Table des matières

Table des figures

Remerciements

Avant toute chose, je tiens à remercier mes responsables au sein de l'Observatoire Astronomique de Strasbourg, Mireille LOUYS et André SCHAAFF. En premier lieu pour m'avoir accepté pour ce stage et aussi pour l'aide et le suivi qu'ils m'ont accordé. Aussi je remercie François BONNAREL pour avoir pris de son temps pour m'expliquer le fonctionnement du serveur d'images ALADIN qu'il a développé.

Introduction

Ce rapport concerne mon stage de Licence Professionnelle des Métiers de l'Informatique. Celui-ci s'étendait du 7 avril au 26 juillet 2003. Il a été accompli à l'Observatoire Astronomique de Strasbourg et plus particulièrement au Centre de Données Astronomiques de Strasbourg (CDS). Ce rapport est composé de la manière suivante : pour commencer une présentation du centre de recherche, par la suite je présenterai la démarche que j'ai suivi par rapport au projet et enfin je décrirai plus précisément le projet en lui-même.

Chapitre 1

Présentation du laboratoire

1.1 Structure juridique

L'Observatoire de Strasbourg est une Unité de Formation et de Recherche (UFR) de l'Université Louis Pasteur. L'Observatoire de Strasbourg est une Unité Mixte de Recherche du CNRS et de l'Université Louis Pasteur (UMR 7550). [10]

1.2 Présentation générale

L'Observatoire est situé sur le campus de l'Esplanade ; ses bâtiments font partie du campus historique de l'université de Strasbourg.

Enseignements dispensés :

- DEA « Analyse et Traitement des Données sur les Milieux Astronomiques »
- DEUG Sciences et autres DEUG (enseignements d'ouverture).
- Licence de Géosciences.
- Maîtrise de Géosciences, Maîtrise de Physique, Maîtrise de Sciences Naturelles.
- Préparation au CAPES et à l'Agrégation.
- DESS Applications des Technologies Spatiales.
- Diffusion de la Culture

Le Planétarium est ouvert au public pour la vulgarisation de l'Astronomie.

L'Observatoire se compose de quatre équipes de recherche :

1.2.1 Hautes énergies

L'équipe Astrophysique des Hautes énergies [11] a pour thème l'étude des astres et sites de l'univers émetteurs de photons de haute énergie. Cette thématique générale recouvre des aspects variés, comme l'étude des astres compacts en fin d'évolution, la physique de leur activité, les phénomènes de haute énergie intéressant les étoiles jeunes ou le soleil, ou l'étude de ces phénomènes à l'échelle galactique, qu'il s'agisse de l'activité des noyaux de galaxies, de l'émission X des diverses populations stellaires ou de celle de structures plus globales comme les zones de sursaut de formation d'étoiles.

Ses recherches se sont largement appuyées sur les données acquises par le satellite ROSAT et s'appuieront dans l'avenir sur celles des satellites X de nouvelle génération, tout spécialement XMM. Le groupe hautes énergies de Strasbourg participe au Survey Science Center (SSC) de XMM, une implication qui lui permet aujourd'hui de bénéficier de temps garanti et lui facilitera l'accès au temps ouvert. L'équipe a aussi une assez forte activité théorique, avec une composante numérique. Les thèmes abordés retrouvent parfois ceux développés par d'autres équipes de l'observatoire, qu'il s'agisse des populations stellaires, de l'activité des étoiles, ou de la thématique de l'équipe CDS.

1.2.2 Physique stellaire : atmosphères, environnements, évolution

Les recherches menées par l'équipe Populations Stellaires et évolution Galactique [18] recouvrent un domaine étendu, incluant les étoiles, les milieux interstellaires, la Galaxie et les galaxies proches. Dans le domaine stellaire, on retrouve la plupart des enjeux scientifiques recensés par l'ASPS, et l'ensemble des objectifs prioritaires. Ces thèmes ont été abordés en faisant appel à des données nouvelles, comme les catalogues Hipparcos et Tycho (publiés en 1997), ou des observations réalisées avec des instruments hautement performants (spectrographe Elodie/T193, SOFI/NTT, HST). Les recherches menées par l'équipe Populations Stellaires et évolution Galactique recouvrent un domaine étendu, incluant les étoiles, les milieux interstellaires, la Galaxie et les galaxies proches. Dans le domaine stellaire, on retrouve la plupart des enjeux scientifiques recensés par l'ASPS, et l'ensemble des objectifs prioritaires. Ces thèmes ont été abordés en faisant appel à des données nouvelles, comme les catalogues Hipparcos et Tycho (publiés en 1997), ou des observations réalisées avec des instruments hautement performants (spectrographe Elodie/T193, SOFI/NTT, HST).

1.2.3 Galaxies

Les activités de l'équipe [12] sont centrées sur les problèmes de la structure du Groupe Local, de ses populations stellaires et sur la dynamique gravitationnelle. De plus, l'équipe possède un savoir faire sur les outils statistiques d'analyse de données et sur les méthodes inverses non paramétriques. Un des objectifs consiste à combiner les informations d'évolution des populations stellaires et celles de dynamique afin de reconstituer les événements déterminants liés aux processus de formation et d'évolution galactique.

Les thèmes déjà engagés sont l'étude des populations stellaires de notre propre Galaxie comme les queues de marée des amas globulaires, celle de la galaxie naine du Sagittaire ou encore l'identification des naines blanches ultra-froides du halo. De même, l'équipe étudie des problématiques similaires liées aux populations stellaires dans des galaxies proches.

Les autres thèmes étudiés sont la cinématique et la dynamique gravitationnelle et les problèmes de stabilités associés. Notre expertise conduit à des coopérations sur les domaines où la physique gravitationnelle est un élément dominant : évolution et structure de notre Galaxie, environnement des trous noirs, stabilité des galaxies disques, dynamique du milieu inter-galactique.

Enfin, la réduction des observations mais aussi leur interprétation et leur comparaison aux modèles nécessitent de mettre en place de nouveaux outils d'analyse statistique : les techniques actuelles font appel aux méthodes inverses non paramétriques.

1.2.4 Le centre de données (CDS)

L'activité de recherche du CDS [8] s'est concentrée sur l'étude de la dynamique galactique et des populations d'étoiles binaires, sur une participation importante à la mission HIPPARCOS de l'Agence Spatiale Européenne, ainsi que sur le développement de méthodologies nouvelles applicables à l'analyse et au traitement de données astronomiques.

1.3 Organisation du personnel

- Directeur : Jean-Marie HAMEURY
- Responsable administratif : Sandrine LANGENBACHER
- 23 enseignants et chercheurs
- 20 personnels ingénieurs, techniciens et administratifs

1.4 Centre de données astronomique (CDS)

J'ai effectué mon stage au sein du Centre de Données astronomiques de Strasbourg (CDS) qui est un centre de données dédié à la collection et à la distribution dans le monde entier de données astronomiques.

Le CDS héberge la base de données SIMBAD, la base de référence mondiale pour l'identification d'objets astronomiques. Le but du CDS est de :

- rassembler toutes les informations utiles, concernant les objets astronomiques, disponibles sous forme informatisée : données d'observations produites par les observatoires du monde entier, au sol ou dans l'espace ;
- mettre en valeur ces données par des évaluations et des comparaisons critiques ;
- distribuer les résultats dans la communauté astronomique ;
- conduire des recherches utilisant ces données.

Le CDS joue, ou a joué, un rôle dans d'importantes missions astronomiques spatiales : contribuant aux catalogues d'étoiles guides, aidant à identifier les sources observées ou organisant l'accès aux archives, etc. Le CDS contribue au XMM Survey Science Center, sous la responsabilité de l'équipe "Hautes-Energies" de l'Observatoire de Strasbourg.

Le CDS a signé des accords d'échanges internationaux avec les organismes suivants :

- NASA,
- National Astronomical Observatory (Tokyo, Japon),
- l'Académie des Sciences de Russie,
- le réseau PPARC Starlink au Royaume-Uni,
- l'Observatoire de Beijing (Chine),
- l'Université de Porto Allegre au Brésil,
- l'Université de La Plata en Argentine,
- InterUniversity Center for Astronomy and Astrophysics (Inde).

Le CDS est membre de la Fédération des Services d'Analyse de Données Astrophysiques et Géophysiques.

Le CDS coopère avec l'Agence spatiale Européenne (transfert au CDS du service de catalogues du projet ESIS : le projet VizieR), et avec la NASA : en particulier le CDS abrite une copie miroir du Système de Données Astrophysiques (ADS) et ADS abrite une copie miroir de SIMBAD au SAO. Le CDS contribue aussi au projet NASA Astro-Browse. Le CDS abrite aussi les copies miroirs Européennes des journaux de l'American Astronomical Society (AAS).

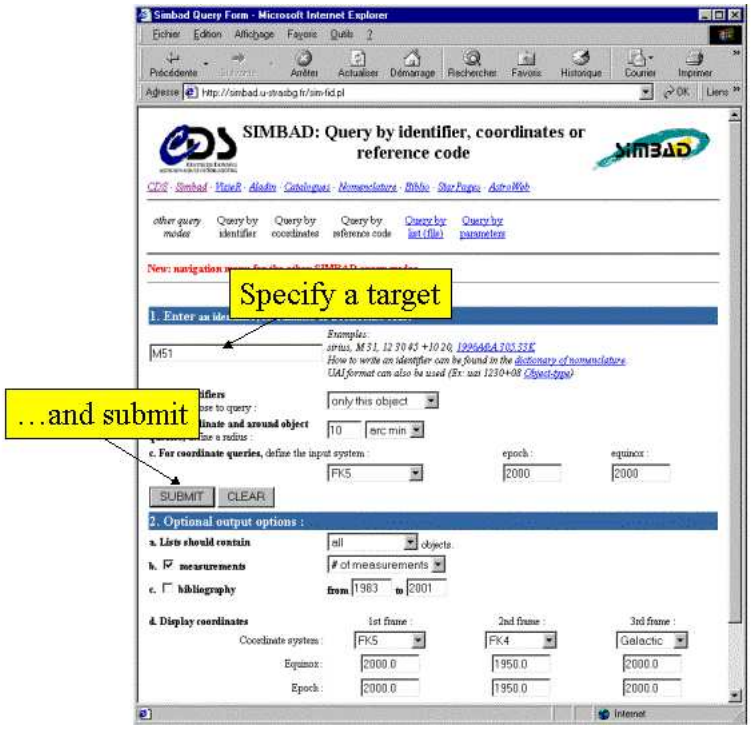

FIG. 1.1 – Page Web permettant d'effectuer une requête sur Simbad

1.5 Les Services du CDS

1.5.1 Simbad

Simbad est une base de données de référence pour les identifications et la bibliographie d'objets astronomiques. [17]

Simbad contient plus 7,5 millions d'identificateurs pour plus de 2,8 millions d'objets différents. Pour chaque objet figurent dans la base quelques mesures (position, magnitude dans différents domaines de longueurs d'ondes), ainsi que les références bibliographiques où l'objet est cité (plus de 110 000 articles sont concernés).

L'utilisateur peut choisir le format du fichier où seront entreposés les résultats de la requête(Fig. 1.1). En effet, Simbad peut générer des fichiers HTML, XML ou xls (fichiers Excel).

Cet ensemble de données résulte d'un long travail d'identification croisée entre de nombreux catalogues, listes d'objets et articles de journaux, entrepris au début des années 1980, et constamment développé et mis à jour depuis.

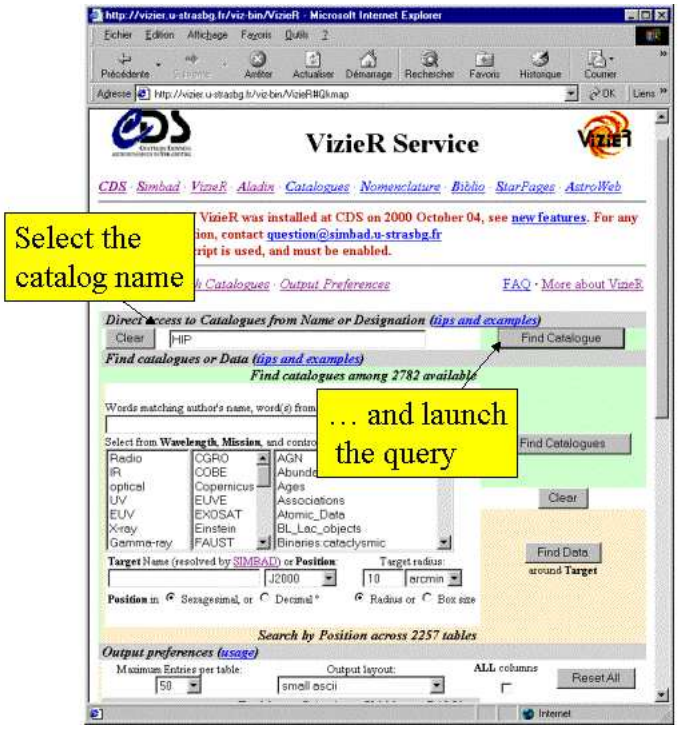

FIG. 1.2 – Page Web permettant d'effectuer une requête sur VizieR

1.5.2 VizieR

VizieR est une base de données rassemblant plusieurs milliers de catalogues astronomiques sous un format homogène. [20] Une description standardisée du contenu des catalogues permet leur inclusion dans un système de gestion de base de données (SGBD) relationnel. Un ensemble de liens, entre les tables de VizieR, et avec des services externes (bibliographiques, archives externes, serveurs d'images), permettent de naviguer entre les données des catalogues et d'autres données associées (Fig. 1.2). Il faut noter que les très grands catalogues (plus de 10^7 enregistrements) ne peuvent pas être gérés par un SGBD relationnel pour des raisons de performances. Des outils spécifiques doivent être utilisés.

1.5.3 Aladin

Aladin est un atlas interactif du ciel permettant d'accéder simultanément à des images numérisées du ciel, ainsi qu'à des catalogues et bases de données astronomiques. [4]

Cet outil permet de superposer, sur des images du ciel optique, les objets présents dans Simbad, des sources de catalogues contenus dans VizieR, mais aussi d'autres données, locales ou situées sur des serveurs distants (archives, HST, …).

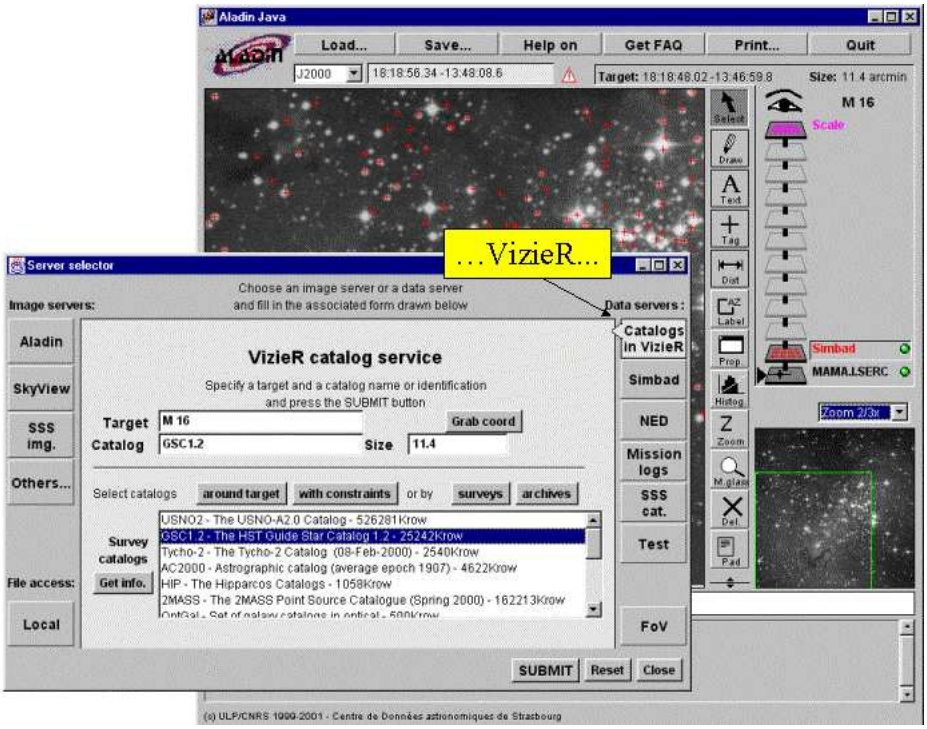

FIG. 1.3 – Exemple d'utilisation d'Aladin

Chapitre 2

La démarche par rapport au projet

Le projet s'incrit dans le contexte du projet IDHA[1] *qui fait partie de l'Observatoire Astronomique Virtuel. Il est composé de deux parties : génération semi-automatique de schema XML et Mise en oeuvre d'un service Web SOAP. Ces deux parties n'ont pas de liens directs entre elles.*

Le découpage des 14 semaines de stage est : 70% partie 1 et 30% partie 2.

2.1 Génération semi-automatique de schéma XML

2.1.1 Situation

Les informations utilisées par la communauté astronomique sont très hétérogènes : tables de mesures, catalogues d'étoiles, de galaxies, collections d'images du ciel à différents niveaux de précision et dans différentes longueurs d'onde. Au vu de la quantité de données, les images du ciel sont stockées sur des serveurs différents avec des modèles spécifiques.

Le CDS joue un rôle moteur dans la mise en place de liens entre les services d'information en astronomie et dans le développement de standards, en partenariat avec les autres producteurs de service, en particulier avec les grands observatoires et les journaux spécialisés. Il participe très activement au projet d'« Observatoire Astronomique Virtuel »en cours en Europe et aux États-Unis.

[1] Image Distribuées Hétérogènes pour l'astronomie

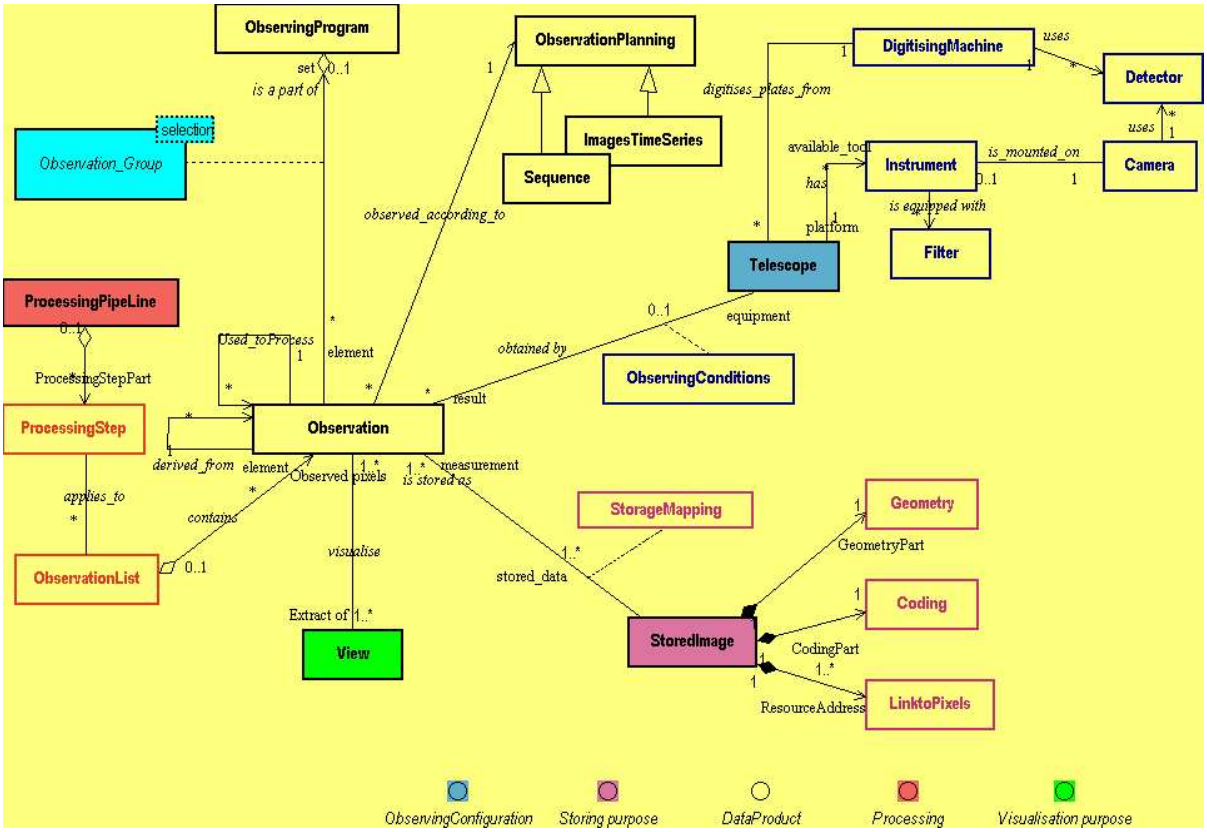

FIG. 2.1 – Modèle de données IDHA

2.1.2 Base de solution

Ce niveau d'interopérabilité nécessite de construire des standards d'échange permettant de décrire les différents types d'informations et leurs liens logiques. L'observatoire a proposé un modèle préliminaire adapté à la description d'images d'archives, sous la forme de modèle UML (Fig. 2.1). Il s'agit de dériver de ce modèle, des descriptions partielles sous forme de schéma XML, permettant de répondre aux différentes utilisations prévues des images. Ceci se fera grâce à une application Java utilisant un processeur XSLT pour appliquer une feuille de style XSL, et en sortie récupérer un schéma XML validé.

2.1.3 Avantages de cette solution

Cette solution a pour but de construire des standards d'échange, elle se doit donc d'utiliser des technologies standards et portables.

Ainsi l'application permettant de générer le schéma XML est écrite en Java, ce qui permet de créer des applications offrant une meilleure sécurité au niveau de l'exécution.

Cette sécurité est possible surtout par la vérification de tout le code avant son exécution. Elle vient aussi de l'absence, dans le langage, de la notion de pointeur.

Java permet aussi de créer des applications adaptables à des environnements changeants, simplement par le fait que le code peut être téléchargé avant l'exécution, à partir de n'importe quel point serveur d'un réseau.

Cette notion d'adaptabilité doit être reliée à la notion de portabilité assurée par la machine virtuelle Java (dans la mesure où celle-ci existe pour chaque type de machine). Cette portabilité est clairement mise en avant dans le leitmotiv de Java :

write once, run everywhere

Du point de vue du programmeur, Java permet de réduire le temps de développement d'une application grâce à la réutilisabilité du code développé et la librairie graphique Swing très facile à appréhender.

De plus nous avons fait le choix des schémas XML car les définitions de type de document (DTD) ont rapidement laissé apparaître des carences dans le domaine des contraintes de structures et de données. Aussi la syntaxe des DTD est issue du langage EBNF (Extended Backus Naur Form), beaucoup plus compliqué et lourd que ne l'est le langage XML.

Le langage de schéma XML apporte une grande souplesse et une puissance inégalée dans la définition des documents XML. La fonctionnalité la plus remarquable des schémas XML est la prise en charge des types de données garantissant le contenu a affecter à un élément XML et apportant une validation plus efficace, non seulement sur la structure du document, mais aussi sur son contenu. Ainsi, une balise destinée à accueillir une date sera définie comme telle dans le schéma et par la suite sera en mesure de délivrer sa valeur à un programme sans soucis de conversion d'une chaîne de caractères en date.

Le modèle de contenu devient davantage exhaustif avec les schémas XML qu'il ne l'était auparavant avec les DTD. En effet, outre que les schémas définissent tous les éléments et la structure constituant un document XML, ils déterminent également le nombre d'occurrences des éléments, gèrent les contenus mixtes, les éléments nommés, les groupes d'attributs ainsi que des annotations utilisées pour la création d'une documentation à propos du modèle de contenu.

2.1.4 Moyens

Tous les développements ont été fait sous Linux Mandrake 9.0 en Java (j2sdk1.4.2).

Les outils suivant ont été utilisés :

– Environement de développement : Borland JBuilder 8.0 [13]

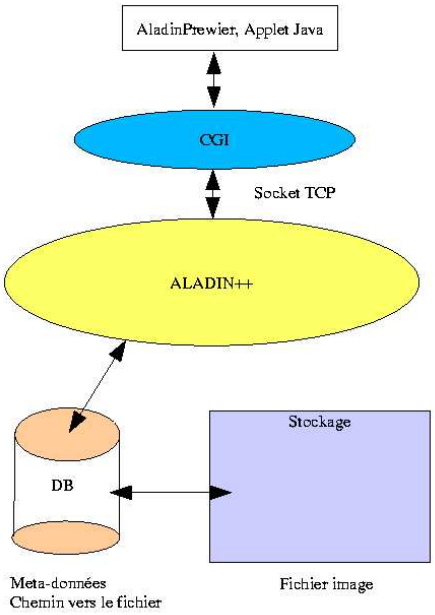

FIG. 2.2 – Fonctionnment d'Aladin

- Processeur XSL : Xalan 2.5.1 [2]
- Parseur XML et validateur de schémas : Xerces 2.0 [1]
- Installeur : InstallAnyWhere 5.5 [3]
- LaTeXpour l'élaboration de ce rapport

2.2 Mise en place d'un service Web SOAP

2.2.1 Situation

Actuellement, Aladin est interrogeable par deux moyens (Fig. 2.2) :

- un applet qui permet de rechercher des images, de superposer des entrées de catalogues astronomiques, de placer des filtres, de calculer des distances, ...
- une interface web permettant seulement de visualiser les images.

2.2.2 Base de solution

Il s'agit de déployer un service Web SOAP qui interrogera le servueur d'image Aladin, dans un premier temps en encapsulant le CGI actuel, puis dans un second temps comminiquera avec Aladin directement pour socket TCP.

Aladin génère, suite à une interrogation sur une position dans le ciel, un document

XML au format VOTable spécifique au milieur Astronomique. Ce document pourra être traité à l'aide du parser SavotParser dévoloppé par mon responsable de stage André SCHAAFF, pour ensuite instancier les classes Java correspondant au modèle de données IDHA (Fig. 2.1).

2.2.3 Avantages de cette solution

SOAP (Simple Object Access Protocol) offre un moyen simple et léger d'échanger des informations structurées et typées entre pairs dans un environnement décentralisé et distribué utilisant XML. Cela permet d'utiliser SOAP dans un grand nombre de systèmes allant des systèmes de messagerie aux appels de procédure à distance (RPC).

SOAP permet aux applications de communiquer entre elles d'une manière unique grâce à l'infrastructure du Web. Aujourd'hui, l'énorme majorité du trafic Web est constituée par l'envoi de pages HTML depuis des serveurs vers des navigateurs. Avec SOAP, le Web peut devenir bien plus que cela : SOAP permet aux applications de communiquer entre elles et fourni l'infrastructure pour connecter des sites Web à des applications pour créer des services Web.

Les services Web relient des sites aux applications pour apporter des fonctionnalités que des composants individuels ne pourraient pas fournir. Par exemple, imaginez une application qui relie entre elles les informations que vous recevez d'un site contenant des données sur le trafic routier grâce à une destination qui lui aurait été fournie au préalable par votre gestionnaire de calendrier. Cela pourrait donner une application qui vous rappelle de partir 15 minutes plus tôt à votre rendez-vous, et vous indique et recalcule le trajet idéal en tenant compte du trafic routier, des accidents, etc.

SOAP apporte la manière pour chacun de ces services de présenter les fonctionnalités et communiquer avec d'autres services. Grâce à SOAP, il est possible de lier les services entre-eux comme des composants et de construire ce genre d'application très rapidement. SOAP permet à des composants homogènes ou hétérogènes de communiquer par Internet d'une manière simple. Il permet une communication de service à service, de composant à service ou de composant à composant.

Il y a d'autres avantages à utiliser HTTP et le XML. Cela permet de franchir aisément les firewalls et proxy et facilite le traitement en cas de filtrage. XML pour structurer les requêtes et les réponses, indiquer les paramètres des méthodes, les valeurs de retours, et les éventuelles erreurs de traitements. SOAP n'est tenu par aucun système d'exploitation, langage de programmation ou modèle objet. Par exemple, il permet aux applications CORBA de communiquer avec des applications DCOM tournant sur n'importe quelle plate-forme. Un autre avantage est que SOAP est relativement simple à implémenter. En

partant de la spécification SOAP publiée, il est possible d'implémenter une application SOAP qui sera en mesure de tourner en quelques jours seulement.

XML joue un rôle prépondérant dans SOAP, on peut distinguer plusieurs éléments :

- une enveloppe, expliquant comment la requête doit être traitée et présentant les éléments contenus dans le message.
- un ensemble de règles de codage, permettant de différencier les types de données transmises.
- une convention permettant de représenter les appels aux procédures de traitement et les réponses.

Même si SOAP ressemble beaucoup en termes de fonctionnalités à des protocols comme IIOP pour CORBA, ORPC pour DCOM, ou JRMP (Java Remote Method Protocol) pour JAVA, il possède un avantage indéniable. En effets, SOAP utilise un mode texte alors que les autres fonctionnent en mode binaire, cela facilite le passage des équipements de sécurité.

Pour expliquer le fonctionnement des webservices, il convient de distinguer plusieurs étapes :

- Recherche dans un annuaire UDDI : le client cherche un service particulier, il s adresse à un annuaire qui va lui fournir la liste des prestataires habilités à satisfaire sa demande. L annuaire UDDI peut être comparé à un moteur de recherche sauf que les documents sont remplacés par des services.
- Recherche de l'interface du composant à contacter : une fois la réponse reçue (en XML) de l annuaire, le client va chercher à communiquer via une interface. Cette interface décrite en langage WSDL fournit l'ensemble des services disponibles. Elle offre la possibilité de vérifier que le contrat correspond bien aux besoins demandés.
- Invocation du service : le client doit maintenant passer les paramètres attendus par le service et assurer la communication avec le serveur. L'outil utilisé pour cela est le Proxy, c'est l'objet qui permet la communication entre le client et le serveur. Il est généré par le client en utilisant l'interface WSDL.

WSDL (Web Service Definition Langage) est un format de représentation des interfaces de services Web en XML. Une analogie avec CORBA peut être faite, en effet WSDL est la représentation XML du langage IDL (description d interfaces). WSDL a deux rôles prépondérants :

- Il sert de référence à la génération de Proxies
- Il assure le couplage entre le client et le serveur par le biais des interfaces.

2.2.4 Moyens

Pour la réalisation de ce projet, les moyens suivants ont été utilisés :

- moteur de servlettes : Apache Tomcat 4.1.24 [19]. Tomcat est un servlet - c'est à dire un programme Java tournant sur un serveur - qui gère d'autres servlets et des pages serveur Java (JSP ou Java Server Pages) permettant de générer des pages web dynamiques. Il peut être utilisé seul ou couplé avec un serveur (dont la liste est spécifiée dans la documentation). Tomcat répond aux spécifications de Sun concernant les servlets et leurs gestionnaires. Il supporte à ce titre l'ensemble des classes définies par Sun dans le document " Java API Servlet Spécifications ". Tomcat est totalement gratuit et généralement utilisé en couplage avec un serveur Apache. Ecrit en Java, il nécessite, pour pouvoir fonctionner, la présence d'une machine virtuelle Java, et plus précisément du SDK - Sun Developpement Kit - complet. Ceci entraîne le fait que Tomcat est totalement portable et peut être mis en oeuvre sur des systèmes radicalement différents, tels que Linux ou Windows.
- implémentation de la spécification SOAP : Apache Axis [5]. Développée par la fondation Apache, Axis est une nouvelle implémentation de la spécification SOAP, qui succède à Apache SOAP. Cette nouvelle implémentation se veut plus performante, plus modulaire et plus extensible que son prédécesseur.
- parseur de document XML VOTable : SavotParser [16]

Chapitre 3

Le Projet

3.1 Description de la réalisation réelle

3.1.1 Génération semi-automatique de schémas XML

Analyse du modèle de données IDHA

Le projet Images Distribuées Hétérogènes pour l'Astronomie (IDHA) (Fig. 2.1) [9] vise à évaluer et à développer les méthodes et les outils pour intégrer les résultats de requêtes à des services hétérogènes distribuant des images, pour les besoins de l'Observatoire Astronomique Virtuel. Il comporte trois aspects :

- élaboration de méta-données (construction d'un standard XML) ;
- étude de méthodes multiéchelles/multirésolutions d'identification croisées sur images multispectrales, mise au point de prototypes ;
- ébauche d'un réseau de compétences au niveau national.

Analyse du fichier XMI

L'outil de modélisation UML (ObjectEering) utilisé à l'Observatoire permet d'exporter les modèles sous forme de fichier XMI (Fig. 3.1).

Le XML Metadata Interchange Format (XMI) [21] a été proposé en réponse à une proposition de l'Object Management Group (OMG). L'OMG proposait de définir un format d'échange permettant d'importer et d'exporter toute sorte de modèles basés sur le « Meta Object Facility »(MOF) et non seulement des modèles UML.

En établissant un standard pour enregistrer et partager les informations sur la programmation objet, les groupes de développement utilisant divers outils peuvent collaborer

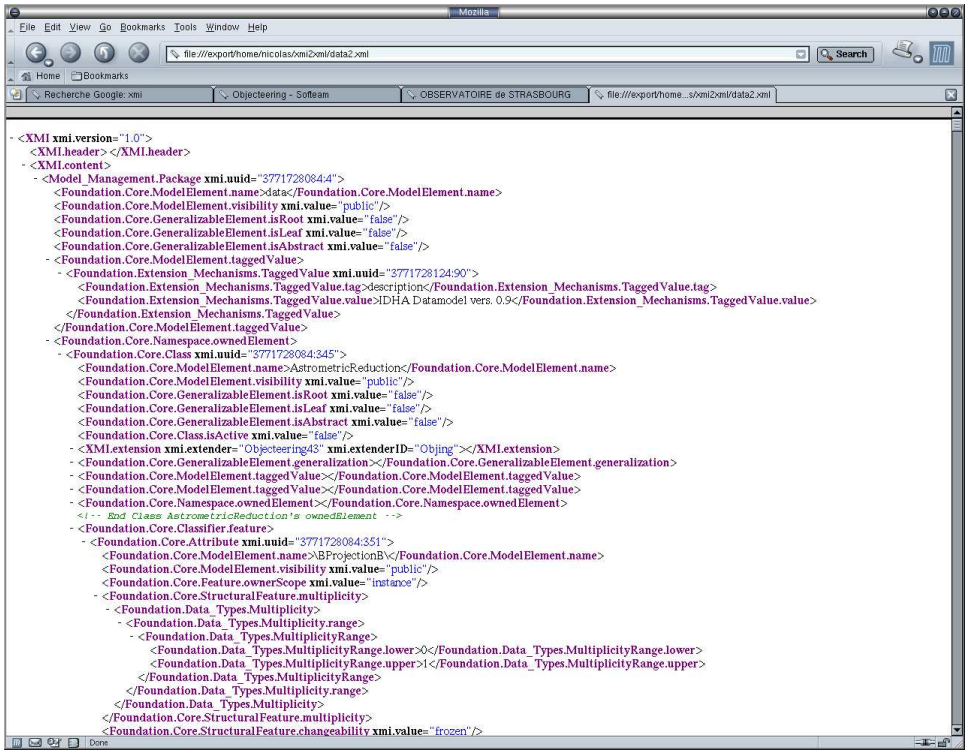

FIG. 3.1 – Extrait du fichier XMI

facilement.

Le fichier XMI généré pour le modèle IDHA est composé de 8 909 lignes. Ceci est le prix à payer de la standardisation. De plus ce fichier contient beaucoup de références croisées entres les éléments, tout ceci rendrait son traitement par l'application à développer très lourd.

Dans notre cas nous n'avons pas besoin de toutes les informations contenues dans ce fichier, c'est pourquoi le choix d'appliquer une feuille de style XSL est vite apparu.

Réalisation de la feuille de style XSL

Cette feuille de style XSL nous permettra d'adapter à nos besoins la structure de document que nous traiterons avec notre application.

La feuille de style est appliquée à l'aide du processeur XSL Xalan dès l'ouverture du fichier XMI et nous permet de réduire la taille du fichier de 8 909 à 2549 lignes.

Le document XML en sortie a la forme suivante :

```
<?xml version="1.0" encoding="UTF-8"?>
<model name="data">
```

```
<interfaces>
<interface name="DataProduct"/>
</interfaces>
<classes>
<class name="AstrometricReduction">
<attributes>
<attribute name="Projection">
<visibility>public</visibility>
<type/>
</attribute>
<attribute name="Mode">
<visibility>public</visibility>
<type>ResourceLink</type>
<description>
Differents types of reduction: local to this
observed region,
global to the reduction region attached
</description>
</attribute>
  </attributes>
<super_classes>
<class name="ProcessingPipeLine"/>
</super_classes>
<relations>
<association>
<linked_class name="Observation_Group(0,*)"/>
</association>
</relations>
</class>
</classes>
</model>
```

Représentation d'un document XML sous forme d'arbre

Une fois notre feuille de style appliquée et notre document XML de sortie généré, le choix de représenter ce dernier sous forme d'un arbre à été fait (Fig. 3.2). En effet tout document XML est déjà structuré sous la forme d'un arbre, donc le composant Java JTree est particulièrement adapté pour représenter ce fichier.

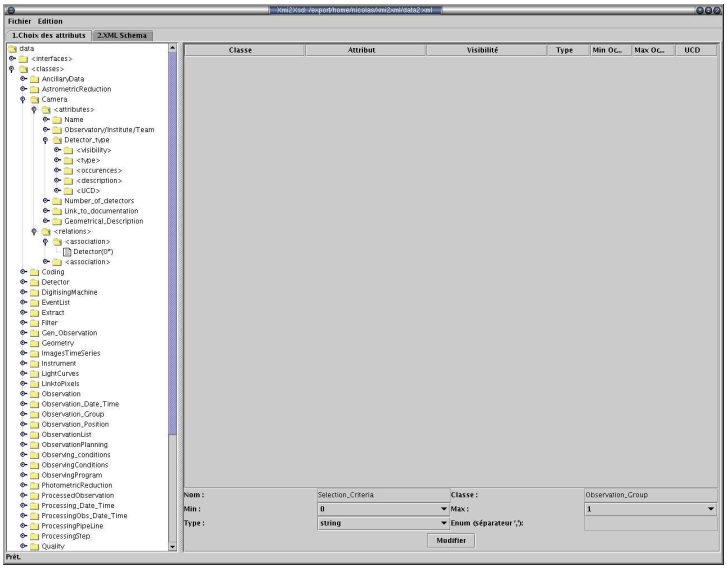

FIG. 3.2 – Exemple d'arbre généré pour un modèle de données

Pour construire cet arbre, le parseur XML Xerces à été utilisé. Xerces Java est un projet Open-Source dirigé par apache.org. Cet outil propose deux interfaces XML :

- interface basée sur un arbre (DOMParser) : le parseur va créé en mémoire centrale un arbre de noeuds. Vous pouvez donc parcourir l'arbre et extraire l'information souhaitée.
- interface basée sur les évènements (SAX Parser) : le parseur déclenche des évènements pour chaque élément significatif. L'application intercepte ces évènements et extraie les données XML.

Exemple Pour le fichier suivant :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE WEATHER SYSTEM "Weather.dtd">
<WEATHER>
  <CITY NAME="New York">
    <TEMP>64</TEMP>
  </CITY>
</WEATHER>
```

Le choix de l'interface basée sur l'arbre génère en mémoire l'arbre Fig. 3.3 et le chois de l'interface basée sur les événements déclenche les événements Fig. 3.3.

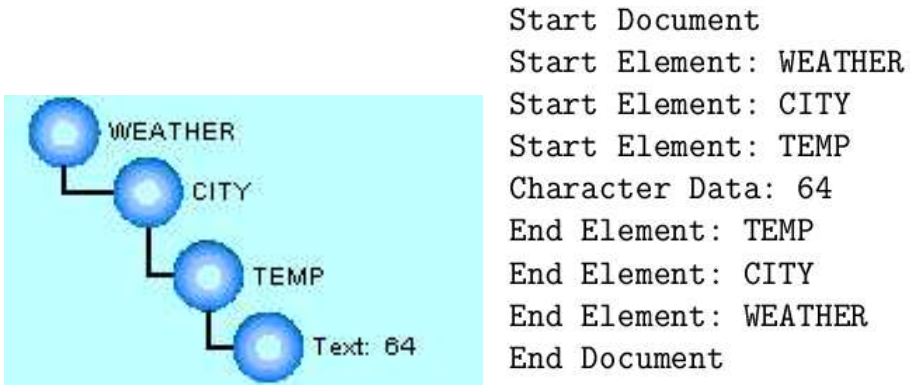

FIG. 3.3 – DOM Parser (à gauche), SAX Parser (à droite)

Sélections des classes à partir de l'arbre

Pour permettre la création de sous-modèles, il devait être possible de sélectionner certaines classes à partir de l'arbre. Lorsqu'une classe est sélectionnée dans l'arbre, il est possible d'insérer tous les attributs de la classe à la liste (Fig. 3.4). Si la classe à insérer hérite et/ou est associée à d'autres classes, de façon à satisfaire les dépendances dans le schéma XML qui sera généré, toutes les classes liées sont automatiquement ajoutées dans la liste.

Certaines propriétés d'un attribut dans la liste sont modifiables telles que le type et les occurrences. La liste des attributs peut êtres sauvegardée de façon à ne pas être obligé de re-sélectionner les classes à chaque exécutions.

Génération du schéma XML

Le schéma XML est généré à partir de la liste des attributs (Fig. 3.5).

Validation du schéma XML

La validation du schéma XML se fait indépendamment de tout document XML associé au schéma. Pour ce faire le parseur XML Xerces[1] a été utlisé.

Si des erreurs sont survenues lors de la validation, un fenêtre s'ouvre indiquant les messages d'erreurs, la localisation et des suggestions pour les résoudre.

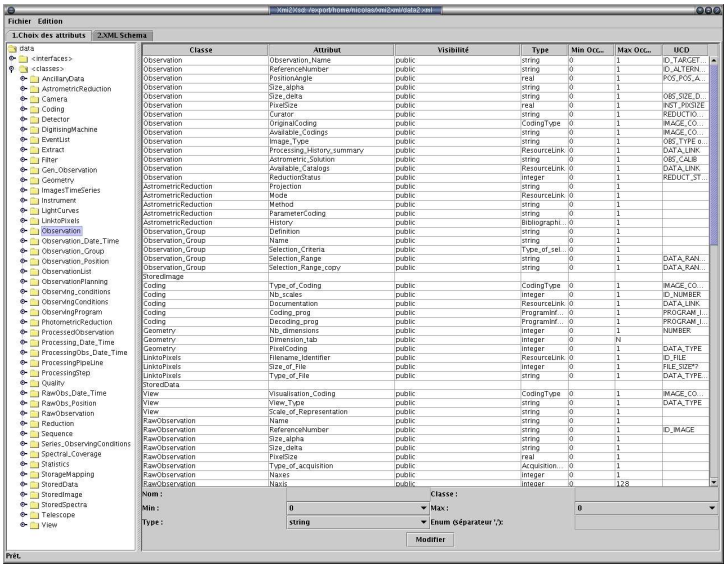

FIG. 3.4 – Exemple de liste d'attributs

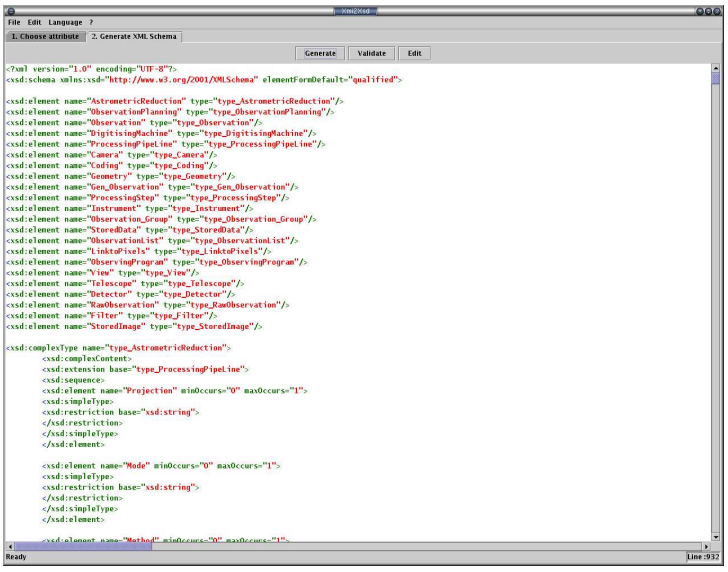

FIG. 3.5 – Exemple de schéma XML généré

Colorisation synthaxique du Schéma XML généré

Pour rendre plus lisible le schéma XML généré, la colorisation synthaxique a été implantée grâce à l'editeur OpenSource Jedit[14]. Cela est possible en réutilisant une partie du code source de cet éditeur tout en respectant la licence sous laquelle il est distribué.

Interface graphique multi-langues

Dans le milieu de l'astronomie, l'anglais est la langue la plus utilisée. Cependant pour ne pas imposer de langue, j'ai fait le choix d'inclure la possibilté de chosir dans quelle langage l'interface graphique sera affichée. Pour le moment seuls le français et l'anglais sont disponibles, il est cependant très facile d'ajouter d'autres langues mais malheureusement mes compétences de traducteur sont très réduites.

Mode ligne de commande

Il est util de pouvoir générer automatiquement un schema XML à partir d'un fichier XMI sans avoir à choisir les attributs que l'on souhaite inclure. Ceci est possbile grâce au mode ligne de commande : `java -jar xmi2xsd.jar <xmi_file> <xsd_file>` (si aucun paramètre n'est passé, l'interface graphique est lancée).

3.1.2 Service Web SOAP

Analyse du format VOTABLE

Le format VOTABLE a été défini pour les échanges de données tabulaires au sein du projet Observatoire Virtuel.

Dans ce contexte, une table est un ensemble non ordoné de lignes, chacune d'un format uniforme, comme indiqué dans les metadonnées de table. Chaque lignes est une sequence des cellules, et chacune de ces dernières sont de types de données primitifs, ou un tableau de types primitifs.

Exemple de fichier VOTABLE :

```
<?xml version="1.0"?>
<!DOCTYPE VOTABLE SYSTEM "http://us-vo.org/xml/VOTable.dtd">
<VOTABLE version="1.0">
  <DEFINITIONS>
```

```
  <COOSYS ID="myJ2000" equinox="2000." epoch="2000." system="eq_FK5"/>
  </DEFINITIONS>
  <RESOURCE>
    <PARAM name="Observer" datatype="char" arraysize="*" value="William Herschel">
      <DESCRIPTION>This parameter is designed to store the observer's name
      </DESCRIPTION>
    </PARAM>
    <TABLE name="Stars">
      <DESCRIPTION>Some bright stars</DESCRIPTION>
      <FIELD name="Star-Name" ucd="ID_MAIN" datatype="char" arraysize="10"/>
      <FIELD name="RA" ucd="POS_EQ_RA" ref="myJ2000" unit="deg"
             datatype="float" precision="F3" width="7"/>
      <FIELD name="Dec" ucd="POS_EQ_DEC" ref="myJ2000" unit="deg"
             datatype="float" precision="F3" width="7"/>
      <FIELD name="Counts" ucd="NUMBER" datatype="int" arraysize="2x3x*"/>
      <DATA>
        <TABLEDATA>
        <TR>
          <TD>Procyon</TD><TD>114.827</TD><TD> 5.227</TD>
          <TD>4 5 3 4 3 2 1 2 3 3 5 6</TD>
        </TR>
        <TR>
          <TD>Vega</TD><TD>279.234</TD>
          <TD>38.782</TD><TD>8 7 8 6 8 6</TD>
        </TR>
        </TABLEDATA>
      </DATA>
    </TABLE>
  </RESOURCE>
</VOTABLE>
```

Utilisation du parseur Savot

En interrogeant le serveur d'images Aladin sur une position dans le ciel, le serveur retourne un document VOTABLE dont les différentes ressources correspondent aux objets du modèle IDHA.

Pour analyser ce document, mon responsable de stage André SCHAAFF a développé un parseur en Java pour les fichier XML au format VOTABLE[16]. Ce parseur surchage

l'API kXML[15] qui est un parseur XML minimal donc peu gourmand en mémoire vive.

Instanciation du modèle IDHA

Après avoir parser le document VOTABLE, le service Web instancie les classes Java (Beans) qui correspondent au modèle IDHA. Ces classes seront renvoyées au client suite à une requête.

Déploiement du Service

Pour déployer un service Web, il faut créer un fichier WSDD (Web Service Deployement Descriptor) qui permet de décrire le service, les fonctions qu'il contient, les types de retour, etc. .

Apache Axis permet aussi pour des petits services, de déployer directement sans fichier WSDD, simplement en copiant les fichiers *.java dans un répertoire spécifique du serveur en les renomant avec l'extension *.jws . Néanmoins cette méthode appellée *Instant Deployement* ne permet pas de paramétrer toutes les options offertes par Apache Axis. Elle ne permet pas :

- l'utilisation des *Java Beans*[1] pour les types évolués
- de ne pas diffuser le code source
- de maîtriser le cycle de vie du service (création de l'objet et desctruction)
- d'agir sur le fichier WSDL engendré
- de contrôler les méthodes exportées

Elaboration d'un client

Une fois le service déployé, un fichier WSDL (Web Service Descriptor Language) est créé et visible par tous les clients potentiels, en accédant à une certaine URL. Ce fichier décrit les fonctions offertes par le service et leur type de retour.

Il existe trois méthodes pour créer un client :

1. le Client Magique : on lance `java org.apache.axis.wsdl.WSDL2Java` avec en paramètre l'url à laquelle est disponible le fichier WSDL. On obtient alors un package consituté de :
   - `Aladin` : interface SDI (Service Definition Interface), c'est l'interface utilisée pour accéder au service

[1]Les Java Beans sont des composants Java persistants construits selon un gabarit de construction spécifique afin de garantir des propriétés de persistance, d'introspection, de configuration et d'assemblage pour construire une application ou une partie d'application Java.

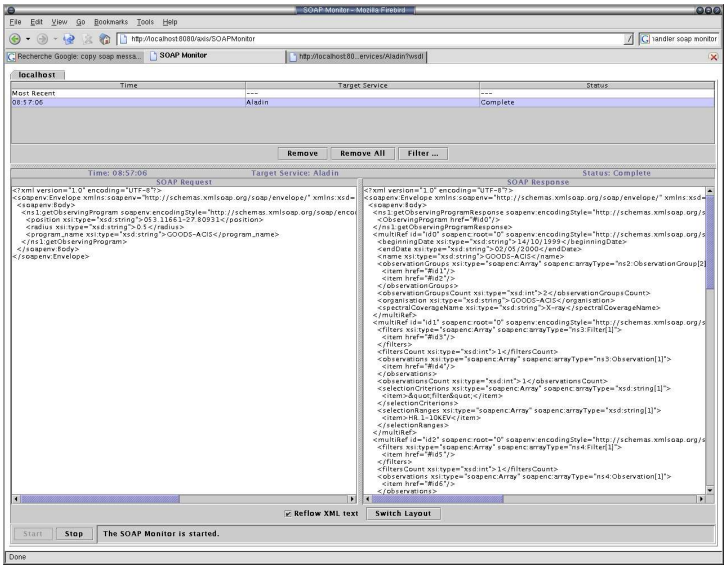

FIG. 3.6 – Messages soap entre le client et le serveur

- `AladinService` : interface de localisation du service (design pattern Abstract Factory)
- `AladinServiceLocator` : implémentation concrète de `AladinService`
- `AladinSoapBindingStub` : permet de traduire les appels locaux em messages SOAP et vice-versa (implémente l'interface `Aladin`)
- toutes les classes permettant de définir les types de retour des méthodes du service.

Il ne reste plus qu'à coder le client en important ce package. Ce client est (en partie) spécifique AXIS.

2. version plus lourde : la solution précédente est statique, il faut engendrer les classes `Java` puis l'appel. L'aspect dynamique consiste à appeler un service web sans le connaître avec le lancement du programme. Cette version est très lourde, il n'y pas de gestion des types, pas d'interfaces, pas de gestion des espaces de noms, etc..

3. version intermédiaire : mécanisme de proxy dynamique basé sur le système de *dynamic proxy* introduit par le jdk 1.3. Un *dynamic proxy* est une classe qui implémente une ou plusieurs interfaces spécifiées **au moment de l'exécution**.

   Le client pour effectuer une requête va envoyer au serveur un message SOAP (Fig.3.6 message de gauche) et le serveur va lui renvoyer un message de réponse ou un message d'erreur (Fig.3.6 message de droite).

3.2 Problèmes rencontrés

3.2.1 Génération semi-automatique de schémas XML

Représentation de l'héritage dans un schéma

La modélisation de l'héritage simple ne pose pas de problèmes dans un schéma XML.

Exemple simple d'héritage :

```
<xsd:complexType name="Point">
<xsd:complexContent>
<xsd:sequence>
<xsd:element name="x" minOccurs="1" maxOccurs="1">
<xsd:simpleType>
<xsd:restriction base="xsd:double">
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="y" minOccurs="1" maxOccurs="1">
<xsd:simpleType>
<xsd:restriction base="xsd:double">
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
</xsd:sequence>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="PointCouleur">
<xsd:complexContent>
<xsd:extension base="Point">
<xsd:sequence>
<xsd:element name="couleur" minOccurs="0" maxOccurs="1">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
```

```
</xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
```

Le problème vient en fait de l'héritage multiple car la spécification des schémas XML ne permet pour un *complexType* d'avoir qu'un seul *complexContent* qui peut contenir lui aussi une seule *extension*.

Après avoir demander conseil sur différents forums et demander à la société Ontogenics [7] qui commercialise un plugin pour l'environnement de développement Eclipse permettant à partir de fichier XMI de générer des schémas XML, la seule solution est de ne pas utiliser l'élément *¡xsd :extension¿* et d'inclure directement les éléments hérités. Pour l'exemple précédent cela donne :

```
<xsd:complexType name="PointCouleur">
<xsd:complexContent>
<xsd:sequence>
<xsd:element name="x" minOccurs="1" maxOccurs="1">
<xsd:simpleType>
<xsd:restriction base="xsd:double">
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="y" minOccurs="1" maxOccurs="1">
<xsd:simpleType>
<xsd:restriction base="xsd:double">
</xsd:restriction>
</xsd:simpleType>
<xsd:element name="couleur" minOccurs="0" maxOccurs="1">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
</xsd:element>
</xsd:sequence>
</xsd:complexContent>
</xsd:complexType>
```

Représentation des relations entre les classes dans un schéma

Il existe trois types de relation entre classes en UML :
- les associations navigables ou non
- les agrégations
- les compositions

Il a fallu extraire du fichier XMI toutes les relations avec leurs cardinalités tout en les classant dans une des trois catégorie ci-dessus.

Validation du schéma

Il existe beaucoup d'applications permettant de valider un document XML par rapport à un schéma mais il a été plus difficile de trouver un outils permettant de valider le schéma lui-même. J'ai découvert System Quality Checker [6] qui est une API Java AlphaWorks IBM©. Elle permettant de valider les schémas, de localiser les erreurs et de donner des suggestions pour la résolution des erreurs.

Cependant cette API n'est pas gratuite il a donc fallu trouver une solution de rechange. En étudiant de plus près le parseur XML Xerces, j'ai découvert qu'il permettait également de valider des schemas XML (`parser.setFeature("http ://apache.org/xml/features/validation/schema-full-checking", true)`) en disposant d'une connection Internet.

3.2.2 Mise en oeuvre d'un service Web

Type de retour des méthodes

Les méthodes du service web retourne des liste d'objets, j'ai donc dans un premier temps voulu utiliser l'object `ArrayList` comme type de retour pour des questions de performances. Ceci n'est évidement pas possible, en effet il ne faut pas oublier que les types de retour des méthodes seront sérialisés dans un message SOAP, qui correspond à un fichier XML, pour permettre à un client, n'étant pas obligatoirement écrit en Java, d'interroger le service. Il est alors tout à fait normal qu'on ne puisse pas utilisé des types spécifiques à Java.

La solution consiste à renvoyer des tableaux d'objets. Cependant lors de l'implémentation, des exceptions se déclenchaient lors de l'exécution du client. Il faut en fait préciser à Axis deux déserialiseurs, un pour le tableau lui-même et l'autre pour l'objet contenu dans le tableau pour que le client puisse à partir du message SOAP reconstituer le tableau d'objets.

```
<typeMapping
        xmlns:ns="urn:Aladin"
        qname="ns:ArrayOf_tns2_ObservingProgram"
        type="java:AladinWS.IDHAModel.ObservingProgram[]"
        serializer="org.apache.axis.encoding.ser.ArraySerializerFactory"
        deserializer="org.apache.axis.encoding.ser.ArrayDeserializerFactory"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
/>
<typeMapping
        xmlns:ns="urn:Aladin"
        qname="ns:ObservingProgram"
        type="java:AladinWS.IDHAModel.ObservingProgram"
        serializer="org.apache.axis.encoding.ser.BeanSerializerFactory"
        deserializer="org.apache.axis.encoding.ser.BeanDeserializerFactory"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
/>
```

Conclusion

J'ai eu la chance de bénéficier d'un excellent encadrement tout au long de ce stage, ce qui m'a permi d'avancer rapidement dans mes travaux.

De plus, j'ai pu découvrir une manière différente de travailler de celle que j'ai pratiquée l'annéé dernière au cours de mon stage de deuxième année de DUT au sein d'une multinationale.

Ainsi ce stage m'a beaucoup appris tant sur le plan technique que sur le plan professionnel. J'ai découvert des technologies nouvelles telles que les schémas XML, les services web SOAP et le calcul distribué en général.

Sur le plan culturel j'ai été initié au monde de l'astronmomie qui m'était inconnu jusqu'à lors. Ce domaine est très intéressant pour un informaticien car il fait appel à toute la puissance d'un système d'information aussi bien sur le plan matériel que logiciel. En effet j'était plutôt habitué à des bases de données de l'odre d'un million du tuples alors que pour le domaine atronomique il est nécessaire d'enregistrer des dixaines de milliards de tuples.

Bibliographie

[1] Xerces 2.0. site web. http ://xml.apache.org/xerces2-j/index.html.

[2] Xalan 2.5.1. site web. http ://xml.apache.org/xalan-j/.

[3] Zero G InstallAnyWhere 5.5. site web. http ://www.zerog.com/.

[4] Service Aladin. site web. http ://aladin.u-strasbg.fr/aladin.gml.

[5] Apache Axis. site web. http ://ws.apache.org/axis/.

[6] System Quality Checker. site web. https ://www.alphaworks.ibm.com/tech/xmlsqc.

[7] Ontogenics Corp. site web. http ://www.ontogenics.com/.

[8] Equipe Centre de Données astronomiques de Strasbourg. site web. http ://astro.u-strasbg.fr/Obs/CDS/CDS.html.

[9] Modèle de données IDHA. site web. http ://alinda.u-strasbg.fr/IDHA/lastmod.

[10] Observatoire Astronomique de Strasbourg. site web. http ://astro.u-strasbg.fr/Obs.html.

[11] Equipe des Hautes Energies. site web. http ://astro.u-strasbg.fr/Obs/ENERGIE/ENERGIE.html.

[12] Equipe Glaxies. site web. http ://astro.u-strasbg.fr/Obs/GALAXIES/.

[13] Borland JBuilder. site web. http ://www.borland.com/jbuilder/.

[14] jEdit Open Source programmer's text editor. site web. http ://www.jedit.org/.

[15] kXML2. site web. http ://www.kxml.org.

[16] SavotParser. site web. http ://simbad.u-strasbg.fr/public/cdsjava.gml.

[17] Service Simbad. site web. http ://simbad.u-strasbg.fr/sim-fid.pl.

[18] Equipe Physique stellaire. site web. http ://astro.u-strasbg.fr/Obs/PHYSTEL/PHYSTEL.html.

[19] Jakarta Apache Tomcat. site web. http ://jakarta.apache.org/tomcat/.

[20] Service VizieR. site web. http ://vizier.u-strasbg.fr/viz-bin/VizieR.

[21] Format XMI. site web. http ://xml.coverpages.org/xmi.html.