *dtd* qui puisse représenter plusieurs domaines. L'objectif visé étant de limiter au maximum le travail d'intégration des nouveaux domaines à la plate-forme. En effet, s'il est possible de réutiliser cette *dtd*, le travail d'intégration est minimal car la plate-forme est déjà techniquement capable de gérer des contenus organisés selon cette *dtd*[81]. Si cette prise en charge n'est pas possible, alors il faudra procéder de manière standard en créant des *dtd* spécifiques qui étendent la base commune. Pour information, dans le code déjà rédigé, la stratégie de *dtd* implémentée (qui accueille le domaine de l'hématologie) s'appelle « *générale* », en référence à cet objectif de réutilisation.

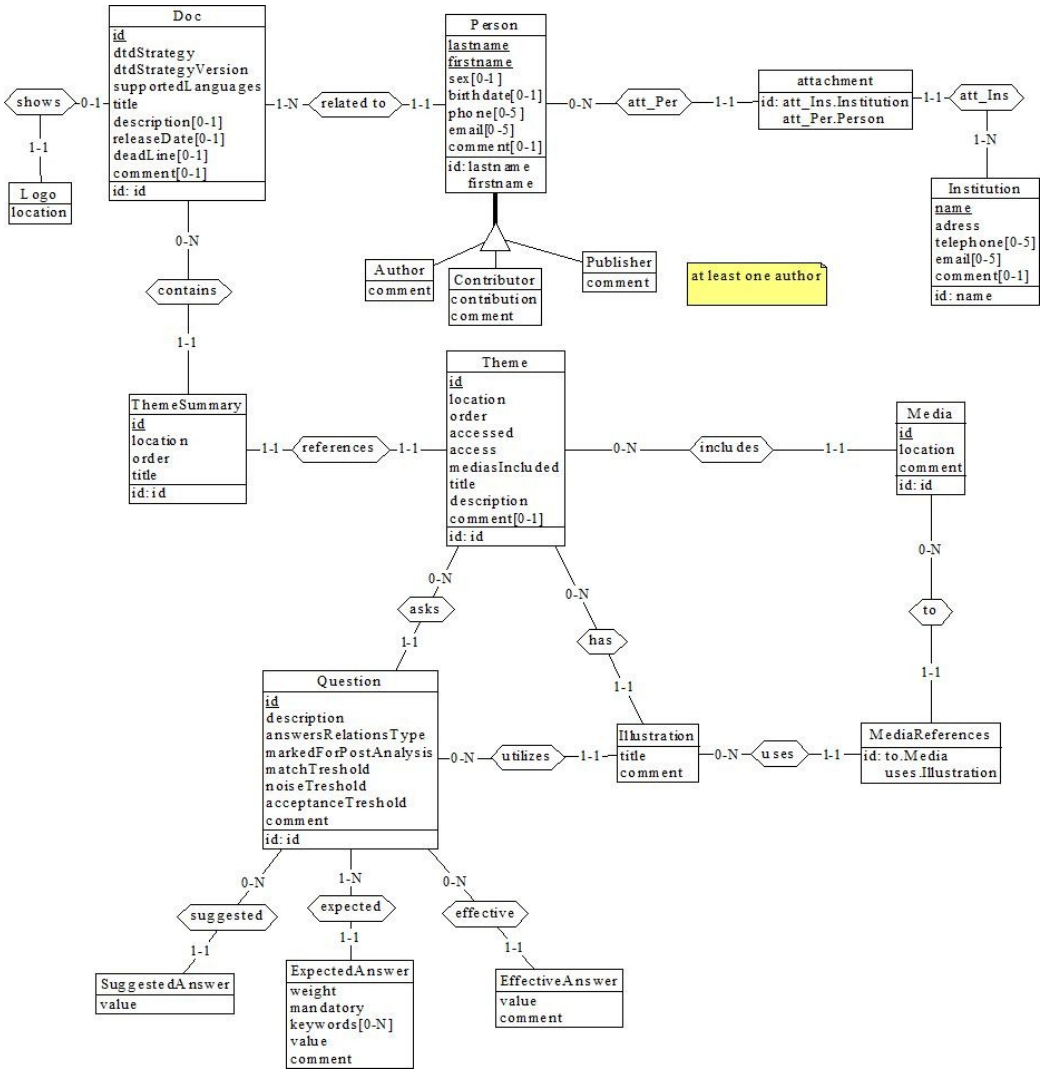

Illustration 35: Schéma ERA de la stratégie DTD pour l'hématologie (ou stratégie générale) (sans les scénarios)

81 Il faudra tout de même veiller à fournir un template adéquat (voir la section 3.3.7)

Ainsi, nous obtenons un modèle de représentation logique de l'information adaptable à différents domaines. Mais il est une autre adaptation dont il faut parler : celle au contexte de visualisation. En effet, si nous reprenons l'exemple de l'hématologie, lors de l'analyse des besoins il a été précisé, par exemple, que les futures contrôles puissent être embarquables sur CD-ROM et, à terme, consultables en ligne. Dans le premier cas, c'est une application qui affichera le contenu de l'évaluation et dans le second cas, c'est un navigateur qui s'en chargera. Le contenu logique, lui, reste le même pour les deux contextes. Dès lors, il faut compléter le modèle en incluant des opérations de transformation de ce contenu logique en un contenu adapté présentable en fonction du contexte de visualisation. Nous quittons le monde du fond pour la forme.

3.3.4. SCÉNARIOS ET TRANSFORMATIONS

Adapter un contenu logique en fonction d'un contexte revient à créer une manifestation de ce document qui intègre les dimensions spatiales, temporelles et hypermedia associées aux éléments qu'il contient. Cette adaptation est envisagée à l'aide de scénarios qui sont adjoints aux contenus logiques. Ceux ci renferment une description des transformations à appliquer pour obtenir une présentation adaptée du document. Ces transformations de contenu portent sur les différents fichiers *xml* et le moyen typique de transformer ce type de fichier est d'utiliser des feuilles de styles *xsl*. En d'autres termes, un scénario contient une description des transformations à réaliser (sous la forme d'un fichier *scn.xml*) et les feuilles de transformations qu'il utilise (**.xsl*). Il a donc un nouveau dossier qui apparaît dans l'arborescence (de l'illustration 31) et qui contient l'ensemble des scénarios applicables au contenu logique. La figure suivante montre l'existence d'un scénario adaptant le contenu à une visualisation depuis un navigateur Internet classique (en convertissant les fichiers au format *html*).

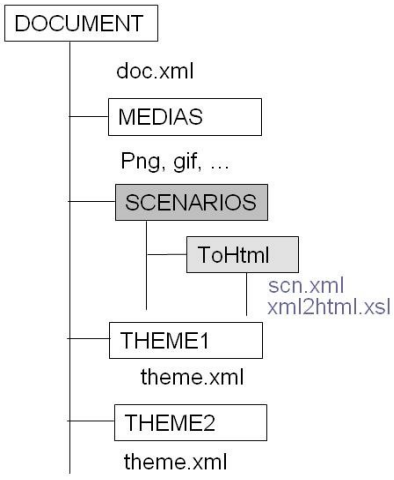

Illustration 36: Présentation de l'arborescence du « package » représentant un contenu avec les scénarios

Remarquons que la structuration des documents EMIM ([emim]) placent les scénarios à l'intérieur même des différents thèmes (appelé « *Pages* » dans leur modèle), ce qui constitue la différence majeure entre un modèle de document EMIM et celui ci. Cela vient du fait que les domaines d'application sont différents. En effet, dans le cadre du projet EMIM, les utilisateurs consultent des documents médicaux et vont vers la page[82] qui concerne leur spécialité (IRM, scanner, ...). La consultation est donc ciblée et les transformations appliquées pour adapter la présentation se font localement sur la page consultée. Dans le cadre d'une évaluation externe de qualité, le document de contrôle représente un ensemble moins divisible. En effet, lorsqu'un contrôle est envoyé, c'est généralement dans sa totalité et les utilisateurs ne sont pas limités à une partie de celui ci (que du contraire, ils doivent en traiter tous les thèmes). Il convient donc d'inclure des scénarios qui permettent d'adapter la présentation de l'ensemble des pages (ou « thèmes »). Cependant, il faut conserver une cohérence dans les transformations qui sont appliquées. En effet, si une transformation *T1* est appliquée sur la page *P1*, alors il faut également appliquer *T1* sur les autres pages P_i car le contexte est le même pour toutes les pages (ce qui n'est pas le cas dans le domaine d'application du projet EMIM). C'est pour ces raisons que, dans le cas qui nous occupe, les scénarios sont remontés au niveau du document général plutôt qu'au niveau des thèmes. Cela offre une vue globale de la portée des scénarios sur les différents thèmes (on peut même envisager de créer des scénarios qui génèrent des manifestations sur un sous-ensemble de pages).

Si l'on considère l'illustration 33, on s'imagine qu'une description des fichiers *scn.xml* sous la forme d'une *dtd* va apparaître dans la base commune, avec des extensions pour les différents domaines. C'est partiellement vrai. En fait, les scénarios représentent la structuration logique d'une information (en l'occurrence, la manière d'opérer des transformations) qui est indépendante[83] des domaines. Ainsi, les scénarios sont des objets non extensible. Tous les domaines travaillent

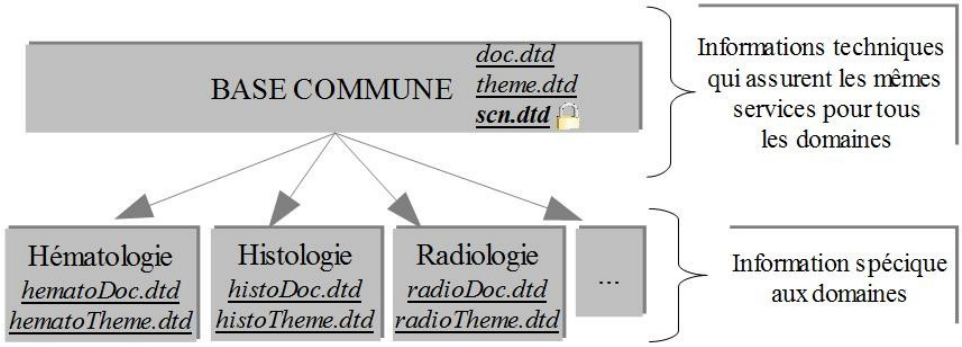

Illustration 37: Extension de la base commune du modèle de contenu avec les scénarios (qui sont non extensibles)

82 Au sens « EMIM » du terme, ce qui correspond aux thème pour contenu de contrôle. Dans la discussion qui va suivre, « theme » et « page » représentent le même concept

83 Les transformations (dans les fichiers *xsl*) sont liées aux domaines, mais les fichiers *scn.xml* qui décrivent un scénario utilisant ces fichiers *xsl* est le même pour tous les domaines

sur la même représentation logique de scénarios.

Voyons maintenant à quoi ressemble, conceptuellement, la base abstraite commune. L'illustration 38 montre le schéma de la figure 34 complété par les scénarios. On y découvre qu'un scénario possède des attributs standards tels qu'un identifiant ou le nom de son auteur, mais également des informations de *post-processing* et de *pré-processing* du document (nous en parlerons plus loin en illustrant la figure 40). Il contient également des informations de transformation sur des thèmes. Ces transformations prennent des paramètres tels que le type de transformation à effectuer (*xsl*, *fo* ou *smil*, comme illustré sur le schéma 39), le nom du document en entrée, le nom qu'il prendra en *output* et la feuille de style à utiliser. Il est possible de passer des paramètres à cette feuille de style ou encore d'indiquer s'il existe des liaisons entre les thèmes. Remarquons enfin que, puisque la description logique des scénarios est non extensible, elle se retrouve telle quelle dans la description étendue relative à un domaine. En d'autres terme, l'ensemble rouge (scénarios) dans le schéma ci-dessous se retrouve inséré intégralement dans le schéma 35.

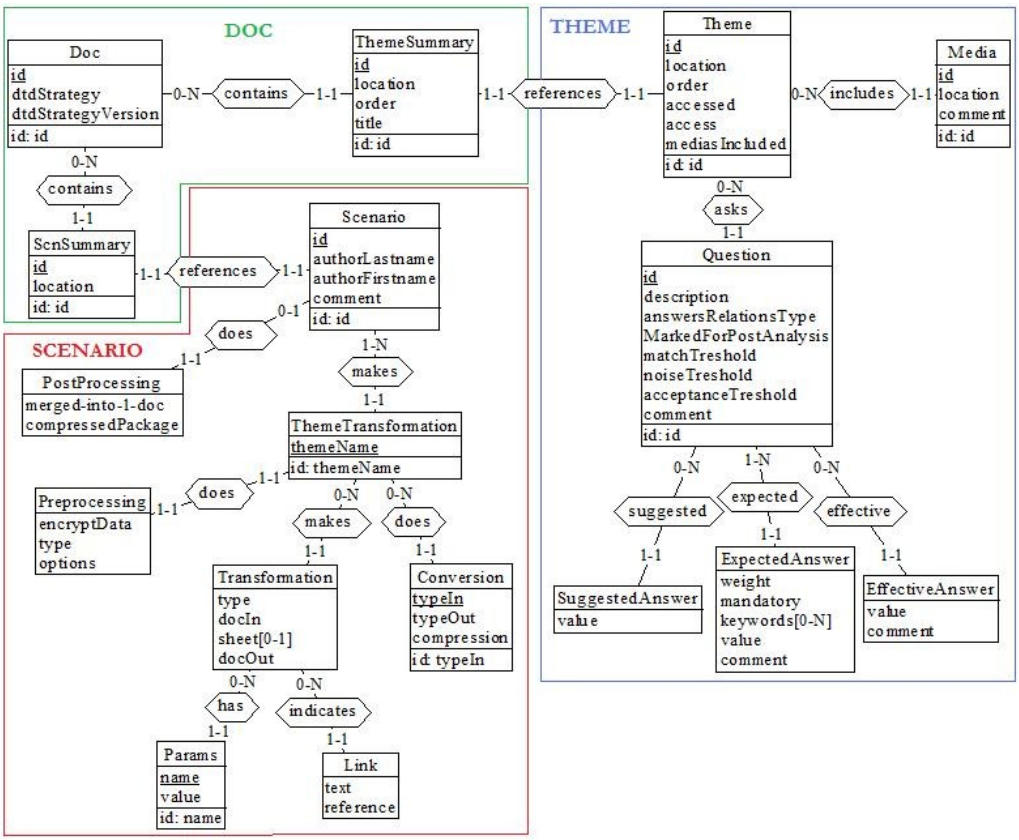

Illustration 38: Schéma ERA de la base commune du modèle de contenu

L'illustration 39 montre le schéma général des transformations. A gauche se trouve la partie abstraite avec le document logique général et à droite la

partie concrète avec les manifestations possibles qui sont présentables sur différents types de visualiseurs. Le document peut se décliner en une multitude de manifestations variées grâce à *xsl-fo*[84] (qui permet la mise en forme de documents *xml* quelque soit le support : écran, papier, etc) et *smil*[85] (qui permet de synchroniser des contenus multimédias afin de permettre la création de présentations multimédias). Ainsi, il est également possible de transformer un fichier *xml* en fichier **.fo* ou **.smil* (à l'aide de feuilles *xsl*) que l'on transmet ensuite à un moteur approprié qui se chargera de faire la conversion dans le format de sortie souhaité (sur base des spécifications contenues dans ces fichiers). En fonction du visualiseur et du type de manifestation réalisée, différentes dimensions sont possibles pour la visualisation. La dimension interactive est la plus intéressante dans le cas des évaluations externes de qualité car elle permet de récupérer des réponses rédigées par l'utilisateur.

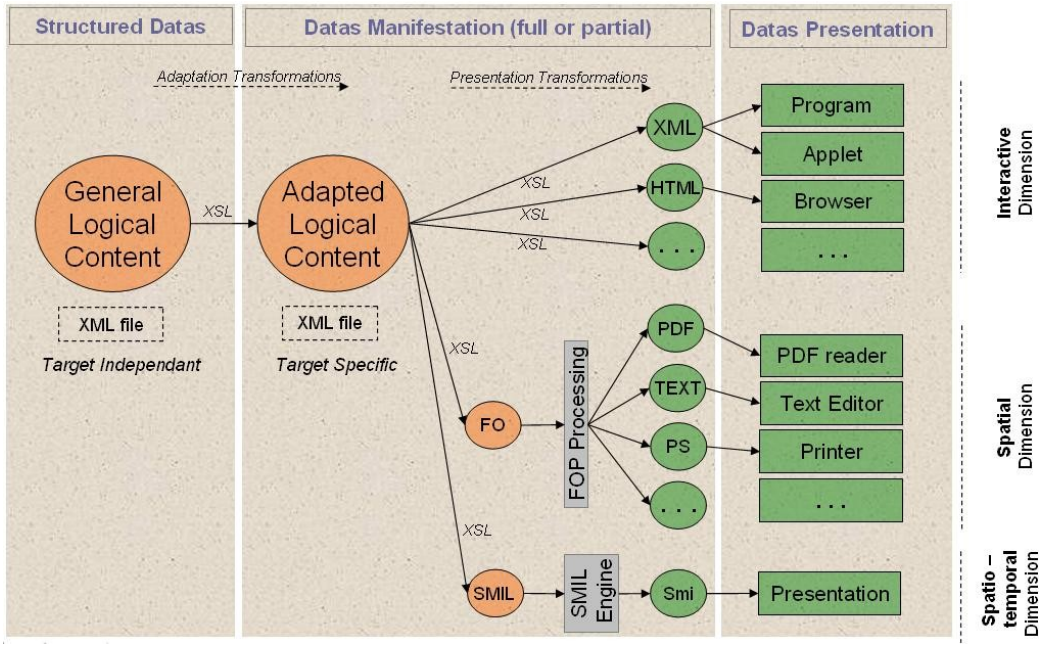

Illustration 39: Schéma général de transformation des contenus logiques

La figure 40 montre une autre vue de ces transformations mettant en évidence les opérations de *pre-processing* et de *post-processing* : la première opération porte sur le contenu des thèmes et la seconde porte sur une manifestation. Ainsi, bien que les scénarios portent principalement sur la forme, il est également possible d'indiquer des opérations bien déterminées à effectuer sur le contenu. Ceci permet de rendre le modèle encore plus général : les scénarios peuvent porter sur le contenu et la forme. Pour l'instant, au niveau du *pre-processing*, la seule opération envisagée est le cryptage[86] des solutions qui apparaissent dans le contenu logique général, ce qui amène à un contenu logique adapté (c'est-à-dire, adapté à la

84 *Formatting Objects*
85 *Synchronized Multimedia Integration Language*

transmission aux utilisateurs, auxquels il faut naturellement masquer les solutions). Ce contenu logique adapté sera ensuite mis en forme selon le format de sortie désiré. C'est vraiment au niveau du *pre-processing* que l'on peut agir sur le contenu même du *package* car cette opération porte sur les thèmes. Le *post-processing*, quant à lui, porte sur une manifestation, ou plus précisément, une pré-manifestation, c'est-à-dire un contenu logique adapté sur lequel on vient d'appliquer les transformations de présentation (comme indiqué sur les schéma ci-dessous). Actuellement, les opérations de *post-processing* considérées sont la fusion des thèmes (par exemple, pour permettre la fusion en un seul document comme, par exemple, un fichier *pdf*, un site avec une seule page générale, etc) et la compression du *package* [87](pour le rendre transportable).

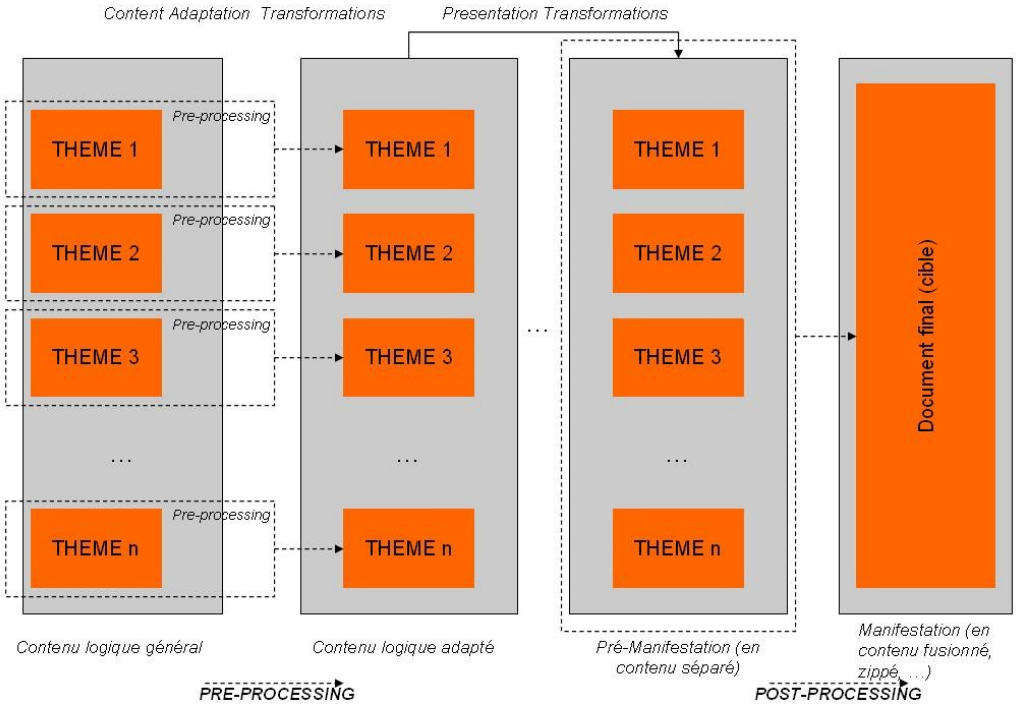

Illustration 40: Mise en évidence des opérations de pré/post-processing des contenus logiques

En conclusion, nous obtenons donc un modèle suffisamment général qui permet de représenter de l'information spécifique à un domaine par extension d'une base abstraite qui, elle, permet une gestion intégrée des contenus dont il est possible d'adapter la contenance ou encore la forme en fonction du contexte de visualisation.

86 De manière à ce que les utilisateurs ne puissent consulter les réponses lorsqu'ils sont soumis à un contrôle

87 Le format **.zip* est envisagé

3.3.5. ARCHITECTURE D'UN SYSTÈME EXPLOITABLE

Cette section présente une architecture idéale de système dans laquelle puisse être utilisé le modèle présenté précédemment. Celle ci se décompose en trois parties distinctes illustrées sur le schéma de la figure 41.

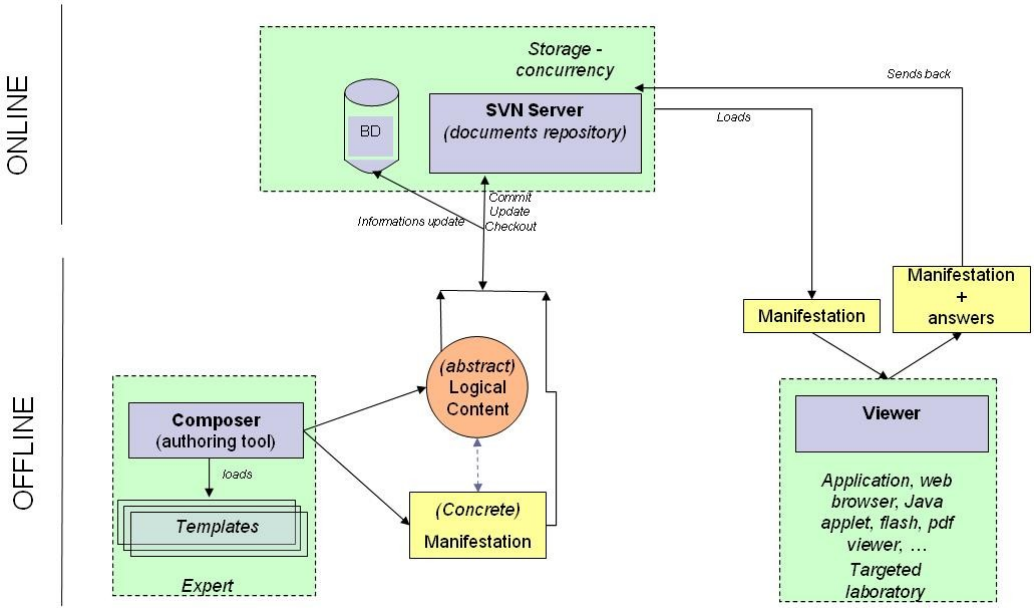

Illustration 41: Architecture idéale pour supporter le modèle

A gauche se trouve la partie abstraite avec la composition des contenus logiques généraux. Le *composer* indiqué dans le schéma correspond à la plate-forme de composition dont nous avons discuté. C'est à partir de celle ci que sont générées des représentations logiques de contenus. A ces documents sont associés des scénarios qui permettent d'en générer des manifestations qui soient exploitables en fonction des contextes de visualisation. La plate-forme est équipée de *templates* dont le but est de masquer à l'utilisateur la complexité générale liée à l'abstraction (voir la section 3.3.7). Les développements qui ont déjà été réalisés portent principalement sur le *composer*. Cette première étape a porté sur les représentations logiques des contenus et leur modélisation sous forme d'objet (au sens « *Java* ») exploitables. L'architecture, quant à elle, s'articule autour de deux *design patterns* (« stratégie » et « singleton »). Des détails supplémentaires sont consultables à l'annexe 6.2.

La partie droite représente la partie concrète du système. Elle est composée des différents visualiseurs possibles. Il existe beaucoup de types de visualisations différentes (pour lesquels les contenus logiques doivent être adaptés) : navigateurs Internet classiques (interprétant de l'*html*), *applets Java*, *viewers Flash*, *pdf readers*, applications *standalone*, etc. En fonction des possibilités d'interactivité du visualiseur et de la nature des manifestations, il est possible de

compléter les contenus par les réponses des utilisateurs, ce qui permet alors de récupérer directement le *package* pour analyses. Remarquons que, pour les besoins de l'évaluation externe de la qualité en hématologie, il sera nécessaire de développer une application *Java* qui puisse afficher les contenus selon les modalités de présentation décidées.

Enfin, au sommet se trouve la composante « réseau » du système. Un serveur accueille et stocke les différents contenus logiques généraux créés. Un système de gestion de versions (*svn*[88]) prendra en charge les transactions (les avantages sont abordés à la section 3.3.8). Une base de donnée permettra d'organiser correctement les contenus et l'information qui leur est associée (identifiant, date de création, date de dernière modification, etc). Deux approches sont donc possibles pour la transmission des contenus : soit en ligne via le réseau (courrier électronique, applications *client-serveur*, etc), soit physiquement (clé USB, CDROM, etc). Des détails supplémentaires seront abordés dans la section 3.3.8.

3.3.6. GESTION DES RÉSULTATS

Les résultats émanant des utilisateurs représentent l'information la plus importante dans le cadre d'une évaluation externe de la qualité. En effet, les réponses qu'ils rédigent représentent leur interprétation du cas qui leur est transmis. C'est cette interprétation qu'il convient d'étudier et dont il faut déterminer la qualité. Cependant, le nombre de contrôles à corriger peut être élevé[89] et ceux ci peuvent contenir beaucoup de questions[90]. Il est donc essentiel d'avoir un support informatique aidant à la gestion de ces résultats. De plus, les réponses aux questions qui sont posées dans le cadre de ces évaluations ne sont pas forcément binaires. Elles peuvent être le fruit d'une interprétation qui amène plusieurs solutions possibles. Il y a donc une certaine flexibilité à introduire dans l'évaluation de ces réponses. La gestion qui va être proposée dans les paragraphes suivants répond à ces considérations et permet une correction semi-automatique des contenus.

Cette gestion est réalisable grâce à l'information qui est associée tant aux questions qu'aux réponses dans le modèle de représentation des contenus. Comme le montre l'illustration 38, ces informations font partie de la base générale commune. Ce qui signifie que tous les domaines disposent de cette gestion des

88 Abréviation de « *Subversion* »

89 Si l'on considère le cas du contrôle nationale en hématologie, il y a plus de 300 laboratoires concernés, ce qui fait autant de contenus à corriger par la suite

90 Par exemple, un thème classique des contrôles en hématologie est la galerie de globules blancs dans laquelle il faut identifier le type des cellules qui la composent. Ainsi, selon le modèle mis en place, chaque identification fait l'objet d'une question. Or cette galerie porte sur plusieurs centaines de cellules (idéalement 200 pour un échantillon représentatif), ce qui fait autant de questions à gérer

résultats. Il y a un *mapping* complet entre les questions et les réponses retournées par les utilisateurs. Ainsi, par exemple dans le cas de l'hématologie, dans l'identification des types de cellules d'une galerie de globules blancs, il est possible de faire de l'analyse des réponses cellule par cellule. Cet exemple le prouve à nouveau : il se peut qu'un contenu contienne beaucoup de questions. Dès lors, un attribut « *markedForPostAnalysis* » existe dans une question pour signaler que la réponse associée est déterminante dans l'élaboration du diagnostic. Ce qui permet une mise en évidence et l'élaboration éventuelle de statistiques (sur l'ensemble des contenus transmis) ciblées sur certaines questions. Les autres attributs sont utilisés pour la correction semi-automatique que nous allons décrire ci-après.

Dans l'absolu, pour déterminer si une question a été répondue correctement, il suffit de comparer la valeur contenue de la réponse attendue (*expected answer*) avec celle de la réponse effective (*effective answer*). Ce système est efficace si l'utilisateur ne doit pas rédiger lui même la réponse mais la choisir dans une liste, par exemple (grâce aux *suggested answers*). Dans le cas contraire, lors de la correction, il faudra effectuer la comparaison sur base des mots-clés définis dans le champ « *Keywords* » d'une réponse attendue. Naturellement, pour ces questions (dont les réponses sont rédigées à la main), une vérification humaine sera nécessaire (d'où l'utilité de les marquer pour les mettre en évidence et faciliter le travail d'analyse des résultats, par exemple). Remarquons que cette correction n'a pas pour but d'attribuer une cote à la question, mais de mettre en évidence les questions dont les réponses effectives ne correspondent pas à ce qui était attendu[91].

De part la nature des réponses à fournir et la complexité des domaines, le modèle doit pouvoir être suffisamment général pour permettre de représenter une majorité de cas possibles (« Qui peut le plus, peut le moins »). Ce service de correction fait partie de la base générale commune du modèle et doit donc être capable de supporter les différents domaines[92]. C'est pour ces raisons qu'une question peut avoir plusieurs réponses. Dans ce cas, il convient d'analyser les correspondances qui peuvent exister entre les réponses attendues et les réponses effectives. Trois types de relations sont considérés :

91 D'ailleurs, rien n'est « corrigé » (au sens classique du terme), ce n'est qu'une évaluation de réponses qui permet aux examinateurs de se focaliser sur les questions qui ont posé problème

92 Il est possible d'étendre les concepts de « questions » et de « réponse » pour les besoins informationnels d'un domaine, mais la correction reste prise en charge par la plate-forme. La plate-forme ne connait que cette base, les extensions sont prises en charges par les *templates* (voir la section 3.3.7)

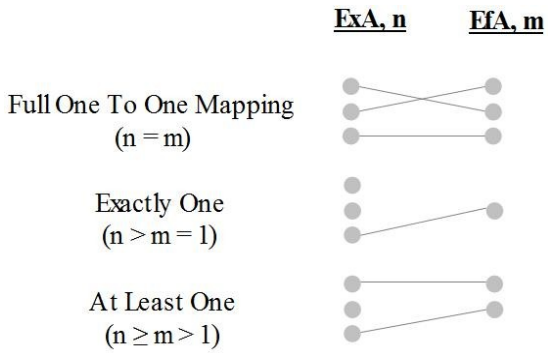

Illustration 42: Schéma des correspondances possibles entre les réponses attendues et les réponses effectives

Le type de relation entre les réponses attendues et effectives est matérialisé par l'attribut « *answersRelationsType* » de l'élément « *Questions* » dans le modèle logique de représentation des contenus. Dans l'illustration 42, « *ExA* » représente les réponses attendues (*expected answers*) et « *EfA* » les réponses effectives (*effective answers*). Le premier cas permet aux examinateurs de considérer que l'utilisateur doit exactement donner toutes les réponses auxquelles ils s'attendent. Dans le second cas, l'utilisateur ne peut se prononcer qu'une seule fois pour la question, mais les examinateurs prévoient plusieurs cas de réponses acceptables. Les pondérations qui apparaissent dans l'attribut *weight*[93] du type d'entité « *ExpectedAnswer* » (illustration 38), leur permettent d'exprimer leur préférence parmi les possibilités. Pour ces deux premiers cas, la correction est assez simple à mettre en œuvre car il suffit de vérifier que toutes les réponses effectives ont une correspondance[94] avec une réponse attendue. Dans le troisième cas, les examinateurs permettent à l'utilisateur de donner plusieurs réponses. Ce dernier type de relations permet des configurations de réponses plus subtiles. Parmi les réponses qui sont attendues, certaines ont plus de poids que d'autres. Les examinateurs s'attendent à ce que les utilisateurs se focalisent sur les réponses qui ont le plus de poids (et qui sont les plus pertinentes vis-à-vis de la question). Même si les réponses de moindre poids ne sont pas fausse, au bout d'un certain nombre, il se peut qu'elles génèrent du bruit et nuisent à la pertinence de la réponse donnée par l'utilisateur. C'est par exemple le cas des examens complémentaires en hématologie : lors de l'élaboration d'un diagnostic, il est généralement demandé si des examens complémentaires doivent être réalisés et, en pratique, on cherche à en déterminer un nombre minimal qui soit le plus pertinent avec l'éventuelle pathologie suspectée. Même si des examens complémentaires ne sont pas une chose négative en soi, certains peuvent se révéler inutiles, moins intéressants, secondaires. Cela signifie qu'au bout d'un certain nombre, ils nuisent à la pertinence

93 Bien que, dans ce cas de figure, toutes les réponses soient bonnes, ce système permet de mettre en évidence des réponses particulièrement bonnes. Rappelons que ce système sert plus à mettre en évidence des réponses qu'à les corriger

94 Dans le premier cas, la correspondance doit être totale, dans le second elle une correspondance unique suffit à valider la réponse

de l'évaluation du problème et génèrent une information qui s'apparente plus à du bruit. Ainsi, dans le troisième cas du schéma 42, seul un sous-ensemble des réponses attendues (celles de poids maximal) représente la réponse idéale à laquelle s'attendent les examinateurs. Il n'empêche que d'autres configurations acceptables de réponses existent et sont proches de cet idéal. Il y a donc une zone d'acceptabilité autour d'un optimum qu'il convient de modéliser.

Pour faciliter la détermination du sous ensemble optimal, les réponses qui le composent seront qualifiées « d'obligatoires[95] », les autres seront qualifiées de « facultatives». En fonction des domaines d'applications, ces réponses facultatives peuvent soit servir la question (et permettre de compenser partiellement l'absence de certaines réponses attendues), soit générer du bruit au bout d'un certain nombre (par exemple, lorsque l'utilisateur a déjà donné l'ensemble des réponses attendues). Ainsi, comme l'illustre le schéma 43, l'ensemble de *n* réponses attendues se décompose en un sous ensemble de *p* réponses obligatoires et *k* réponses facultatives (avec $p+k=n$). Le caractère obligatoire ou pas de la réponse est matérialisé dans le modèle par l'attribut « *mandatory* » du type d'entité « *ExpectedAnswer* » (illustration 38).

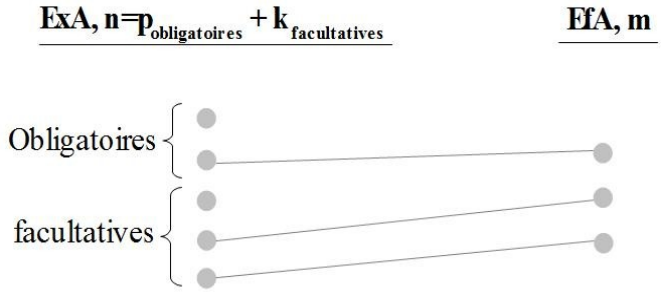

Illustration 43: Décomposition de l'ensemble des réponses attendues

Pour représenter la zone d'acceptabilité, deux variables vont être utilisées : α pour marquer la borne supérieure (qui correspond au seuil du bruit) et γ pour marquer la borne inférieure (qui correspond au seuil minimal d'acceptabilité). Dans le schéma 44, « *MT* » (*match treshold*) représente l'optimum qui correspond à la réponse idéale (composée exactement de toutes les réponses attendues obligatoires). Il est matérialisé dans le modèle par l'attribut du même nom. α et γ, quant à eux, sont représentés par les attributs « *noiseTreshold* » et « *acceptanceTreshold* » (respectivement).

95 C'est-à-dire indispensables (obligatoires) pour avoir une réponse idéale. S'il manque une de ces réponses attendues, alors on aura au mieux une réponse acceptable

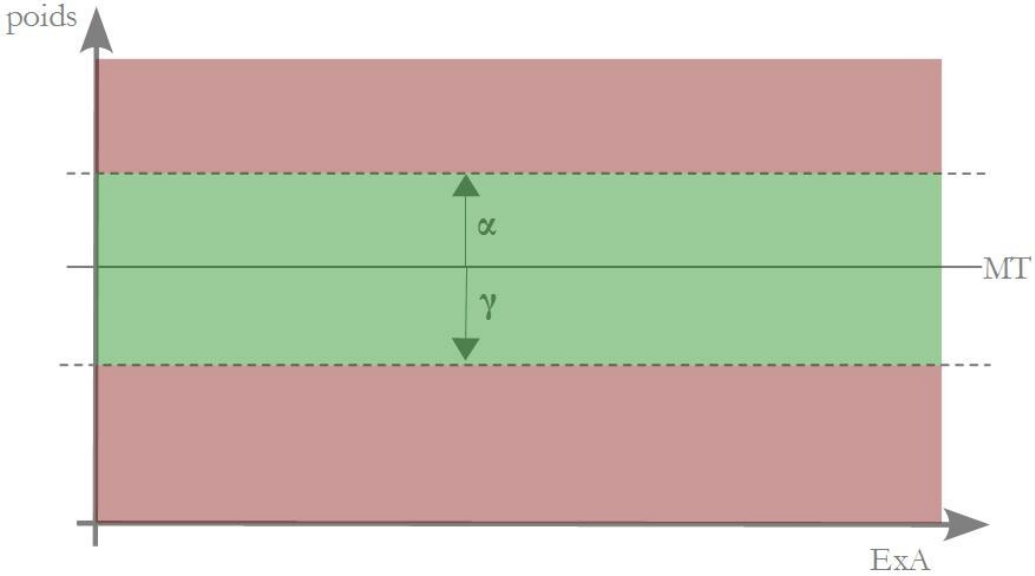

Illustration 44: Schéma de la zone d'acceptabilité autour de l'optimum

Voyons maintenant quelles sont les définitions et contraintes de ce système. Les notations « $ExAo_i$», « $ExAf_i$» représentent respectivement une réponse attendue obligatoire et une réponse attendue facultative. « EfA_i» représente une réponse effective donnée par l'utilisateur. Nous considérons n réponses attendues dont p obligatoires et k facultatives. L'utilisateur fournit m réponses effectives.

- $\sum_{i=1}^{p} poids(ExAo_i) = MT$: la réponse idéale est constituée des réponses attendues obligatoires, ni plus, ni moins
- $\sum_{j=1}^{m} EfA_j \equiv \sum_{i=1}^{p} poids(ExAo_i) * I_{o_i} + \sum_{j=1}^{k} poids(ExAf_j) * I_{f_j}$: la valeur totale des réponses données par l'utilisateur vaut la somme des poids des réponses attendues obligatoires et facultatives auxquelles elles correspondent

  où $I_{o_i} = 1$: si la réponse attendue obligatoire a été choisie, 0 sinon

  et $I_{f_j} = 1$: si la réponse attendue facultative a été choisie, 0 sinon

Les deux équations suivantes indiquent les contraintes d'acceptabilité d'un jeu de *EfA* donné :

- $\sum_{j=1}^{m} EfA_j \leqslant MT + \alpha$, où $\alpha \geqslant 0$: la valeur des réponses données doit se situer sous le seuil de bruit

- $\sum_{j=1}^{m} EfA_j \geqslant MT - \gamma$, où $\gamma \in [0, MT]$: la valeur des réponses données doit se situer au dessus du seuil d'acceptation

Comme seule la somme des poids des *ExAo* représente l'optimum, il faut poser une contrainte supplémentaire sur les *ExAf* de manière à ce que celles ci ne permettent pas d'atteindre cet optimum s'il venait à manquer une *ExAo* :

- $\sum_{i=1}^{p-1} poids(ExA_{o_i}) + \sum_{j=1}^{k} poids(ExA_{f_j}) < MT$

De cette formule, on peut déduire une contrainte sur les poids :

- $\sum_{j=1}^{k} poids(ExA_{f_j}) < poids(ExA_{o_i})$, le poids total que peuvent représenter les *ExAf* doivent être inférieures au poids de la *ExAo* minimale. Nous considérons que toutes les *ExAo* ont un poids unitaire,
- comme : $\forall i, p \in \mathbb{N}_0, p \geqslant i > 0 : poids(ExA_{o_i}) = 1$,
- alors : $\sum_{j=1}^{k} poids(ExA_{f_j}) < 1$

Il reste une dernière contrainte à poser sur α : si l'utilisateur a atteint l'optimum en fournissant toutes les *ExAo*[96], s'il rajoute en plus toutes les *ExAf*, il doit dépasser le seuil de bruit, dès lors :

- $\sum_{i=1}^{p} poids(ExA_{o_i}) + \sum_{j=1}^{k} poids(ExA_{f_j}) \geqslant MT + \alpha$
- donc : $\sum_{j=1}^{k} poids(ExA_{f_j}) \geqslant \alpha$

Remarquons que cette double somme peut néanmoins être égale au seuil de bruit sans le dépasser. Ce cas de figure représente une situation d'un domaine d'application dans lequel les réponses facultatives n'ont pas de valeur pénalisantes : plus on en met, plus cela sert la question.

Si l'on observe les valeurs que peuvent prendre les deux paramètres α et γ, certaines nuances apparaissent sur les contraintes qu'elles représentent :

96 Entendons-nous : il a fournit des *EfA* dont les valeurs correspondent aux *ExAo*

- $\alpha = 0$, contrainte stricte, aucun bruit n'est toléré
- $\alpha > 0$, contrainte douce, on tolère un peu de bruit
- $\alpha = \sum_{j=1}^{k} poids(ExA_{f_j})$, contrainte nulle, les *ExAf* ne génèrent pas de bruit
- $\gamma = 0$, acceptation stricte (compensation nulle)
- $\gamma > 0$, acceptation douce (compensation possible)

Cependant, il est difficile de comparer α et γ entre eux. D'autant que lorsque le seuil d'acceptabilité minimum est placé bas (distant de plus d'une *ExAo* de *MT*), γ sera bien plus grand que α et leur comparaison n'aura pas de sens. En fait, ce qui est intéressant à comparer, c'est la couverture des *ExAf* en terme de compensation par rapport au bruit qu'elles génèrent (α). Dès lors, il est plus judicieux de travailler avec γ', comme indiqué sur l'illustration suivante.

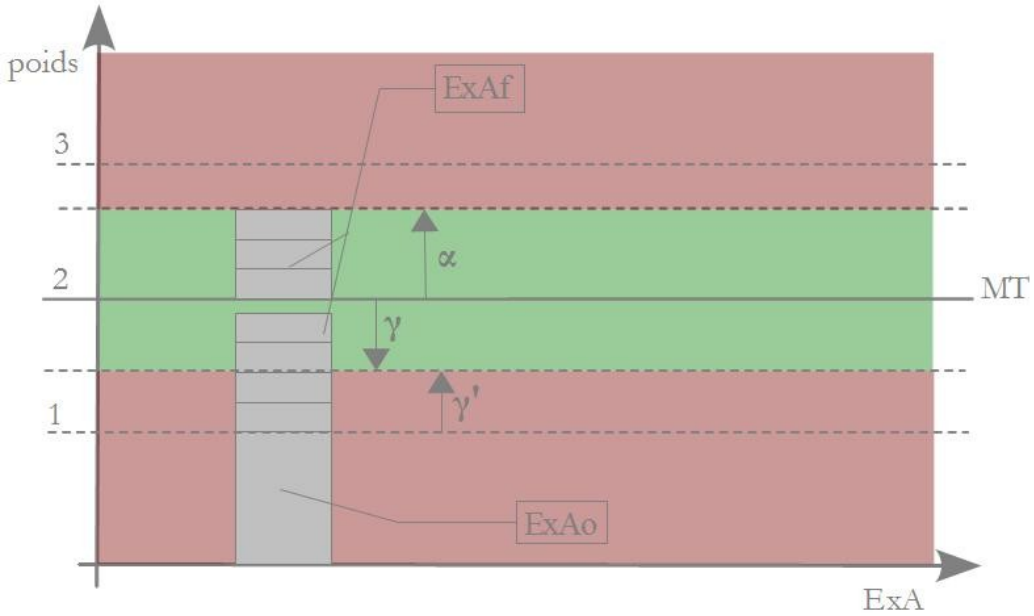

Illustration 45: Redéfinition de la zone d'acceptation autour de l'optimum

γ' est l'écart entre $MT - \gamma$ et la borne inférieure de la *ExAo* dans laquelle γ tombe. Dans le schéma, γ tombe dans la deuxième *ExAo* dont la borne inférieure est 1. On voit également qu'il faut deux *ExAo* pour atteindre le *match treshold*. α est fixé à trois *ExAf* et la réponse est acceptée à partir du moment où l'on donne une *ExAo* avec au moins deux *ExAf*. De cette manière il est plus clair de comparer α et γ' et il est plus facile de leur attribuer des valeurs en fonction de la configuration désirée pour la question. Ainsi, il existe trois relations possibles entre α et γ' :

- $\gamma' > \alpha$, pas de compensation possible (on ne peut atteindre le seuil d'acceptabilité qu'avec des *ExAo*) : typiquement parce que les *ExAf* génèrent rapidement du bruit (qui est très faiblement toléré si on a toutes les *ExAo*, mais dans le cas contraire, elles n'aident pas à compenser le manque)
- $\gamma' = \alpha$, tout juste tolérant au bruit : les *ExAf* font du bruit lorsque l'on a toutes les *ExAo*, mais elles ont quand même une valeur positive et permettent de compenser l'éventuel manque de *ExAo*
- $\gamma' < \alpha$, tolérance au bruit : les *ExAf* font très peu de bruit et ont un fort pouvoir de compensation (d'autant plus que γ' est faible et α élevé)

En terme de valeurs, si l'on considère le cas où la compensation est possible ($\gamma' \leqslant \alpha$), sachant que :

- $\sum_{j=1}^{k} poids(ExA_{f_j}) < 1$ et $\sum_{j=1}^{k} poids(ExA_{f_j}) \geqslant \alpha$,
- alors : $1 > \sum_{j=1}^{k} poids(ExA_{f_j}) \geqslant \alpha \geqslant \gamma'$

Il est donc possible de jouer sur la valeur de α et γ' pour exprimer différentes configurations de réponses en fonction des besoins liés au domaine d'application. De par ces relations, il est également possible d'épargner la détermination des valeurs à l'utilisateur et de les calculer automatiquement en fonction du type de relations qu'il souhaite.

Terminons par un exemple. Supposons qu'un examinateur prévoie pour une question :

- 5 *ExAo* (poids unitaire),
- 5 *ExAf* (supposons qu'elles sont toutes des poids identique égal à 0.15, leur somme est donc bien inférieure à 1),
- Seuil minimal (acceptabilité) : 3.3 (3 *ExAo* et 2 *ExAf*, γ'=0.3),
- Seuil maximal (bruit) : 5.45 (5 *ExAo* et 3 *ExAf*, α=0.45),
- ($\gamma' < \alpha$, les *ExAf* ont une valeur de compensation)

Si l'utilisateur retourne un jeu de réponses (EfA_i) qui correspond à 5 *ExAo* et 2 *ExAf*, son score vaut 5.3 qui est inférieur au seuil maximal et est donc valide. Si une autre configuration de ses réponses correspond à 3 *ExAo* et 4 *ExAf*, son score vaut 3.6 qui est supérieur au seuil minimal d'acceptabilité. Cependant, il a donné 4 *ExAf* alors α (qui représente une mesure du bruit accepté) est fixé à 3

ExAf maximum, elles font donc trop de bruit et la compensation ne vaut pas. La réponse est donc considérée comme erronée. Ainsi, dans le cas où le score entre dans les bornes d'acceptabilité, il convient de vérifier avec un second test si la compensation ne génère pas trop de bruit.

3.3.7. LES TEMPLATES

Le système repose sur un certain niveau d'abstraction. Nous avons vu qu'il est possible de réduire cette abstraction en faisant intervenir des représentations logiques plus proches des domaines (comme illustré sur le schéma 35), mais cela reste encore trop abstrait pour un utilisateur non informaticien. Il s'agit donc de masquer cette complexité et d'offrir aux experts en charge de la réalisation des contenus la possibilité de travailler directement avec les notions et les concepts propres à leur domaine. C'est le rôle des *templates*. Ceux ci se présentent sous la forme d'un jeu d'interface qui réalise ce masquage et qui se veulent le plus proche possible de l'affichage du contenu lors de la visualisation finale (ceci afin de rendre l'interface le plus intuitive possible[97]). Ainsi, si nous reprenons l'exemple de l'hématologie, lorsqu'un utilisateur désire ajouter un grand champ, le *template* associé à ce domaine proposera dans son interface un bouton dédié à l'ajout des méga-images. L'information associée à cette méga-image sera intégrée automatiquement à la représentation logique sous-jacente. Ainsi, à la description de la représentation logique d'un domaine (c'est-à-dire, la stratégie de *dtd* correspondante) est associé un *template* qui réalisera, en quelque sorte, la spécialisation (adaptation) finale au domaine de l'utilisateur. En résumé, lorsque l'on souhaite ajouter un domaine à la plate-forme, il faut :

- Voir s'il est possible d'utiliser la stratégie de *dtd* générale existante (voir illustration 35) pour les besoins informationnels du domaine
- Si ce n'est pas possible, créer une représentation logique du domaine à l'aide de fichiers *dtd* en étendant la base commune (c'est-à-dire les fichiers *doc.dtd* et *theme.dtd*, *scn.dtd* étant à reprendre tel quel car il n'est pas extensible[98])
- Créer un *template* qui facilitera la tâche de composition de l'utilisateur (par extension d'une classe abstraite « *Template* » existante au sein de la plate-forme)

Une fois ces opérations effectuées, il est alors possible de composer des contenus qui bénéficient de l'abstraction du modèle. Leur gestion, leur *parsing*,

97 De manière à ce que l'affichage de ce qu'il compose corresponde à l'affichage du résultat final lors de la visualisation. En d'autre terme, cela revient à du « *WYSIWYG* » (littéralement : « *What You See Is What You Get* »)

98 Voir l'illustration 28

leur adaptation, leur transmission et leur stockage sont pris en charge par le système, il n'y a rien à coder à ce niveau lors des intégrations de domaines.

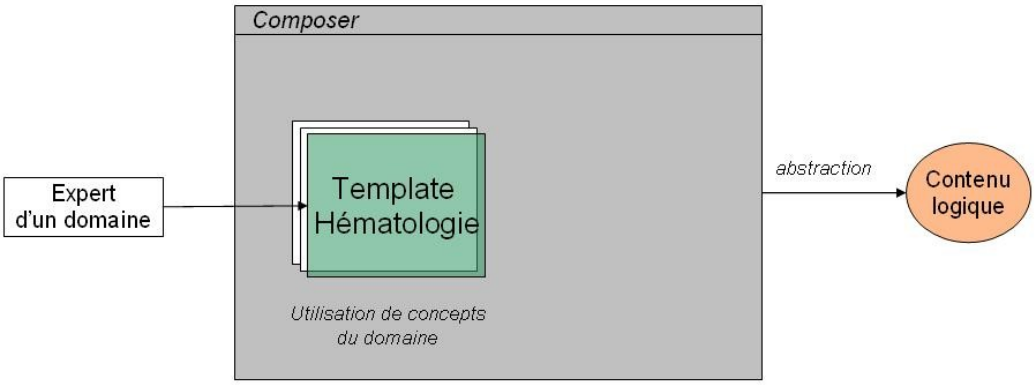

Illustration 46: Illustration de l'utilisation d'un template au sein de la plate-forme de composition

3.3.8. LA COOPÉRATION

Le rôle du réseau est double : d'une part, il offre la possibilité de récupération des contenus de contrôle en ligne et, d'autre part, il permet une composition coopérative de ces contenus. Grâce au serveur *svn* et son système de *versionning*, il est possible de rapatrier des contenus (en cours de réalisations ou terminés), d'ajouter de l'information et de *commiter* les changements. Cependant, ce système peut poser un problème de taille : les conflits de versions. Ces conflits surviennent lorsque deux utilisateurs travaillent dans le même document et au même endroit (dans ce document). En effet, considérons les utilisateurs A et B. Tous deux travaillent sur le même document et au même endroit. Supposons que A termine avant B et *commit* ses modifications. Sur le serveur, le document va incrémenter le document d'une *revision*. Lorsque l'utilisateur B va vouloir *commiter* à son tour ses modifications, le serveur va d'abord lui signaler qu'il ne peut réaliser l'opération car une version plus récente existe et il va l'inviter à faire un *update* pour se mettre à jour. Cela va mettre en concurrence les deux versions du document qui ont des modifications différentes au même endroit, ce qui va générer une erreur et entraîner la nécessité d'une résolution manuelle du conflit. Or, on ne peut pas demander aux utilisateurs de devoir plonger dans les méandres du document pour lever lui-même ces conflits. Il faut donc éviter que cette situation se produise. Pour se faire, l'idée est d'utiliser un système de verrouillage similaire à celui utilisé par *WebDAV*[99].

Ainsi, lorsqu'un utilisateur souhaite modifier un contenu, il doit signaler aux autres son intention en activant un « *lock* » sur ce document. On ne

99 *Web-based Distributed Authoring and Versionning* est une extension du protocole *HTTP* défini par le groupe *IETF* en vue de simplifier la gestion de fichiers avec des serveurs distants

peut modifier un document que lorsque l'on a le « *lock* » dessus. Grâce à la base de donnée (qui, rappelons-le, gère notamment l'information technique autour des contenus créés), il est possible de savoir quels sont les documents disponibles sur le serveur et quels sont ceux qui sont actuellement verrouillés ou pas. Ceci évite les conflits *svn* et permet de bénéficier des avantages de ce système : c'est-à-dire un transfert des modifications uniquement (ce qui est pratique car les contenus générés peuvent être de grande taille[100]) et un système de *versionning* automatique qui permet de récupérer une version antérieure en cas de problème. Deux primitives seront utilisées : *update* (ou *checkout* lorsque le contenu est récupéré pour la première fois en local) et *commit* (pour envoyer les modifications réalisées au serveur). Lorsqu'un document n'est pas verrouillé, il est possible de faire un *update* pour récupérer la dernière version provenant du serveur, suite à quoi il faut mettre le *lock*, réaliser les modifications nécessaires (pendant ce temps, les autres utilisateurs ne peuvent plus modifier le contenu), ensuite *commiter* et enfin enlever le *lock*. Remarquons qu'un cas de conflit peut quand même se produire : supposons qu'un utilisateur A fasse un *update*, il récupère la version la plus récente du contenu depuis le serveur, il entame ses modifications et oublie de mettre le *lock* sur le document. Si un autre utilisateur en profite pour modifier le contenu et le *commiter*, lorsque l'utilisateur A va vouloir soumettre ses changements au serveur, ce dernier va détecter que la version soumise est moins récente. Dans ce cas, la version du serveur sera prioritaire (ce qui entrainera la perte des modifications de l'utilisateur A), d'où l'importance de ne pas oublier de mettre le *lock* sur un document lorsque l'on souhaite le modifier. A cet effet, il faudra probablement envisager une activation automatique du *lock* lorsque l'utilisateur commence les modifications d'un contenu ou lorsqu'il fait un *update* (mais il n'*update* pas forcément pour faire des modifications, il peut simplement vouloir le consulter, dans ce cas le verrouillage serait inutile et bloquerait sans raison).

100Dans le cas de l'hématologie, les méga-images peuvent dépasser la centaine de *megasbytes*.

QUATRIÈME PARTIE : DISCUSSION

4. QUATRIÈME PARTIE : DISCUSSION

Le chapitre 3 met en lumière les travaux qui ont été effectués afin de servir l'objectif de réalisation d'une plate-forme de composition multimédia dédiée à l'évaluation externe de la qualité en hématologie. Ces travaux abordent deux fronts de développements complémentaires : un front technique et un front formel.

Le premier front traite des améliorations à apporter à la station de microscopie virtuelle utilisée pour la réalisation des méga-images. Rappelons que ces modifications s'inscrivent dans le cadre des perspectives de développement mises en place en vue de compléter les capacités de traitement (en terme d'acquisition et de visualisation) de la station. En effet, il n'a pas été question de transformer le microscope virtuel en un outil dédié à l'évaluation externe de la qualité. Il doit conserver son indépendance et rester cantonné à un outil de capture et de visualisation de frottis numériques. Les travaux réalisés aboutissent sur un système de capture semi-automatique d'images permettant la réalisation de grands champs.

Le second front porte une réflexion sur les besoins à couvrir dans le cas d'une plate-forme capable de prendre en charge la composition de contenus qui ne serait pas limitée à un seul domaine. L'hématologie sert de base de départ, mais la volonté de l'Institut de Santé Publique de mettre en place des contrôles dans d'autres branches de la médecine, nous ont poussé à réfléchir sur la mise en place d'une abstraction sur laquelle la plate-forme reposerait. Ainsi, une réflexion a été menée sur la manière de représenter logiquement les contenus en fonction des domaines ainsi que sur leur gestion et sur leur présentation. Cette réflexions ont débouché sur un modèle semi-général constitué d'une base extensible pouvant accueillir les besoins informationnels des domaines. L'implémentation du projet a également démarré et les développements réalisés portent principalement sur l'architecture générale du système et la conversion de contenus en objets Java (*parsing*). La manipulation des éléments décrits dans les fichiers *xml* et leur modélisation sous forme d'objets informatiques est, en effet, la première étape à réaliser.

4.1. MICROSCOPIE VIRTUELLE

4.1.1. PRISE EN CHARGE DE LA CAMÉRA

L'application Java *R.I.S.C.,* qui a été présentée dans le chapitre 2, apporte une alternative au logiciel AnalySIS. Bien qu'elle prenne entièrement en charge le pilotage du microscope, elle n'offre pas encore un pilotage de la caméra

DP-71. Malgré les tentatives à base de *TWAIN*, il n'a pas été possible d'apporter une solution qui assure cette prise en charge. Dès lors, la capture des images se fait par l'intermédiaire d'une application extérieure : *DP-Controller*. *R.I.S.C.* et *DP-Controller* sont donc complémentaires et utilisés conjointement dans les processus de capture. La nécessite d'une intervention manuelle au sein du *DP-Controller* pour déclencher un *snapshot* d'une image empêche toute capture entièrement automatique. Pour pallier à ce problème, dans un premier temps, la solution qui a semblé la plus raisonnable se base sur l'utilisation d'un robot informatique (*autoIt*) pour exercer une pression virtuelle sur le bouton. Cependant, même si cette possibilité peut amener une solution concrète, dans l'absolu, cela relève du bricolage informatique. Heureusement, l'arrivée[101] de librairies *dll* et de fichier *C* permettant le contrôle de la caméra va permettre d'apporter un dénouement correct à ce problème. Grâce à ceux-ci, il sera possible de piloter directement la *DP-71* à partir d'une application Java (grâce à JNI ou en travaillant directement sur les *dll*). Comme R.I.S.C. se propose d'être une alternative à AnalySIS, il s'agira de développer un module supplémentaire qui assurera la prise en charge du contrôle de la caméra. Ainsi, cette approche renoue avec l'objectif de capture entièrement automatique car les différents traitements pourront se faire à partir du même programme.

4.1.2. AMÉLIORATION DE L'AUTO-FOCUS

L'*auto-focus software* effectue le calcul préalable des hauteurs théoriques des plans focaux sur base de trois images de référence. Cette méthode comporte des avantages et des inconvénients. Elle donne de bons résultats pour des échantillons de taille raisonnable, mais lorsque lorsque ceux-ci sont plus grands, il peut apparaître des zones de flou dans le mur d'images. Des améliorations de cette approche sont envisageables.

Une des amélioration serait de permettre à l'utilisateur de cliquer sur une image au sein des zones de flou qui apparaissent après l'exécution de l'*auto-focus*, de manière à pouvoir déclencher une correction manuelle de la netteté de l'image. Sur base de la correction manuelle effectuée, les calculs des hauteurs pourraient être effectués à nouveau dans cette zone et raffiner la précision de l'interpolation.

Puisqu'il sera possible de piloter directement la caméra, on peut également améliorer les résultats et la fiabilité de la méthode en appliquant un *auto-focus software* automatique (piloté par un algorithme utilisant les images transmises par la caméra), soit sur l'ensemble des images d'une zone de capture, soit sur un sous-ensemble de celles-ci (de manière à augmenter le nombre de *tiles* de référence).

101 Celles-ci nous ont été envoyées après la durée du stage, il n'a donc pas été possible de réaliser le moindre test ou la moindre implémentation

Les calculs qui sont effectués actuellement se basent sur une interpolation linéaires des valeurs à attribuer aux hauteurs des *tiles*. Afin de mieux épouser les aspérités de l'échantillon, une interpolation polynomiale pourrait être utilisée. Cependant, il faudrait tenir compte des contraintes de performances liées à ces calculs. La version la plus simple[102] impose que le polynôme passe par tous les points de références, ce qui amène à la résolution d'un système linéaire qui implique une inversion matricielle[103]. Si le mur d'image est de grande taille, cela peut nuire fortement à la réactivité de l'application.

4.1.3. *INTÉGRATION DU CLIENT DE COREGISTRATION*

L'application *R.I.S.C.* n'est pas encore liée au processus de coregistration des images. Pour permettre la numérisation automatique des échantillons, cette opération est indispensable. La prise en charge de l'assemblage des images (qui est une opération lourde) se situe au niveau du serveur de coregistration. Le client de coregistration (ou client de création), quant à lui, se charge de transmettre les images capturées au serveur. Dès lors, afin de conserver l'approche client-serveur (comme l'illustre la figure 7 du chapitre 2) qui définit l'architecture du système de coregistration, c'est ce client qu'il convient d'assimiler à l'application *R.I.S.C.* En fait, c'est l'application elle-même qui devient le client de création. En effet, c'est elle qui, à terme, prendra en charge les acquisitions des images (via le pilotage du microscope et le pilotage de la caméra, comme le faisait AnalySIS que l'on commandait à l'aide de modules *C*).

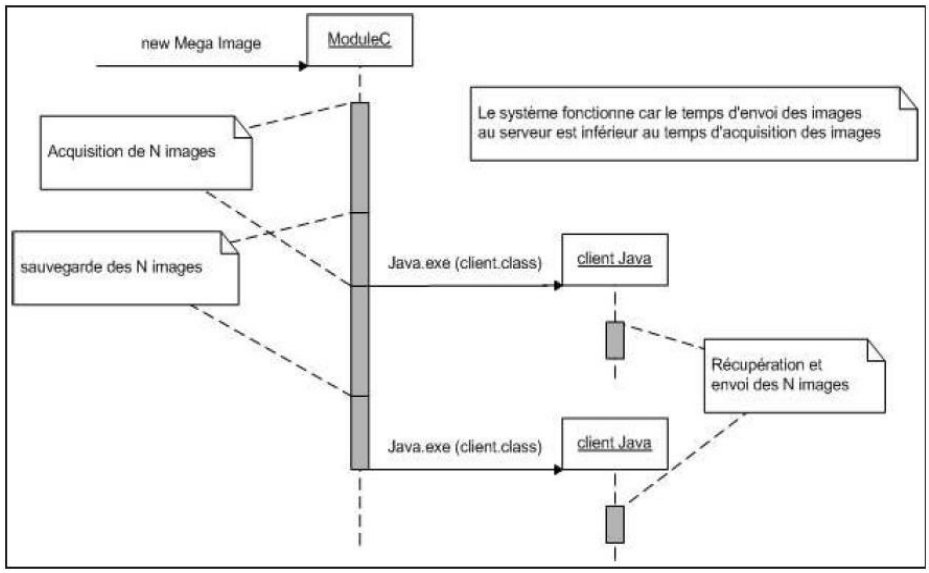

Illustration 47: Communication entre le module C et le client Java (tiré de [zuy03])

102Le polynôme d'interpolation de Lagrange

103Avec une méthode telle l'élimination de Gauss-Jordan, la complexité de l'opération est de l'ordre de $O(\frac{2n^3}{3})$. A ce propos, le calcul des différences divisées permet de remplacer la résolution du système linéaire et offre de meilleures performances.

Pour associer le processus de coregistration à l'application *R.I.S.C.*, deux approches sont possibles. Avant de les évoquer, il convient de se remémorer la manière dont fonctionne les interactions entre le module d'acquisition intégré à AnalySIS et le client Java de création.

Le module d'acquisition capture une série d'images qu'il sauve dans un dossier temporaire. Puis, il démarre un client de création Java qui se charge de transmettre ces images au serveur de coregistration, suite à quoi il meurt. L'opération se répète tant qu'il y a des images qui sont capturées. La transmission des images par l'intermédiaire d'un dossier temporaire est nécessaire car il n'est pas possible de faire communiquer directement le module *C* et le client Java. Ce constat nous amène à la première méthode d'intégration du client de coregistration à l'application *R.I.S.C.* : il suffit de considérer ces deux entités comme distinctes et de les faire communiquer via un dossier temporaire. Cette solution est la plus rapide, mais probablement la moins élégante. L'autre solution consiste à fusionner ces deux entités pour obtenir une transmission propre des images au serveur de coregistration. Les futurs développements devront naturellement s'orienter vers cette seconde solution.

4.1.4. AJOUT DE FONCTIONNALITÉS DE POST-TRAITEMENT

L'application *R.I.S.C.* est destinée à la gestion de l'acquisition des images. Pour se faire, les traitements qu'elle prend en charge se situent principalement autour du pilotage du microscope et de la caméra. Comme elle permet la génération d'images, elle se situe du côté de la composition. Pour compléter son fonctionnement, on pourrait ajouter un module de traitement d'image. Ce module pourrait reprendre les fonctionnalités d'homogénéisation et de création d'une galerie de globules blancs du client de visualisation du microscope virtuel. Ce qui recadrerait les rôles de ces deux entités : l'une sert à la génération de d'images, l'autre à leur visualisation. De plus, en ayant directement accès aux images reçues par la caméra, il est même possible d'envisager la création d'une galerie de globules blancs directement à partir de l'échantillon que l'on parcourrait à l'objectif maximal (100x). De manière générale, ces fonctionnalités de post-traitement pourraient servir à améliorer la qualité des images capturées (en agissant sur les couleurs ou la luminosité[104]) pour des méga-images de rendu optimal.

104On peut imaginer un menu qui permette d'appliquer des effets sur l'image

4.2. EVALUATION EXTERNE DE LA QUALITÉ

4.2.1. POURSUITE DE L'IMPLÉMENTATION DU MODÈLE

Comme nous l'avons vu au chapitre 2, l'objectif concernant l'évaluation externe de la qualité ne faisait pas partie des objectifs initiaux fixés pour ce mémoire. Lorsqu'il est apparu, il a été nécessaire de définir une direction globale qui puisse permettre de déterminer une perspective de développement qui serve l'ensemble des objectifs choisis. Ainsi, cette direction coïncide avec le démarrage d'un nouveau projet de création d'une plate-forme de composition multimédia pour l'évaluation externe de la qualité. Les besoins pour la réalisation d'une telle plate-forme faisaient notamment intervenir une série de considérations techniques, ce qui a permis d'englober dans ce projet les développements inscrits dans la perspective d'amélioration du microscope virtuel du laboratoire d'hématologie.

Puisqu'il s'agit d'un nouveau projet, dans le souci d'aboutir sur un produit qui corresponde le mieux aux attentes des différents intervenants, il convient de procéder à une série d'analyses (des exigences, de l'existant) et de réflexions (élaboration d'un modèle implémentable) avant d'entamer la rédaction du code. C'est sur ces points que portent principalement la seconde partie des résultats de ce mémoire. En effet, étant donné l'ensemble des objectifs à traiter et la taille du nouveau projet, il n'était pas concevable d'imaginer pouvoir terminer la réalisation de la plate-forme dans le cadre d'un seul stage. Ce fut également le cas avec le projet du microscope virtuel qui est d'ailleurs toujours en phase d'amélioration. Dans le souci de servir au mieux le projet et de faciliter le travail des successeurs, il était raisonnable de se focaliser prioritairement sur ces analyses préalables et d'entamer ensuite le début du codage. Ainsi, il revient aux suivants de continuer l'implémentation de la plate-forme de composition sur base des spécifications établies et du code déjà rédigé.

4.2.2. CONVERSION DES DTD'S EN XSD

Le modèle de représentation logique des contenus et l'implémentation déjà réalisée s'appuient sur l'utilisation de fichier *dtd*. La section 3.3.3 du chapitre mentionne le manque d'expressivité du langage en terme d'extension. Et pour cause : ce mécanisme est inexistant dans ce langage. Une alternative qui permet de pallier à ce manque est d'utiliser un autre langage de description qui sont les schémas *xml*. Ceux-ci comportent une série d'avantages par rapport aux fichiers *dtd* : ils sont eux-même des documents *xml* (ce qui permet leur manipulation et facilite leur évolution), ils permettent de déterminer un type (simple ou complexe) pour les éléments qu'ils contiennent et ils incluent des mécanismes d'extension. Ces raisons incitent à les préférer au *dtd* car ces fichiers permettent de représenter plus

adéquatement l'idée de l'extension de la base abstraite du modèle et permettent de vérifier la validité de ces extensions (rappelons que seule l'extension de base est autorisée, les restrictions sont interdites)[105]. Le remplacement des fichiers *dtd* par les fichiers *xsd* dans le code n'est pas une opération lourde. En effet, les objets qui sont dérivés des éléments décrits par ces fichiers restent les mêmes. Il s'agit en fait d'une traduction mais la sémantique des objets ne change pas. Le travail de conversion se situe principalement au niveau des *parsers* qui doivent dorénavant prendre en paramètre des fichiers de type *xsd* au lieu de fichiers de type *dtd* (la création et la manipulation des objets Java associés aux éléments décrits dans ces fichiers ne changent pas).

105De manière à faciliter leur évolution et a pouvoir assurer une certaine rétro-compatibilité

CINQUIÈME PARTIE : CONCLUSION

5. CONCLUSION

Les avancées en matière de technologie de l'information et de la communication ont permis l'essor de la télémédecine et, plus particulièrement, de la télémicroscopie. Il est clair que cette discipline va continuer à se développer à l'avenir tant les limitations techniques au niveau des technologies de stockage, des capacités de traitement et des performances réseaux sont sans cesse repoussées. Dans le même temps, la qualité des images générées et la taille de la surface des échantillons numérisés vont inévitablement améliorer la qualité des contrôles externes et ouvrir de nouvelles perspectives.

Dans la continuité des travaux réalisés au laboratoire, ce mémoire a consisté à poursuivre la perspective d'amélioration de la station de microscopie virtuelle et à amener un cas concret d'utilisation de cette station à travers l'évaluation externe de la qualité. Ainsi, le travail qui a été effectué et qui est rapporté dans ce mémoire se scinde en deux fronts de développements : l'un porte sur des considérations techniques liées aux améliorations à apporter à la station, l'autre concerne la modélisation d'une plate-forme générale de composition de contenus multimédias exploitables dans le cadre des évaluations externes de la qualité.

Ainsi, dans un premier temps, nous avons naturellement pris connaissance du contexte existant en matière de télémédecine, de télémicroscopie et de contrôle externe de qualité.

Suite à cela, nous avons défini une configuration d'objectifs qui répondent le mieux à l'urgence des besoins à couvrir. Puis, nous avons pris connaissance des spécificités du microscope virtuel et de l'existant en matière de composition média ([EMIM]) en milieu hospitalier.

Les résultats obtenus couvrent partiellement les objectifs prévus. En effet, l'apparition de nouveaux problèmes au niveau du pilotage du microscope bloquaient les autres développements. Il a donc fallu s'adapter et redéfinir la stratégie de prise en charge des objectifs. Cette redéfinition a mené à la réalisation d'une nouvelle application de pilotage qui puisse prendre en charge les opérations supportées par AnalySIS (principalement le pilotage du microscope et de la caméra). Pour se faire, il a été nécessaire d'analyser de manière détaillée le fonctionnement du microscope et les interactions entre les modules qui le composent. Il a également fallu apporter une solution à la désynchronisation qui peut survenir dans la gestion des *stacks* et déterminer la manière de réaliser une calibration propre du plateau motorisé. Cette nouvelle application a permis de lever les problèmes liés au pilotage du microscope et propose également une solution *software* pour la gestion de l'*auto-focus*. Cependant, en ce qui concerne le pilotage de

la caméra, les tentatives visant à utiliser le standard *TWAIN* nous ont appris que cette voie était sans issue. Néanmoins, la réception de librairies permettant le contrôle de la caméra permettront d'apporter la solution la plus élégante à ce problème. Ainsi, ces développements mènent à un système de capture semi-automatique qu'il conviendra de compléter par la prise en charge de la caméra et la fusion avec le client de coregistration.

L'objectif général de modélisation de la plate-forme de composition aboutit à un modèle qui intègre une représentation logique semi-générale de l'information associée aux contenus. Cette représentation est constituée d'une base commune abstraite que l'on étend en fonction des besoins informationnels liés aux domaines. Cette extension interdit l'altération des types existants, de manière à permettre l'évolution des représentations logiques étendues ou encore d'assurer leur rétro-compatibilité. Le modèle intègre des mécanismes de transformation des contenus de contrôle visant à adapter ceux-ci au contexte de visualisation. Ces transformations portent sur l'information qu'ils contiennent et sur les formes qu'ils peuvent prendre. Un système de gestion des résultats permet leur évaluation semi-automatique. Ce système supporte des configurations complexes de questions et est fortement paramétrable. Le modèle fait également intervenir des *templates* qui permettent de masquer la complexité générale liée à l'abstraction. Dans le souci de faciliter le travail de composition, ces *templates* ont également pour vocation de faire correspondre artificiellement le contexte de composition avec celui de visualisation (*WYSIWYG*). Enfin, une dimension « réseau » permet d'envisager des scénarios de contrôles *online* et d'intégrer de la coopération entre experts dans le travail de composition. L'implémentation du modèle a pu démarrer et porte actuellement sur l'architecture générale du système (les classes de haut niveau qui définissent la gestion du contexte, des documents, stratégies de dtd's, des *parsers* et des *templates*) et la gestion de l'information contenue dans les représentations logiques. Les futurs développements devront naturellement poursuivre cette implémentation.

Je clôturerai ce mémoire en signalant que ce stage a été une expérience enrichissante. Il m'a permis de découvrir le monde de la médecine (et plus particulièrement celui de l'hématologie) que je ne connaissais pas et de m'intégrer dans une routine de travail en milieu professionnel.

ANNEXES

6. ANNEXES

6.1. CONNEXION DE WINPOS : LES COMMANDES ENVOYÉES

Lorsque l'application WinPos se connecte au port sur lequel est connecté le microscope (plus précisément, le *Multi-Control*), elle envoie une série de commande dans le but d'établir une connexion fiable. Cela a notamment pour effet de réparer une désynchronisation éventuelle des *stacks*, un problème évoqué lors des analyses préalables au développement de l'application de pilotage (point 3.2.2). Ces commandes ont servi de première approche dans l'élaboration d'un jeu de commandes visant rétablir la synchronisation. Cependant, l'utilisation de *Macros* (telles que *CleanBuffers*, derrière laquelle se cache peut être la solution) masque une partie des commandes.

Commande	**Information**
version	Retourne le numéro de version du *firmware* du contrôleur
identify	Retourne le numéro d'identification du contrôleur
0 j st 1 j st 0 j st	Non renseigné. « st » (status) retourne des informations de statut sur certaines éléments du contrôleur en mode « host »
2 setdim	Défini 2 axes (ordonnées et abscisses)
1 getMotorType	Retourne le type du moteur sélectionné
2 getMotorType	Retourne le type du moteur sélectionné
CleanBuffers	*Macro*
0 j st 1 j st 0 j st	Non renseigné. « st » (status) retourne des informations de statut sur certaines éléments du contrôleur en mode « host »
getDim	Retourne le nombre de dimensions définies par « setDim » (en l'occurence, « 2 » dans ce cas ci)
1 getAxis	Retourne le statut de l'axe sélectionné
2 getAxis	Retourne le statut de l'axe sélectionné

0 getUnit	Retourne l'unité de l'axe X
1 getUnit	Retourne l'unité de l'axe Y
2 getUnit	Retourne l'unité de l'axe Z
getAccel	Retourne la valeur d'accélération maximum
getVel	Retourne la vitesse de déplacement courante du plateau
getLimit	Retourne les positions des butées mécaniques du plateau (distance maximale indexable)
CheckSavedValues	Macro
getDim	Retourne le nombre de dimensions
getLimit	Retourne les positions des butées mécaniques du plateau
-1 getPitch	Retourne la valeur définie par la commande « setPitch »
getJoystickType	Retourne le type de *joystick* utilisé avec le microscope
getJoyspeed	Retourne la vitesse de déplacement du *joystick*
-1 getCloop	Indique si la condition « closed loop » est activée (« -1 » signifie « pour tous les axes »)
-1 getClFactor	Non renseigné
getCalVel	Retourne la vitesse de déplacement du plateau lors de l'appel « cal » (calibration)
getRmVel	Retourne la vitesse de déplacement du plateau lors de l'appel « rm » (range mesures)

6.2. EQA : ILLUSTRATION DE L'IMPLÉMENTATION

6.2.1. LE PARSING D'UN DOCUMENT DEPUIS UN TEMPLATE

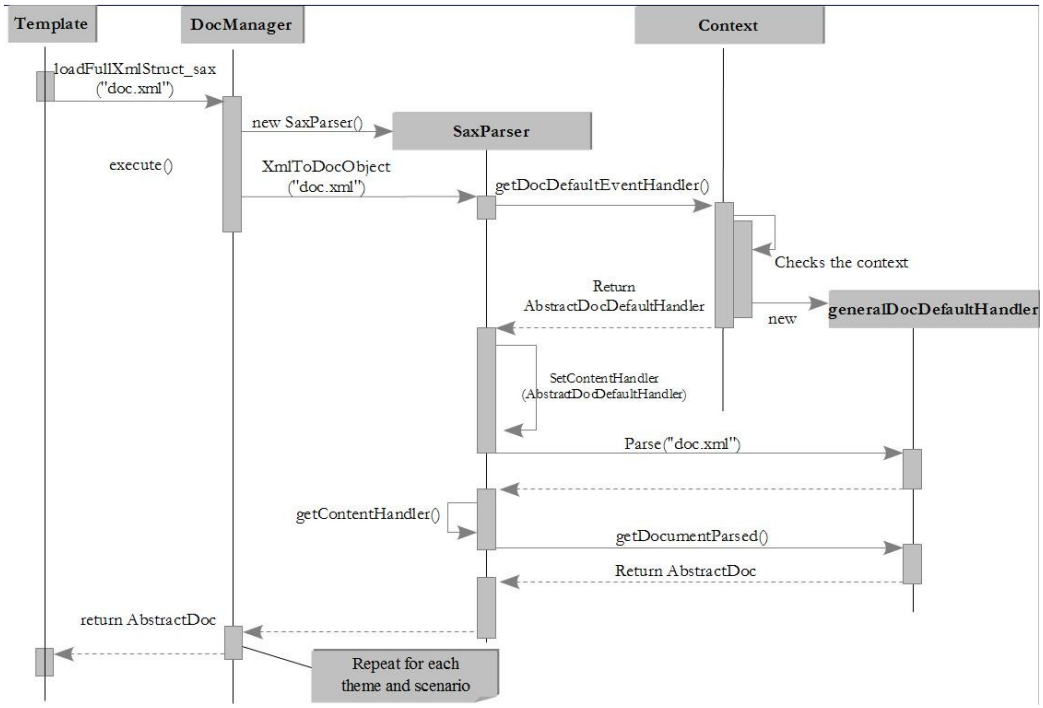

Illustration 48: Diagramme de séquence du parsing d'un document xml depuis un template

Le schéma ci-dessus montre le parsing d'un document *xml* depuis un *template*. L'objectif en ce qui concerne la programmation dans un *template* est que celle-ci doit rester simple pour permettre un développement rapide. Comme on le voit dans le schéma, le *template* ne fait que commander un *parsing* et c'est les différents *managers* intervenant en aval qui se chargent de la complexité liée au contexte dynamique.

Ainsi, pour *parser* un document depuis un *template*, on fait appel à la fonction « loadFullXmlStruct_sax » du *DocManager* (qui gère les documents, qui sont des objets *AbstractDoc*, eux-même extensibles en fonction des contextes liés aux différents domaines qui correspondent aux stratégies de *dtd*). Celui-ci démarre crée un *parser* Sax. Le parser va ensuite prendre connaissance du contexte dans lequel il se trouve (le contexte actuel du programme, c'est-à-dire, apporter une réponse à la question : l'utilisateur est en train de travailler dans quel domaine ?). Pour ce faire il interroge le *manager* en charge du contexte. Celui-ci consulte quelle stratégie de *dtd* a été chargée. Dans notre exemple, la stratégie de *dtd*, et donc le domaine, chargé correspond à la stratégie « générale ». Pour rappel, cette stratégie est la stratégie qui prend en charge le domaine de l'hématologie. Cependant, elle

s'appelle « générale » car elle est destinée à prendre en charge d'autres domaines (voir la fin de la section 3.3.3 à ce sujet). Ainsi, le contexte crée un objet « *generalDocDefaultHandler* » (pour le domaine courant) qui est une extension de « *AbstractDocDefaultHandler* » (pour la base commune) qui est lui-même une extension de « *DefaultHandler* » qui représente un *handler* Sax classique. Grâce à ces spécialisation, le « *generalDocDefaultHandler* » est capable de manipuler l'information associée au domaine de l'hématologie qui est décrite dans le fichier *xml*. Maintenant que le contexte a créé le *handler* adéquat, il rend la main au *parser* Sax qui va utiliser ce *handler*. Il lui suffit donc ensuite de lancer un « *parse* » classique sur le document *xml* pour que le *handler* fasse son travail. Une fois le que le *parser* récupère la main (après le traitement du *handler*), il récupère le *handler* courant (c'est-à-dire celui qui vient de terminer son travail avec le fichier *xml*) et lui demande le résultat sous la forme d'un objet « AbstractDoc ». Remarquons que cet objet, qui fait partie de la base commune, est étendu dans le *handler* par un objet « GeneralDoc » qui renferme les structures de données qui peuvent accueillir les concepts liés au domaine décrit par la stratégie de *dtd*. Enfin, il transmet ce document au *DocManager*. Ce traitement sera similairement appliquer pour les thèmes et les scénarios (sauf que pour ces derniers, l'extension des objets ne dépasse pas la base abstraite commune, comme l'indique l'illustration 37).

6.2.2. ILLUSTRATION DES DESIGN PATTERNS SINGLETON ET STRATEGIE

```
public class Context {
	private static Context context=null;
	private int dtdStrategy;
	private AbstractDocDefaultHandler docDefaultEventHandler = null;
	...


	//locked constructor
	private Context(){}

	//returning an instance of the context
	public static Context getInstance()
	{
		if(context==null)
		{
			context = new Context();
		}
		return context;
	}

	...

	/*----------------- SAX PARSER HANDLERS*/
	private void setDocDefaultEventHandler()
	{
		switch(this.dtdStrategy)
		{
```

```
                    case
                    AbstractDtd.DTD_STRATEGY_GENERIC :
                              this.docDefaultEventHandler = new
                              GeneralDocDefaultEventHandler();
                                        break;

                    case AbstractDtd.DTD_STRATEGY_RADIOLOGY :
                              this.docDefaultEventHandler = new
                              RadiologyDocDefaultEventHandler();
                                        break;
                    case ...
                    }
          }

          ...

          public AbstractDocDefaultHandler getDocDefaultEventHandler()
          {
                    if(this.docDefaultEventHandler==null)
                    {
                              this.setDocDefaultEventHandler();
                    }
                    return this.docDefaultEventHandler;
          }

...

}
```

Le code ci-dessus est tiré de la classe *Context*. Celle-ci est un singleton travaillant selon le *design pattern* « stratégie ».

Pour le singleton, on voit que cela revient à bloquer le constructeur et appeler statiquement l'instance de la classe. De cette manière, il n'est pas possible d'avoir plus d'un contexte. Pour la stratégie, on voit que la création et la récupération d'un *handler* est conditionnée à la stratégie de *dtd* active.

6.3. Structure d'un document (Emim2)

6.3.1. Le document (dmg.dtd)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- DTD for DMG/EMIM 2.0-->

<!-- Generic markup's-->
<!ELEMENT identification (#PCDATA)>
<!ELEMENT firstName (#PCDATA)>
<!ELEMENT lastName (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT mail (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT volume (#PCDATA)>
```

```
<!ELEMENT origin (#PCDATA)>
<!ELEMENT creationDate (#PCDATA)>
<!ELEMENT transmissionDate (#PCDATA)>

<!-- DMG -->
<!ELEMENT DMG (infoDMG, infoAccess, infoPatient, listDMP?)>
<!ATTLIST DMG
    name ID #REQUIRED
    version CDATA "EMIM_DMG_2.0"
    status (approved | temporary) "temporary"
>
<!-- infoDMG -->
<!ELEMENT infoDMG (type?, title?, volume?, origin?, creationDate, transmissionDate?,
validationDate?, description?)>
<!ELEMENT validationDate (#PCDATA)>
<!ELEMENT description (#PCDATA)>

<!-- infoAccess -->
<!ELEMENT infoAccess (creator?, author?, reader?)>
<!ELEMENT creator (identification, firstName, lastName, address?, mail?)>
<!ELEMENT author (listGroup?, listUser?)>
<!ELEMENT reader (listGroup?, listUser?)>
<!ELEMENT listGroup (group+)>
<!ELEMENT listUser (user+)>
<!ELEMENT group (#PCDATA)>
<!ELEMENT user (#PCDATA)>

<!-- infoPatient -->
<!ELEMENT infoPatient (identification, firstName?, lastName?, mutuel?, occupation?, birthDate?,
sex?, size?, weight?, address?, tel1?, tel2?, comment?, physician?, admission?)>
<!ELEMENT occupation (#PCDATA)>
<!ELEMENT birthDate (#PCDATA)>
<!ELEMENT sex (#PCDATA)>
<!ELEMENT size (#PCDATA)>
<!ELEMENT weight (#PCDATA)>
<!ELEMENT tel1 (#PCDATA)>
<!ELEMENT tel2 (#PCDATA)>
<!ELEMENT mutuel (#PCDATA)>
<!ELEMENT comment (#PCDATA)>
<!ELEMENT physician (identification?, firstName?, lastName?, address?, mail?)>
<!ELEMENT admission (institutionName, institutionAddress?, departement?, admittingDate,
admittingDescription?)>
<!ELEMENT institutionName (#PCDATA)>
<!ELEMENT institutionAddress (#PCDATA)>
<!ELEMENT departement (#PCDATA)>
<!ELEMENT admittingDate (#PCDATA)>
<!ELEMENT admittingDescription (#PCDATA)>

<!-- listDMP -->
<!ELEMENT listDMP (DMP+)>
<!ELEMENT DMP (infoDMP)>
<!ATTLIST DMP
    name ID #REQUIRED
    version CDATA "EMIM_DMP_2.0"
    status (approved | temporary) "temporary"
>
<!ELEMENT infoDMP (number?,type?, title?, volume?, location, creationDate,
transmissionDate?)>
<!ELEMENT number (#PCDATA)>
<!ELEMENT location EMPTY>
```

```
<!ATTLIST location
    href CDATA #REQUIRED
>
```

6.3.2. LA PAGE (DMP.DTD)

```
 <?xml version="1.0" encoding="UTF-8"?>
<!-- DTD for DMP/EMIM 2.0-->

<!-- Generic markup's-->
<!ELEMENT identification (#PCDATA)>
<!ELEMENT firstName (#PCDATA)>
<!ELEMENT lastName (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT mail (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT volume (#PCDATA)>
<!ELEMENT comment (#PCDATA)>
<!ELEMENT origin (#PCDATA)>
<!ELEMENT creationDate (#PCDATA)>
<!ELEMENT transmissionDate (#PCDATA)>
<!ELEMENT location EMPTY>
<!ATTLIST location
    href CDATA #REQUIRED
>

<!-- DMP -->
<!ELEMENT DMP (infoDMP, infoAccess, infoStudy, listSCN?)>
<!ATTLIST DMP
    name ID #REQUIRED
    version CDATA "EMIM_DMP_2.0"
    status (approved | temporary) "temporary"
>

<!-- infoDMP -->
<!ELEMENT infoDMP (number?,type?, title?, volume?, origin?, creationDate, transmissionDate?,
location)>
<!ELEMENT number (#PCDATA)>

<!--infoAccess -->
<!ELEMENT infoAccess (creator?,author?, reader?)>
<!ELEMENT creator (identification, firstName,lastName, address?,mail?)>
<!ELEMENT author (listGroup?, listUser?)>
<!ELEMENT reader (listGroup?, listUser?)>
<!ELEMENT listGroup (group+)>
<!ELEMENT listUser (user+)>
<!ELEMENT group (#PCDATA)>
<!ELEMENT user (#PCDATA)>

<!--infoStudy -->
<!ELEMENT infoStudy (beginDate?, endDate?, institutionName?, institutionAddress?,modality,
reason?, results?, comment?, furnisher?, listInterpretation?, listEM)>
<!ELEMENT beginDate (#PCDATA)>
<!ELEMENT endDate (#PCDATA)>
<!ELEMENT institutionName (#PCDATA)>
<!ELEMENT institutionAddress (#PCDATA)>
<!ELEMENT modality (#PCDATA)>
```

```
<!ELEMENT reason (#PCDATA)>
<!ELEMENT results (#PCDATA)>
<!ELEMENT furnisher (identification?, firstName?, lastName?, address?, mail?)>
<!ELEMENT listInterpretation (interpretation+)>
<!ELEMENT interpretation (date, time, identification, firstName?, lastName?, address?,mail?,
details)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT time (#PCDATA)>
<!ELEMENT details (#PCDATA)>
<!ELEMENT listEM (numberText, numberImage, numberAudio, numberVideo, EM*)>
<!ELEMENT numberText (#PCDATA)>
<!ELEMENT numberImage (#PCDATA)>
<!ELEMENT numberAudio (#PCDATA)>
<!ELEMENT numberVideo (#PCDATA)>

<!-- listSCN -->
<!ELEMENT listSCN (SCN+)>
<!ELEMENT SCN (infoSCN)>
<!ATTLIST SCN
    name ID #REQUIRED
>
<!--infoSCN -->
<!ELEMENT infoSCN (location)>

<!--EM -->
<!ELEMENT EM (type, volume?, location, serieNumber?, spatialPosition?, temporalPosition?,
comment?)>
<!ATTLIST EM
    name ID #REQUIRED
>
<!ELEMENT serieNumber (#PCDATA)>
<!ELEMENT spatialPosition (#PCDATA)>
<!ELEMENT temporalPosition (#PCDATA)>
```

6.3.3. LE SCÉNARIO (SCN.DTD)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- DTD for SCN/EMIM 2.0-->
<!-- Generic markup's-->
<!ELEMENT location EMPTY>
<!ATTLIST location
    href CDATA #REQUIRED
>
<!ELEMENT type (#PCDATA)>
<!-- SCN -->
<!ELEMENT SCN (infoSCN, infoConstraint?, listConversion?, listTransformation?, listEM)>
<!ATTLIST SCN
    name ID #REQUIRED
>
<!--infoSCN -->
<!ELEMENT infoSCN (location)>
<!ELEMENT infoConstraint (numberTextAccepted?, numberImageAccepted?,
numberAudioAccepted?, numberVideoAccepted?, name?, number?, spatial?, temporal?)>
<!ELEMENT numberTextAccepted EMPTY>
<!ATTLIST numberTextAccepted
    min CDATA #IMPLIED
    max CDATA #IMPLIED
    exact CDATA #IMPLIED
>
```

```
<!ELEMENT numberImageAccepted EMPTY>
<!ATTLIST numberImageAccepted
    min CDATA #IMPLIED
    max CDATA #IMPLIED
    exact CDATA #IMPLIED
>
<!ELEMENT numberAudioAccepted EMPTY>
<!ATTLIST numberAudioAccepted
    min CDATA #IMPLIED
    max CDATA #IMPLIED
    exact CDATA #IMPLIED
>
<!ELEMENT numberVideoAccepted EMPTY>
<!ATTLIST numberVideoAccepted
    min CDATA #IMPLIED
    max CDATA #IMPLIED
    exact CDATA #IMPLIED
>
<!ELEMENT name (value+)>
<!ELEMENT number (value+)>
<!ELEMENT spatial (value+)>
<!ELEMENT temporal (value+)>
<!ELEMENT value (#PCDATA)>
<!ELEMENT listConversion (conversion+)>
<!ELEMENT conversion (typeIn, typeOut, compression)>
<!ELEMENT typeIn (#PCDATA)>
<!ELEMENT typeOut (#PCDATA)>
<!ELEMENT compression EMPTY>
<!ATTLIST compression
    type CDATA #REQUIRED
    rate CDATA #IMPLIED
>
<!ELEMENT listTransformation (transformation+)>
<!ELEMENT transformation (input+, sheet+, output+)>
<!ELEMENT input (type, basic, location)>
<!ELEMENT sheet (type, basic, location)>
<!ELEMENT output (type, basic, location)>
<!ELEMENT basic (#PCDATA)>
<!ELEMENT listEM (numberText, numberImage, numberAudio, numberVideo, EM*)>
<!ELEMENT numberText (#PCDATA)>
<!ELEMENT numberImage (#PCDATA)>
<!ELEMENT numberAudio (#PCDATA)>
<!ELEMENT numberVideo (#PCDATA)>
<!--EM -->
<!ELEMENT EM (type, volume?, location, serieNumber?, spatialPosition?, temporalPosition?,
comment?)>
<!ATTLIST EM
    name ID #REQUIRED
>
<!ELEMENT volume (#PCDATA)>
<!ELEMENT serieNumber (#PCDATA)>
<!ELEMENT spatialPosition (#PCDATA)>
<!ELEMENT temporalPosition (#PCDATA)>
<!ELEMENT comment (#PCDATA)>
```

6.4. STRUCTURE D'UN CONTENU EN HÉMATOLOGIE (EQA)

6.4.1. LE DOCUMENT (DOC.DTD)

```
<!ELEMENT examination (examinationInfos, caseInfos?, theme*, scenario*)>

<!-- _____ Examination infos -->
<!ELEMENT examinationInfos (exam_logo?, exam_title, exam_description?, exam_releaseDate?,
exam_deadline?, author+, contributor*, publisher+, exam_comment)>
<!ATTLIST examinationInfos
                        id CDATA #REQUIRED
                        dtdStrategy NMTOKENS #REQUIRED
                        dtdStrategyVersion NMTOKEN #REQUIRED
                        type NMTOKENS #IMPLIED
                        supportedLanguages NMTOKENs #REQUIRED
>
<!ELEMENT exam_logo (#PCDATA)>
<!ATTLIST exam_logo
                        location #CDATA #REQUIRED>
<!ELEMENT exam_title (#PCDATA)>
<!ELEMENT exam_description (#PCDATA)>
<!ELEMENT exam_releaseDate(#PCDATA|EMPTY)>
<!ATTLIST exam_releaseDate
                        day NMTOKEN #IMPLIED
                        month NMTOKEN #IMPLIED
                        year NMTOKEN #IMPLIED>
<!ELEMENT exam_deadline(#PCDATA)
<!ATTLIST exam_deadline
                        day NMTOKEN #IMPLIED
                        month NMTOKEN #IMPLIED
                        year NMTOKEN #IMPLIED>
<!ELEMENT exam_comment (#PCDATA)>

<!-- an author -->
<!ELEMENT author (person,author_comment)>
<!ELEMENT author_comment (#PCDATA)> <!-- com sur participation -->

<!-- a contributor (help, consultance, thanks, ...) -->
<!ELEMENT contributor (person, contribution, contrib_comment)>
<!ELEMENT contributor_comment (#PCDATA)> <!-- com sur contribution -->

<!-- a publisher -->
<!ELEMENT publisher ( person, publisher_comment)>
<!ELEMENT publisher_comment (#PCDATA)> <!-- com sur Edition/publication -->

<!-- a person -->
<!ELEMENT person (lastname, firstname, sex?, birthdate?, institution*, phone*, email*,
comment)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT sex (#PCDATA)>
<!ELEMENT birthdate (#PCDATA)>
<!ATTLIST birthdate
                        day NMTOKEN #IMPLIED
                        month NMTOKEN #IMPLIED
                        year NMTOKEN #IMPLIED
```

```
>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENt comment (#PCDATA)> <!-- com sur la personne -->


<!-- an institution -->
<!ELEMENT institution (institutionAdress, institutionPhone, institutionEmail,
institutionComment)>
<!ATTLIST institution
                        name #CDATA #REQUIRED
>
<!ELEMENT institutionAdress (#PCDATA)>
<!ELEMENT institutionEmail (#PCDATA)>
<!ELEMENT institutionComment (#PCDATA)> <!-- com sur l'institution -->

<!-- _____________________________________________________ Case infos -->
<!ELEMENT caseInfos (virtualPatient, caseOrigins?)>

<!-- virtualPatient -->
<!ELEMENT virtualPatient(lastname?, firstname?, sex?, age?, size?, weight?, nationality?,
description, vp_comment)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT sex (#PCDATA)>
<!ELEMENT age (#PCDATA)>
<!ELEMENT size (#PCDATA)>
<!ELEMENT weight (#PCDATA)>
<!ELEMENT nationality (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT vp_comment (#PCDATA)>

<!-- caseOrigins -->
<!ELEMENT co_caseOrigins (co_location?, co_diagnosticDate?, co_description)>
<!ELEMENT co_location (#PCDATA)>
<!ELEMENT co_diagnosticDate (#PCDATA)>
<!ATTLIST co_diagnosticDate
                        day NMTOKEN #IMPLIED
                        month NMTOKEN #IMPLIED
                        year NMTOKEN #IMPLIED
>
<!ELEMENT co_description (#PCDATA)>

<!-- _____________________________________________________ ThemeSummary -->
<!-- theme -->
<!ELEMENT theme (theme_title)>
<!ATTLIST theme
                        id CDATA #REQUIRED
                        location #CDATA #REQUIRED
                        order NMTOKEN #REQUIRED>

<!ELEMENT theme_title (#PCDATA)>



<!-- _____________________________________________________ ScnSummary  -->
<!-- scenario-->
<!ELEMENT scenario (EMPTY)>
<!ATTLIST scenario
                        id CDATA #REQUIRED
```

```
                    location #CDATA #REQUIRED>
```

6.4.2. LE THEME (THEME.DTD)

```
<!ELEMENT theme(themeInfos, media*, illustration*, question*)>

<!-- themeInfos -->
<!ELEMENT themeInfos (theme_title, theme_description, theme_comment)>
<!ATTLIST themeInfos
                    id #CDATA #REQUIRED
                    doctype NMTOKEN #REQUIRED
                    order NMTOKEN #IMPLIED
                    access (free | restricted)
                    accessed (true | false)>

<!ELEMENT theme_title (#PCDATA)>
<!ELEMENT theme_description (#PCDATA)>
<!ELEMENT theme_comment (#PCDATA)>

<!-- illustration -->
<!ELEMENT illustration (illustr_title?, mediaReference*, illustr_comment*)>
<!ELEMENT illustr_title (#PCDATA)>

<!-- media -->
<!ELEMENT media (media_location, type, media_comment?)>
<!ATTLIST media
                    name ID #REQUIRED>
<!ELEMENT media_location EMPTY>
<!ATTLIST media_location
                    location #CDATA #REQUIRED>
<!ELEMENT type (#PCDATA)>
<!ELEMENT media_comment (#PCDATA)>

<!-- media reference -->
<!ELEMENT mediaReference EMPTY>
<!ATTLIST mediaReferences
                    id ID #REQUIRED
                    references IDREF #IMPLIED>

<!ELEMENT illustr_comment (#PCDATA)>
<!ATTLIST illustr_comment
          references IDREF S#IMPLED> <!-- references an mediaReference -->

          <!-- rem :    ref  = 1 comment - 1 media
                        refS = 1 comment - several medias  (note : ref separated by a space)
                        empty= 1 comment for all medias in illustration -->

<!ELEMENT question (question_description?, answersRelationType, markedForPostanalysis,
matchTreshold, noiseTreshold, acceptanceTreshold, illustration*, suggested_answer*,
expected_answer+, answer*, answerComment*)>
<!ATTLIST question
                    id NMTOKEN #REQUIRED>
<!ELEMENT question_description (#PCDATA)>
<!ELEMENT answersRelationType (#PCDATA)>
<!ELEMENT markedForPostanalysisEMPTY>
<!ATTLIST markedForPostanalysis
                    value (yes | no) #REQUIRED>
```

```
<!ELEMENT matchTreshold (#PCDATA)>
<!ELEMENT noiseTreshold (#PCDATA)>
<!ELEMENT acceptanceTreshold (#PCDATA)>
<!ELEMENT suggested_answer (#PCDATA)>
<!ATTLIST suggested_answer
                value NMTOKENS #REQUIRED>
<!ATTLIST suggested_answer
                                id ID #REQUIRED>
<!ELEMENT expected_answer (#PCDATA)>
<!ATTLIST expected_answer
                                weight NMTOKEN #REQUIRED
                                mandatory (yes | no) #REQUIRED
                                keywordAccepted NMTOKENS #IMPLIED>
<!ELEMENT answer (#PCDATA)>
<!ATTLIST  answer
                value NMTOKENS #REQUIRED>
```

6.4.3. LE SCÉNARIO (SCN.DTD)

```
<!ELEMENT scenario (description, post-processing?,themeTransformation+)>
<!ELEMENT description (name, author, readers, comment)>
<!ATTLIST description
                                id NMTOKEN #REQUIRED
                                doctype NMTOKEN #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!-- an author -->
<!ELEMENT author (person,author_comment)>

<!-- a person -->
<!ELEMENT author (lastname, firstname, comment)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT firstname (#PCDATA)>
<!-- no need of more information, the rest is in the DB coz the scn has been made by a registered
   user -->
<!ELEMENt comment (#PCDATA)>

<!-- post processing -->
<!ELEMENT post-processing (merged-into-1-doc, compressedPackage)>
<!ELEMENT merged-into-1-doc EMTPY>
<!ATTLIST merged-into-1-doc
                                value (yes | no ) #REQUIRED>
<!ELEMENT compressedPackage EMTPY>
<!ATTLIST compressedPackage
                                value (yes | no ) #REQUIRED>

<!-- theme Transformation -->
<!ELEMENT themeTransformation (themeName, options, conversion*, transformation*)>
<!ELEMENT themeName (#PCDATA)>
<!ELEMENT options (encryption)>
<!ELEMENT encryption (type, encryption_options)>
<!ATTLIST encryption
                                value (yes | no) #REQUIRED>
<!ELEMENT type (#PCDATA)> <!-- ex : MD5 -->
<!ELEMENT encryption_options (#PCDATA)>

<!-- conversion -->
<!ELEMENT conversion (typeIn, typeOut, compression)>
<!ELEMENT typeIn (#PCDATA)>
```

```
<!ELEMENT typeOut (#PCDATA)>
<!ELEMENT compression (#PCDATA)>

<!-- transformation -->
<!ELEMENT transformation (type, docIn, sheet?, param*, docOut, link*)>
<!-- type = xsl, fo, smil -->
<!ELEMENT docIn (#PCDATA)>
<!ELEMENT sheet (#PCDATA)>
<!ELEMENT docOut (#PCDATA)>

<!-- link -->
<!ELEMENT link (#PCDATA)>
<ATTLIST link
                        ref CDATA #REQUIRED>

<!-- param -->
<!ELEMENT param EMPTY>
<!ATTLIST param
                        name NMTOKEN #REQUIRED
                        value CDATA #REQUIRED>
```

RÉFÉRENCES

7. RÉFÉRENCES

7.1. BIBLIOGRAPHIE

- [anglo] – Jesus Anglo Lopez. Morphologie mathématique et indexation d'images couleur. Application à la microscopie en biomédecine. Thèse pour obtenir le grade de Docteur de l'école des Mines de Paris, Paris, 2003
- [arm04] – F. Arman. Composition multimédia en support à la démarche diagnostique. Mémoire de fin d'études en informatique , FUNDP, 2004
- [bge02] – B. Georges. La télémicroscopie en cytologie hématologique. Mémoire de fin d'études en informatique, FUNDP, 2002
- [boc08] – X. Bocken. Amélioration d'une plateforme de microscopie virtuelle. Mémoire de fin d'études en informatique, FUNDP, 2008
- [dec04] – G. Decock. Composition automatique de frottis numériques – Application aux gigaimages. Mémoire de fin d'études en informatique, FUNDP, 2004
- [desvi] – Michel Desvignes, Jala M. Fadili, Frederic Bataille. Mesure de netteté par Transformée en ondelettes. Définition et comparaison pour l'autofocus de caméra. Caen/Grenoble, 2003
- [gruy01] – Walter de Gruyter. Guidelines for blood smear preparation and staining procedure for setting up an external quality assessment scheme for blood smear interpretation. Part 1 : Control material. Berlin, 2004
- [gruy01] – Walter de Gruyter. Guidelines for setting up and External Quality Assessment Scheme for blood smear interpretation. Part 2 : survey preparation, statistical evaluation and reporting. Berlin, 2006
- [emim] – FUNDP/DGTRE. EMIM : Emission Multimédia des données en provenance de plateaux techniques d'Imagerie Médicale. Résumé. Projet de recherche
- [jad05] – C. Jadoule. Imagerie Médicale – Etude et développement d'un système de télémicroscopie virtuelle. Mémoire de fin d'études en informatique, FUNDP, 2005
- [klass01] – Dr. Kassen. Venus-1 for ITK positionning controller – Command language.
- [klass02] – Dr. Kassen. Winpos – User interface for ITK motion controllers – Manual.
- [olym01] – Olympus. DP71 microscope digital camera – Instructions
- [olym02] – Olympus. MultiControl 2000 – Documentation. Détails sur les commandes du MultiControl, 1993.
- [olym02] – Olympus. U-AF, Système autofocus pour microscopes d'emploi

général (dispositif de mise au point motorisé utilisant la méthode U-ZD). Mode d'emploi

- [olym04] – Olympus. U-IFRS(carte RS232C)/U-IFGP(carte GPIB) – Informations complémentaires
- [pee07] – C. Peeters. Microscopie virtuelle – Localisation, extraction et classification des globules blancs dans un frotti sanguin. Mémoire de fin d'études en informatique, FUNDP, 2007
- [per06] – B. Permentier. Etude et développement d'un système de télémicroscopie virtuelle. Mémoire de fin d'études en informatique, IESN, 2006.
- [zuy03] – L. Zuyderhoff. Composition automatique de frottis numériques. Mémoire de fin d'étude en informatique, FUNDP, 2003

7.2. WEBOGRAPHIE

- [devli01] – developerlife.com. Xml, Java, Databases and the Web. http://developerlife.com/tutorials/?p=33
- [devli02] – developerlife.Com. Sax tutorial http://developerlife.com/tutorials/?p=29
- [dev] – Developpez. Com. Articles sur les technologies xml (xpath, xsl/xslt, dtd, etc). http://xml.developpez.com/cours/
- [disa] – Yann D'Isanto. La sérialisation binaire en Java. http://ydisanto.developpez.com/tutoriels/j2se/serialisation/partie1/ http://developerlife.com/tutorials/?p=33
- [eck] – Bruce Eckel. Thinking in Java. Third Edition http://sina.sharif.edu/~abolhassani/courses/ap/tij3.pdf
- [flu] – Java design patterns reference and examples. http://www.fluffycat.com/Java-Design-Patterns/
- [hemat] – formation continue destinée aux professionels de la biologie médicale. « http://www.e-hematimage.com »
- [humc] – Jason Hunter, Brett MacLaughlin. Java+xml=jdom. 2000. http://www.jdom.org/dist/docs/presentations/mtvjug04262000.pdf
- [jea] – Jean Etienne. Explication sur les différents ports. Ports Séries. http://www.poirrier.be/~jean-etienne/notes/structordi/ports/
- [lewi] – S. M. Lewis. Le système international d'évaluation externe de la qualité en hématologie organisé par l'OMS. 1988
- [meri] – Meric Sébastien. Lecture d'un flux xml via Sax. http://smeric.developpez.com/java/cours/xml/sax/

- [mil] – milton : an open-source server-side webdav api for Java. http://milton.ettrema.com/index.html

- [shar] – Shari Jones. An introduction to Jdom. 2001. http://xml.sys-con.com/node/40219
- [sun] – Sun. Complete tutorial on Swing. http://java.sun.com/docs/books/tutorial/uiswing/components/index.html
- [wik01] – Wikipedia. La télémédecine http://fr.wikipedia.org/wiki/Télémedecine
- [wik02] – Wikipedia. The Dublin's core. http://fr.wikipedia.org/wiki/Dublin_Core
- [wik03] – Wikipedia. Le traitement d'images. http://fr.wikipedia.org/wiki/Traitement_d%27images
- [w3s01] – w3Schools. Spécifications sur le langage XSL. http://www.w3schools.com/xsl/
- [w3s02] – w3Schools. Le langage dtd http://www.w3schools.com/dtd/
- [w3s03] – w3Schools. Les schémas xml http://www.w3schools.com/schema/schema_intro.asp
- [w304] – w3Schools. Smil. http://www.w3schools.com/smil/default.asp
- [zvon01] – zvon. Le langage xsl http://www.zvon.org/xxl/XSLTutorial/Output_fre/contents.html