Université Louis-Pasteur
Observatoire Astronomique

Thesis

to obtain the title:
Doctor of Philosophy of
the University Strasbourg I

Discipline: **Astronomy, Computer Science Applications**

defended on January 21, 2000

by

Damien Guillaume

DISTRIBUTED INFORMATION RETRIEVAL, SEARCH AND PROCESSING IN ASTRONOMY

Thesis supervisor: Prof. Fionn Murtagh

Examining board:

Prof. Daniel Egret, President
Prof. Fionn Murtagh
Dr. François Ochsenbein
Dr. Miguel Albrecht
Prof. Jerzy Korczak

ABSTRACT in English

The thesis aims at improving information retrieval and knowledge discovery in astronomy. While good solutions exist to store and process purely unstructured data, strictly structured data, or graphic multimedia objects, a better support was needed for documents mixing structured, unstructured data, and multimedia objects. Also, a standard metadata syntax was required. Based on the concept of object-oriented information, and on flexible data and software structures, the solutions tested in the course of the thesis provide a better way of exchanging data and searching it.

With XML, a new meta-language providing a uniform, flexible yet rigorous way of storing complex multimedia data and metadata, a new language for astronomical data and metadata has been created: AML, Astronomical Markup Language. AML is composed of a number of objects (currently: astronomical object, article, image, table, set of tables, person and project) that can be expressed in a unified way. AML uses a standard linking system, it is designed to handle scientific data, and it provides customisation possibilities.

A Java browser has been created to display these objects, and the many links between them can be followed to navigate through a collection. A knowledge discovery application is the *AML map*: a set of AML documents is clustered with the information residing in the links between the documents, and the available keywords. This clustering is done with an improved graph-partitioning algorithm, and the resulting synthetic classification can be browsed with a semantic map interface linked to the AML browser.

Existing distributed search frameworks (Harvest, Ingrid and Emerge) have been extended to make use of AML, and applied to collections of AML documents, thus providing a way to search through various heterogeneous datasets.

Finally, efforts were made to promote XML to the astronomical community, apply the new developments in existing data centres, and work on the adoption of new standards.

ABSTRACT in French

L'objectif de la thèse est de définir les moyens d'améliorer la recherche d'information et la découverte de connaissances en astronomie. Alors que de bonnes solutions existent pour stocker et traiter des données non structurées, des données structurées, ou des objets multimédia graphiques, un meilleur support était nécessaire pour des documents mélangeant ces trois types de données. Une syntaxe standard pour les méta-données était aussi requise. Basées sur le concept d'orienté-objet, et sur des structures de données et de software flexibles, les solutions testées pendant la thèse définissent une meilleure façon d'échanger et de rechercher les données.

Avec XML, un nouveau méta-langage fournissant une façon uniforme, flexible et rigoureuse de formater des données multimédia et des méta-données complexes, un langage pour les données et méta-données en astronomie a été créé: AML, Astronomical Markup Language. AML est composé d'objets (actuellement: objet astronomique, article, image, table, ensemble de tables, personne et projet) qui sont exprimés de façon unifiée.

Un navigateur modularisé a été écrit en Java pour afficher ces objets. Les nombreux liens entre les objets peuvent être suivis pour naviguer dans une collection. Une application en découvertes de connaissances est la *carte AML*: un ensemble de documents AML est classifié à partir des liens entre les documents et des mots-clés disponibles. Cette classification est effectuée avec un algorithme amélioré de partition de graphe, et la classification synthétique résultante peut être utilisée avec une interface de carte sémantique.

Des systèmes de recherche distribuée existants (Harvest, Ingrid et Emerge) ont été étendus pour utiliser AML, et appliqués avec succès à des collections de documents hétérogènes.

Finalement, des efforts ont été fournis pour promouvoir XML dans la communauté astronomique, pour appliquer les nouveaux développements dans des centres de données, et pour l'adoption de nouveaux standards.

TITLE in French: Recherche, Extraction et Traitement Distribués de l'Information en Astronomie

DISCIPLINE: Astronomy, Computer Science Applications

KEYWORDS: [astronomy,] XML, object-oriented, information retrieval, distributed, classification, semantic map, metadata, knowledge discovery, data mining

SPECIAL THANKS

to…

- Fionn Murtagh, from advice to organising workshops on relevant themes

- Philippe Poinçot, for numerous discussions about knowledge discovery in astronomy

- Paul Francis, for long discussions about search frameworks

- Ray Plante, Robert McGrath and Joe Futrelle, building the future digital libraries

- David Dubin, reviewer, for valuable comments to improve this report

- Daniel Egret, who inspired me with a training period under his direction to obtain uniform results from queries in the NED and Simbad databases

- François Ochsenbein and Miguel Albrecht, who were both reviewers and members of the examining board

- Jerzy Korczak, also a member of the examining board

- Alberto Accomazzi, for an active support of the ADS database

- The speculators who contributed to the exchange rate between the dollar and the mark in 1998, that generated a surplus for ADASS'98, allowing me to get a financial support to go to ADASS'99 and meet many people including researchers of the University of Illinois, where I got a job that let me finish this thesis

… and to many other people who contributed, more or less directly, to this thesis

TABLE OF CONTENTS

INTRODUCTION

The thesis tries to answer a question originating from the field of astronomy, where massive data collections which are sometimes not compatible, from many different sources, and stored in many different ways, need to be accessed in the same environment : how is it possible to perform automatic data mining with these heterogeneous collections ?

The special needs of information presentation and processing in astronomy have led to an early use of the internet. Data exist in large volumes, coming from different kinds of catalogues, from raw satellite data to processed and homogenised catalogues. The specific issues in astronomy this thesis tries to address, together with the current solutions and their limits, will be presented in the first chapter.

The present solution to this question is big centralised databases available on the web, such as Simbad (Wenger et al., 1996, CDS 1999c), NED (Schmitz et al., 1995a, IPAC 1999) and LEDA (CRAL 1999) for astronomical objects, ADC (GSFC 1999) and VizieR (CDS 1999a) for tables or ADS (Demleitner et al., 1999, ADS 1999) for astronomical articles. There are many other databases, with more or less data, and sometimes specialised on a particular subject. It's a huge work to collect the data and to store it all with the same format in one database (e.g. Ochsenbein et al., 1995). This standardisation work is useful, but the centralisation of the data in one server, and therefore its control by one institute, cannot be complete. Instead, a distributed database, with an agreement on an independent standard to represent and retrieve information, would allow various institutes to control their data and let users retrieve all the information in a uniform way.

More and more articles and astronomical images are now on-line, adding new kinds of data. The display of images is relatively new on internet and until recently there was no image processing facility on-line. The variety of astronomical data and the high density of the links between them is another reason for designing a distributed collaborative network.

Currently, a user can begin a search by using an object name or its coordinates to retrieve information about an astronomical object. To get a catalogue, the user can choose keywords and initiate a query, or browse a directory of classified catalogues. Most of the time, CGI (Common Gateway Interface) (CGI) programs are linking the HTML pages and the database. The result contains a formatted information, with HTML links to related information on the same server and related information on other servers : for instance, an object query in NED or Simbad returns links to the articles where the object has been cited, and these articles are available at the ADS server. This system is fine for users who want do get a simple information, but has limitations caused by HTML and the CGI, to automate such tasks and use complex queries.

Independently from the software, information retrieval can be tremendously improved with better data structures. The differences between data structures will be presented in chapter 2, from unstructured data such as free text, to structured data used in relational databases.

Bare data with only free text explanations are of use to humans but cannot be used by programs to automate tasks. As a consequence, metadata (data about data) is of primordial importance to do anything automatically. When data is exchanged, it is also important to have a common way of expressing metadata, to know what data is available. Three approaches are studied here : GILS, a big set of attributes oriented towards literature, the Dublin Core, providing a minimum set of attributes for all documents, and RDF (Resource Description Framework) (described in the XML section), a syntax based on XML to express any metadata.

XML (Extensible Markup Language) (W3C 1999a), a new markup meta-language defined by the W3C, is a basis for better information retrieval, and is discussed in chapter 3. XML defines a standard syntax for markup languages, as well as a standard way to define their structure and content.

XML, a simplified version of SGML optimised for the internet, provides a uniform, flexible yet rigorous way of storing complex multimedia data and metadata. Getting the data in HTML is very good for the user interface, but the data retrieved in HTML cannot be used for other work because it is mixed with interface specifications. XML, instead, allows the formatting of information separately from user interface specifications : a new language can be created to structure the information without adding user interface specifications, and a stylesheet document can be used independently to associate this new language with a graphical output. XML documents can be used both for browsing (as HTML) and for reusing information automatically, for instance to search the documents : object coordinates, for instance, are just a number without meaning if written in HTML, whereas they can be identified automatically as longitude and latitude in XML with appropriate tags. After an introduction to XML, a number of related standards are presented here : HyTime (ISO/IEC 10744:1997), and the standards defined by the W3C : XSL, XPointer, XLink, Namespaces, and RDF. An XML parser, and a Java interface for XML trees were implemented (these interfaces were later rendered obsolete by the definition of new standards at the W3C, SAX and DOM). Many ideas about the use of XML are also discussed.

Software structures, studied in chapter 4, are another basis for information retrieval, especially when the data is complex and specific procedures are required to access it. Interfaces and object-oriented programming have played a key role in this regard : an emerging architecture is composed of distributed objects, on the internet or intranet, communicating with appropriate protocols. A link with data structures can be established, for example with the association between an XML DTD (a document describing a set of elements and the structure of the data) or sub-DTD, and a Java object : XML is a possible format for serialising objects. A network of objects on the internet, such as the one linking HTML documents, can become self-organising, in the same way that some neural networks become organised.

AML, Astronomical Markup Language, a new language for astronomical data and metadata created in the course of the thesis, will be presented in chapter 5. AML is composed of a number of objects (currently : astronomical object, article, image, table, set of tables, person and project) that can be expressed in a unified way. It is designed for information retrieval and knowledge discovery, with a standard linking system based on XLink. It is also designed to handle scientific data, and provides customisation possibilities. A Java browser to display these objects, with a different Java class for each object, has been created. Each object can be browsed with a specific software module, sometimes performing very specific operations, such as displaying astronomical images. The many links between the objects can be followed, thus allowing a navigation similar to the one used on the World-Wide-Web.

A knowledge discovery application, described in chapter 6, is the AML map : a set of AML documents is clustered automatically with the information residing in the links between the documents, and the keywords available for some objects. This clustering is done with an improved graph-partitioning algorithm, and the resulting classification can be browsed with a semantic map interface linked to the AML browser. With the data unification provided by AML, this lets users cluster various heterogeneous objects in a unique synthetic classification. This method of classification is compared with the Kohonen map method, and an evaluation has begun with people using Kohonen maps and another method used within the Tétralogie system.

As a number of search systems already exist to perform complex distributed searches, it wasn't needed to design a completely new framework. Instead, existing frameworks (Harvest, Ingrid and Emerge) have been extended to make use of AML, as described in chapter 7. Harvest is divided into 3 software modules : the client, with the user interface, the gatherer, gathering the indexed information on each provider site, and the broker, searching through the indexed data to execute client queries. Ingrid uses a similar framework, but in a more distributed way : links are created between the resources, so that a search can focus on a sub-graph of the grid, without needing to use the metadata for all the resources. Emerge is based on the Z39.50 protocol, to allow complex and interactive queries to be executed on the different servers. Each extended framework is applied to collections of AML documents, thus providing a way to search through various heterogeneous datasets.

Finally, efforts are made to promote XML to the astronomical community, and apply the new developments in existing data centres. In particular, AML is becoming the new internal format for metadata in the ADIL (Astronomy Digital Image Library). This is detailed in the last chapter.

I. OVERVIEW OF IR ISSUES IN ASTRONOMY

IR Problems in Astronomy

A number of problems in astronomy make information retrieval difficult among different archives. These problems have to be solved or avoided in order to do some knowledge discovery. An example would be to use various catalogues maintained at different institutes to do statistics with all the available information. In practice, astronomers usually do not use all the information available, simply because it would require too much work to access, convert and homogenise the data, even though it could slightly improve the results of a study.

Volume of data

The volume of the data in astronomy is already very important, and is still growing very fast (e.g. Louys et al., 1999). This forces astronomers to use the best new techniques for data storage and manipulation, but it is also a constraint on the software used, and can ultimately affect how the data is retrieved. For instance, it is better to select the useful data before retrieving it, or to process it where it is stored if the process requires all the data but is not very CPU-consuming. This issue's importance is attested, for example, by the DIMACS workshop “Massive Datasets in Astronomy and Astrophysics”, held at Princeton University in May 1998. Metadata can be used in order to facilitate the manipulation of these big datasets.

Important peculiarities of the data

Any data has its peculiarities, like the systematic errors and the uncertainties for intensities, or even more specific problems or features, like the early default of Hubble telescope. It is difficult to generalise them to store data from different sources together, but a rigourous analysis must take these peculiarities into account. Only a very flexible file format could handle all the different data in a uniform way. For databases, this means that object-oriented database management systems (OODBMS) are often better suited than relational database management systems (RDBMS) (e.g. Brunner et al., 1995).

Individual projects data is usually not available

Traditionally, teams in astronomy work independantly, and present their results in papers and conferences. They do not publish the reduced data itself in a reusable way : for instance, even if a table is published in a paper, it has to be retyped to be used electronically (with possible errors), and the other tables that led to the published one after some processing are not directly available. As most of the online published data comes from large surveys, leading-edge information resulting from individual projects is, in practice, unavailable. Their publication could allow a deeper analysis (Kurtz 1995).

No universal identifier

Astronomers often refer to the objects they observe by their names, but there are many names associated to a single object, which does not simplify the task of researchers. Also, full bibliographic references in articles are not always spelled exactly in the same way, so they cannot be used as identifiers. To solve this problem, a unique identifier for astronomical articles was created between NED, ADS and the CDS, and was used in the web links : the “bibcode” (Schmitz et al., 1995b). This 19-digits code, although not solving all the problems of identifying bibliographical information, was the source of many projects requiring a unique identifier.

Each time a new catalogue is created, there can be a new name for an astronomical object (with an abbreviation for the catalogue and a number), so object names lists have to be maintained dynamically, but this task is complex : associating object names requires cross-correlations procedures between the catalogues. For instance, in December 1999, Simbad returns 20 names for M31, and NED returns 24 names. NED and Simbad also provide an object name resolution service, with a client-server interface, to get the coordinates and the other object names given an objet name (as well as a few other functions). Because of the complexity and the usefulness of this task, this service is often used today by many online softwares.

Different archives with different formats/software

Online archives, often gathering data from many catalogues, always have different user interfaces with different features. They usually focus on a certain kind of data (like in a given bandpass), which explains the differences in the interfaces, but some projects require the use of data coming from different archives : the data has to be retrieved independantly at the different web sites, and converted when they are not using the same format. The experience of a user trying to find his path in the “jungle” of astronomical on-line data services is described in (Egret 1994).

Many initiatives have tried to solve this problem :

FITS (FITS)

FITS is a standard format for images and tables. It was a starting point for data exchange, and provided useful customisation possibilities. Though very used, this format is not perfect : it is limited to particular kinds of data (images and N-D tables), and is not very flexible (it was designed for punch cards, with a header having 80 characters lines without newline character, and a keyword starting each line). Also, it was designed by and for the astronomical community, and is not used in other disciplines.

Web interfaces

The web was a revolution for data exchange in astronomy : it became easy, for a research group, to create a simple user interface leading to data files, or for an archive center to provide a universal interface without requiring an installation of software on the client side.

The user interface has improved a lot, changing in ten years from text-only queries and replies to graphical interfaces with interactive sky maps. An example of graphical user interface improving IR is the Kohonen maps for some astronomical journals : they present a semantic map of the articles published in one journal, with the articles clustered together when they have similar keywords.

Web links

After the initial boom of new web sites, web links began to appear between archive centers, helping the users to move from one site to another in order to search for related information. For instance, an astronomical object on one server will be linked to the articles related to it, and to other servers providing other kinds of information about this object (e.g. Lesteven et al., 1998).

AstroWeb (Jackson et al., 1995)

AstroWeb is a directory of resources in astronomy, specified by their URL. In particular, it provides links to most of the observatories home pages in a convenient way.

AstroBrowse (Heikkila, McGlynn & White, 1999)

AstroBrowse is a project to access different resources over the internet, starting with a single query. For instance, a user would provide an object name, and the system would return a list of URLs possibly containing information about it. The user can then follow the links instead of rewriting the query with each interface. There are a few implementations for this project, based on the use of GLU to maintain the URLs.

Information search and retrieval systems

How is the information found now ? The most used way of finding new information today is perhaps the use of hypertext links : this kind of search is sometimes hazardous, but it is easy to direct a search and to move very deep into the tree of hypertext documents by refining the search. “Surfing” has even became a popular fashion.

As IR is a matter of communication, the automatic procedures often take inspiration from human-to-human communication, and digitally-enhanced communication could be considered as a part of the global IR. In particular, email and news are very used today, and are often an efficient way of finding information. Documents such as FAQ (Frequently Asked Questions) are very useful, because they can answer your questions even before they are asked, in a human way of questioning/answering. Based on human dialogues, databases can be created and searched in various ways. Mailing lists and automatic information alerts are other examples of IR based on human dialogs, and they tend to be more and more "intelligent" with the use of information filtering techniques.

Also very used on internet are the search engines (Altavista, Excite, HotBot…) and directories (Yahoo, LookSmart, …) (most of these sites now provide the two services). While directories provide high-quality sites hierarchically organised where a search is conducted by categories, search engines provide an access to huge URLs databases searchable by keywords (either with a boolean or a statistical method).

ADS and ESIS

ADS (Astrophysics Data System) was originally a more ambitious project (Eichhorn 1994), similar to the ESIS (European Space Information Project) project (Ansari et al., 1993, Giommi & Ansari, 1995) in Europe. Both projects were supposed to create a unifying architecture for space science data, but they failed to reach this goal and were eventually reduced to a few of their components (like VizieR for ESIS, or the bibliographic service for ADS). It is difficult to know today why these projects have not reached their goal, because, unfortunately, most of the published literature is about success stories. However, from informal discussions, it seems that the technical problems were not the most important ones. When the various components of ESIS were transferred to different institutes, a multi-client/multi-server architecture was planned (Ansari, Donzelli, Simonsen & Ochsenbein, 1996). According to an announcement on the ADS server, the restructuring of the ADS was necessitated by continued budgetary pressures within NASA (ADS 1994). Another problem for these two projects was the emergence of the WWW, when most of the software had just been programmed. Also, as stated in (Egret & Albrecht, 1995), "A key lesson from these experiences is that the data have to reside and be maintained at the same place where the expertise is located".

GLU (Fernique, Ochsenbein & Wenger, 1998, CDS 1999b)

GLU is a system able to represent a URL with a name and parameters, and to distribute automatically the information of the link between the name and the URL among different servers. It is often used today in astronomy to access CGI programs using a name instead of a URL.

Limited access to the metadata and the data (as opposed to a visualisation of it)

Some archives provide search and selection capabilities, or simple processing of the data. However, the generated data itself is usually not avaible. For instance, a query in VizieR results in an HTML document, allowing an easy visualisation of the data, but their is no specification of the HTML document syntax that would allow a parsing program to retrieve the data automatically, so that it can be reused in another software : a user has to write the result by hand to use it later. If the result is too big, the user has to download whole catalogs or unprocessed data and execute the selection or processing again, locally (which requires some specialised software to be installed locally).

Limited scientific queries

Most of the search systems for scientific data on the web do not have any typically scientific feature. This is because most of the freely available software in IR has not been conceived to handle scientific queries, and also because of the lack of standardisation for scientific features. An example of a "scientific query" in astronomy could be "select the redshift measurements for MRK334 having uncertainties < 5%" (using "velocity" instead of "redshift" would give the same result, even though some archives would store the values under the name "redshift" and others under the name "velocity"). There is no "standard query language for scientific data", and different sub-disciplines are using different "context metadata" in the information they publish (i.e. metadata residing in the context of the sub-discipline but not defined explicitly). There has been some experiments to try to solve this problem (Pasian & Smareylia, 1998).

Limits of the current solutions

The current solutions, though improving information retrieval, still do not allow knowledge discovery among different archives. This is mainly due to the current techniques of the web (HTML and the CGI), and to the lack of standards for specific scientific aspects of the data.

HTML

Whenever data is returned in HTML, the main format for the web, it cannot be reused easily. Because HTML surrounds the data with HTML elements used for the user interface, one has to separated the data from the user interface information in the HTML source in order to reuse the data. Depending on the user interface, this can be more or less difficult, but since the user interface is always subject to change without notification, it is a risky process. Because the data is returned in HTML within the AstroBrowse system, it is not possible to know automatically if there was some data matching the query, and a link to a possible result returned by AstroBrowse can very well point to an HTML page with the mention "no match found".

CGI

CGI programs are often used on a server to get the input from an HTML form, and create a resulting HTML document. The input can be specified either as attribute/value pairs (the GET method), or as a whole file (the POST method). The result is a document identified by its MIME type (the first line of the data returned by a CGI program identifies the type of the file) (MIME, Multipurpose Internet Mail, is defined by the IETF (Internet Engineering Task Force) by RFC 1521, RFC 1522, RFC 1590 and RFC 2048). There are several limits to the technique :

- input : There is a standard way to define attribute/value pairs, there is no standard syntax to define more complex types as an input to the CGI program. Also, there is no standard syntax to explain what the attributes mean. To be used properly, a documentation should be written for each CGI script to explain what is the space of correct inputs. Of course, there is no standard syntax for the documentation. ASU (Astronomical Server URL) (CDS 1999d) is a standard for attribute names used for CGI programs in astronomy : the set of attributes is small (and only a few CGI programs conform to it), but such a standard solves most of this problem.

- output : the ouput of a CGI program is a document. Again, there is no standard syntax for the result or to specify its meaning. The fact that the output is a document is fine with HTML documents, but is not for more complex user interfaces. Usually, when the interface cannot be specified with HTML forms, a Java program is used instead. Changing from a CGI program to a Java program is not funny, especially if HTML documents were used with the CGI : everything has to be written again.

- no interaction : the protocol is stateless, which means that a CGI program cannot maintain a session with another program. For instance, when a password is required to execute a number of different queries, and if only a query can be executed at a time (or if it's required by the calling program), the password has to be sent as a parameter for each query.

Lack of standards

Many standards could be defined to easy information retrieval in astronomy. For example, there could be a standard "preferred name" for all the well identified objects. The classifications for the object types could also be standardised to be used for searching objects. A standard way to use units, or to write accepted ranges of errors for a value in a query, could be defined. This work is long and difficult, and can only be validated if there is a political wave in the direction of standardisation. With the increasing use of computer science, and the increasing exchange of information between researchers, we can hope that the needs for standardisations will result in the necessary political decisions.

Centralisation

Centralisation of the information solves many problems, from the unification of the data to the user interfaces. This solution is also ideal for institutes not willing to maintain a server : they can send their data to a data center, which will publish it electronically. However, there are also issues linked to this strategy : data providers are not experts for the data they publish, and hence do not always know how to best publish the information. Because of the volume of the data, data centers cannot convert all the various metadata they get, so the authors still have to convert them into the data centers formats. The volume of the data does not seem to be an issue linked to centralisation currently, because many data storage devices can be used at the same place in a transparent way, and the cost by volume ratio is decreasing with the volume of the data (although this becomes less true as the limits of technology are reached). But there is a limit in the diversity of the data : today, a single database cannot contain all the world information, because it is best stored and processed in different ways by different people. This means that, even if centralised databases are preferred, solutions have to be found to make them communicate, in a way similar to what would have been used with smaller databases. Data ownership considerations are another problem; Van Alstyne and co-authors suggest this is the most important one to choose between a centralised or a decentralised architecture (Van Alstyne, Brynjolfsson & Madnick 1995).

II. DATA STRUCTURES

Improving the data structures can be very helpful to manage the data and search through them. Creating a clear set of metadata also makes exchanges with external entities easier : with good internal data structures, adding a record from an external source or publishing data becomes a matter of independently selecting data items and changing the formatting.

“Unstructured” data (free text, formatted text)

What is called unstructured data here is data with no internal structure, regardless of the user interface information. For instance, a free text, or a formatted text. There is a structure in it (chapters, paragraphs, phrases, words), but this structure is not explicit in the data. The goal of such a file is to present linear information, sometimes multimedia.

Examples of such files are HTML, RTF or Postscript documents. These files, because they only intend to display information, do not have a support for structured data. But they still are the best formats to present multimedia information.

These files are usually text-based, with keywords for formatting information, so that it is possible to edit them in a text editor, and they can be easily exported.

Free text systems can be used to index and search for information, with the use of keywords. The problem is that some terms are ambiguous, and it is the cause of a bad precision in the search results.

Because free texts and formatted texts do not have their own structure, they cannot be used for complex and precise processing.

Structured data (field-based)

Structured data is most of the time stored in databases. In this case, a rigid structure has been chosen to store the information in an efficient way for both data storage and processing. This requires modelling work, and some constraints to be established, but enables further complex processing on the data.

In databases, the new data must adapt to the structure, which is closely related to the way the information is stored. Also, because of this close relation, the structure must comply with certain rules.

Big formatted texts and other media objects are not handled in traditional databases, because it is rather complex to implement without loosing speed.

Because they are composed of many binary files, always in different formats, databases are not easy to export and convert. Big ASCII files are sometimes used, but they are not an ideal solution.

Heterogeneous data or data with a complex structure

More and more, structured data with media objects (audio, video, images, formatted and unformatted text) is required, to enable complex processing on the data and presentations based on it. We want a system having the good properties of the two previous solutions (hypertext and databases), but with no drawback. As hardware-related questions are getting more separated from the concepts we use, thanks to the fast progress of the hardware, it is possible to forget this problem in answering our question.

Different solutions are currently being proposed, but none can solve all of the problems in itself, at the same time.

ORDBMS (Object-Relational Database Management Systems, also called Extended Relational), are extending the basic relational database model, adding media objects and encapsulating methods with data structures. They try to imitate the object-oriented databases, but keeping a compatibility with existing relational databases, so the result is not as good as with real OO databases (the encapsulation is not as good, inheritance is not completely supported). Also, media objects and methods are simply saved in BLOB (binary large object) fields, which does not allow intern structure to these objects. These databases, still being developed, are even more difficult to export than relational databases, because of the media objects.

OODBMS (Object-Oriented Database Management Systems) still do not have a good support for formatted texts : a free text cannot be given a structure which would be mixed with the database structure. Also, usually, an HTML file is created directly from the database to present the information, and these files cannot be used by the users to get structured information. Object oriented databases are not efficient for variably nested hierarchies and sparsely populated trees (which describes most narrative text). Finally, they are not easier than the previous databases in allowing for export of their contents.

SGML and the new XML language (see the next chapter for a presentation) are a link between unstructured data with media objects and structured data : both can be represented in SGML. SGML documents are text files, so they can be easily written and formatted, with or without dedicated software. They are easy to export, and XML is using UNICODE so XML files are truly international.

An SGML document can have links to media objects (as in HTML, a specific application of SGML), and they can also have specific links with other documents (while in a database all records in a file must have the same fields). Style sheet files can be used to give different user interfaces to an SGML (like CSS or XSL for XML) document, so it is an ideal way to present the information.

However, it does not answer the question of storing efficiently structured information to enable efficient processing.

A unified solution

To summarise, there are now databases handling media objects, and formats to use for presenting structured and unstructured information and for exporting. A unified solution is to use OODBMS or ORDBMS for structured information, and to produce XML files with them. XML editors can be used to produce formatted texts separately from the database, and all these XML documents can be displayed together, in different ways, with links between structured and unstructured data.

This solution adds a layer in the current process : currently, HTML is often produced directly from the databases, and no style sheet is needed. HTML documents are also produced separately, and the text is not separated from the interface for its display. But, with the automation and the use of XML-enabled software, it could become transparent for the user.

Whether the data is structured or not, it is often associated with metadata. The metadata, playing a fundamental role in information retrieval, can also be more or less structured, and can be stored with or independently from the associated data.

Metadata

The usual definition for metadata is “data about data”. It is also referred to as information adding a meaning to raw data. But, the distinction between “data” and “metadata” is not an absolute one. In fact, a whole “chain of abstractions” can be created from the data, explaining what is the data, what is the information about this data, and so on.

Not so long ago, many programs used to process files without writing into these files what was the information they carried. A number would be known by the program to have a given meaning, and if other applications wanted to use the same file they would have to know it’s format. However, the number of applications increasing, with various use of same data, and the formats getting more and more complex, the information without metadata often became effectively unavailable.

Metadata is now known as being an important instrument to transform raw data into knowledge, and more attention is paid to it. It is the only way for “intelligent agents” to retrieve and process information on the Internet efficiently. With the help of metadata, the web will elevate its status from *machine-readable* to *machine-understandable*.

GILS

The U.S. Government Information Locator Service (GILS) (GILS 1999), also called Global Information Locator outside the U.S., is a decentralised network of government systems known as locators. A locator provides users with metadata including descriptive, location and access information to public information available from U.S. federal government departments and agencies. GILS was developed to support U.S. government policies that require government departments and agencies to make information accessible to the public and to develop information systems that can be used in an open systems environment. The implementation of GILS results in a distributed collection of information systems that can easily be searched to find agency-based information of interest.

A GILS locator system includes a searchable database of one or more metadata records. These records indicate : (1) what information is available for a specific topic; (2) where the information is located; and (3) how the user can access the information. The content of the locator record is not the information itself, but a description of it. A record can describe a service, a collection, or an individual item regardless of its format or medium. For example, a record may be prepared for an online catalogue, inventory, gopher site, government manual, directory, or electronic serial.

GILS implementation is using the Z39.50 Information Retrieval standard with a specific set of attributes to enable the searching of GILS systems.

Dublin Core

The “Dublin Core” (Dublin Core Directorate 1999) is a simple set of metadata aimed at providing a short unified subset of metadata for any document. It is an answer for the need of a common metadata basis for cross disciplinary semantics.

More metadata is useful in a given discipline, but it requires an expertise to create it. The Dublin Core can be applied to any document with a minimal effort, and this is important for most documents because most documents do not need to contain a lot of metadata.

The following criteria were advanced as desirable features in whatever syntax is to be used in defining a standard format for metadata :

- ability to express optional, repeatable, sequences
- verifiability, robustness
- compactness
- uniformity
- ability to express of qualifiers
- repeatability of groups
- machine processability and human comprehensibility
- internationalisability
- simplicity
- formality

The following functional requirements were identified for the syntax :

- it should be simple enough for authors to create their own metadata (either directly or via a

suitable interface)
- it must be able to represent at least the elements of the Dublin core
- it must support extension by users, with varying degrees of formality and control, so that users can extend the Dublin core
- it must be able to express attribute-value-scheme triplets (since the metadata elements are all expressed as such)
- it must be able to express optionality and repeatability

The Dublin Core Workshops

Employing a multidiscipline approach and focus group methodology to develop consensus, each of the workshops has contributed to the refinement of the arguments and redirection of the research as described below.

The first Dublin Core Metadata Workshop was held in March, 1995. The first workshop's product was a core set of 13 metadata elements (the Dublin Core Metadata Element Set) that could be used to describe networked textual resources.

A year later, in April 1996, a second workshop was convened at Warwick University in England. The products of the Warwick Metadata Workshop included a proposed Dublin Core syntax, guidelines, and a modular based framework (the Warwick Framework) for packaging the metadata.

In September, 1996, the third workshop was the CNI/OCLC Image Metadata Workshop, in Dublin, Ohio. The products of DC-3 included the expansion of the Dublin Core Element Set to 15 elements, and the refinement of the element set to support the description of networked non-textual resources.

Canberra, Australia, hosted the 4th Dublin Core Metadata Workshop in March, 1997. The products of DC-4 included formalised qualifiers (the Canberra Qualifiers) and the development of syntactical expressions related to HTML.

In October, 1997, the 5th Dublin Core Metadata Workshop met in Helsinki, Finland. The relations between the Dublin Core and other standards were discussed, especially HTML4, RDF and Z39.50. To solve the problem of the degree of abstraction described with the Dublin Core (should the metadata relate to an object, an image of it, or the file containing this image ?), sets of metadata (the 1:1 approach) have been considered.

The 6th workshop was held in Washington D.C. (USA), in November 1998. There was more discussion on RDF, Z39.50, multilinguality and the possible evolutions of the Dublin Core. Because of the growing number of implementations, implementation experiences were shared in order to promote interoperability between them.

III. XML

XML is a meta-language : it gives the possibility to create new formats, with a wide range of possible structures but a strict syntax, and describe these new formats explicitly. Therefore, XML is excellent for the manipulation and exchange of data, and is a good basis for knowledge discovery. Because choosing XML was such a strategic decision for this study, it is described here in details.

The limits of HTML

The widely used HTML language has a structural markup : it is composed of nested tags (with the syntax <tag>...</tag>). Although HTML was very successful for displaying information on the web, it has a number of limitations.

It has little internal structure : it's possible to write valid HTML that does not make sense, e.g. an H1 tag inside an H2 one.

The HTML markup has a meaning for a trained reader (able to tell what a word means in italic, for instance), but there are no semantics in HTML other than display information, which means that programs cannot understand the semantics. With the development of HTML, more tags have been added to solve display problems, giving less importance to the semantics, and in the end HTML is, for a developer, only considered as another format to display text and graphics.

For instance, the TITLE and ADDRESS tags, or STRONG as opposed to B, have a semantic meaning. But many other tags like this would be required to deal with all the applications of a format for the web. Not only have these semantics been forgotten by the browser vendors, but they have also added their own tags, incompatible with other browsers. This was their way of providing more possibilities for the many applications of HTML, but the result has been HTML pages only available correctly for those having a given browser, platform and screen resolution, thus breaking with the initial aims of HTML.

GML and SGML

HTML is actually an application of SGML, a meta-language (a language to describe a language) which was the result of a long search to exchange and manipulate various types of documents in an open standard. These efforts date back to the 60s, with GenCode, from the GCA (Graphic Communications Association) and GML (Generalised Markup Language) from IBM.

GML was already using tags, but the rules were not very strict : it was more designed to be easy to read and write for humans than for automatic processing. As a consequence, different compilers had to be written to read different kinds of data.

This led to the definition of SGML (Standard Generalised Markup Language), able to describe various markups in DTD (Document Type Definition) files, and to write documents with markups following the rules of their DTDs. SGML became an ISO standard in 1986. It is portable, formal enough to allow proofs of document validity, structured enough to handle complex documents, and extensible enough to support the management of large information repositories.

However, SGML is difficult to implement completely, because its specification is very long and difficult to understand. Often, programs are able to process documents for a specific DTD, with no compatibility with SGML as a whole. The web browsers, for instance, were only able to read HTML, which is a specific DTD, and the HTML documents are not even validated against the HTML DTD by the browsers. Increasingly, a new standard was needed, with SGML capabilities, and as adapted to the internet as HTML.

The dawn of XML

The W3C (World Wide Web Consortium) started to work on this new language in 1996, and XML reached the stage of "W3C Final Recommendation" in February, 1998. XML, like SGML and unlike HTML, is a meta-language for describing markup languages, as well as a precise syntax for these markup languages. It is actually a small subset of SGML, keeping the most important features (extensibility, structure and validation), but simplifying it so that it becomes easy to implement an XML parser, and to allow its use over the web. XML specification is only 26 pages long, while SGML specification had 500 pages, and it is expressed in EBNF (Extended Backus-Naur Form) while this was impossible with SGML (EBNF is a syntax to describe a language grammar unambiguously).

Even though XML is much better than SGML for web use, it does not aim at replacing it : the complex features available in SGML can be useful for large, complex information repositories, and are used a lot for publishing.

It does not aim either at replacing HTML : HTML will still be useful to create a simple presentation for the web, when the presentation is the only goal. However, HTML will probably be transformed into an XML DTD, and become one of the many applications of XML.

Goals

The W3C Recommendation (W3C 1999a) includes the following goals that prevailed for the definition of XML :

- XML shall be straightforwardly usable over the Internet.
- XML shall support a wide variety of applications.
- XML shall be compatible with SGML.
- It shall be easy to write programs which process XML documents.
- The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
- XML documents should be human-legible and reasonably clear.
- The XML design should be prepared quickly.
- The design of XML shall be formal and concise.
- XML documents shall be easy to create.
- Terseness in XML markup is of minimal importance.

The last point is the most interesting in my opinion, because it's a feature voluntarily limited in XML, to allow or improve other features. The idea behind it is that specific editors (or editors having shortcuts to create XML documents) will be used to create XML documents, or that they will be created automatically. It is still possible to create XML documents completely "by hand", but this should not, like in SGML, make XML specification and implementation complex.

Technical introduction

Example

Here is an XML document example :

```
<?xml version="1.0"?>
<LYRICS>
<SONG title="The Drowning Man" author="The Cure">
<VERSE>
<LINE>She stands twelve feet above the flood</LINE>
<LINE>She stares</LINE>
<LINE>Alone</LINE>
<LINE>Across the water</LINE>
</VERSE>
<!-- more verses... -->
</SONG>
</LYRICS>
```

A document is called an “XML document” if it is *well-formed*, which means it follows the syntactic rules for XML. It is *valid* if it meets further constraints. The example above is only well-formed, because the DTD is not declared.

Physically, an XML document is composed of units called *entities*, which can call other entities so that they are also included in the document. Logically, the document is composed explicitly of :

- document type declaration
  e.g. `<!DOCTYPE lyrics SYSTEM "lyrics.dtd">`
- elements (possibly with attributes)
  e.g. `<LINE>Alone</LINE>` or `<PAUSE/>` (the last one is an *empty* element)
- comments
  e.g. `<!-- for instance, <unprocessed-element/> -->`
- entity references or character references
  e.g. `&copyright;` or `{`
- processing instructions
  e.g. `<?name pidata?>`
- CDATA sections
  e.g. `<![CDATA[ *p=&q;  b=(i<=3); ]]>`

Document type declaration

To be valid, an XML document must have a document type declaration and the document must obey the constraints on the sequence and nesting of tags of the DTD. There are four kinds of declarations :

- elements
- attribute lists
- entities
- notations

For example, the DTD for the previous XML document example is :

```
<!ELEMENT LYRICS (SONG+)>
<!ELEMENT SONG   (VERSE+)>
<ATTLIST SONG
      title       CDATA       #REQUIRED
      author      CDATA       #REQUIRED>
<!ELEMENT VERSE  (LINE+)>
<!ELEMENT LINE   (#PCDATA)>
```

Elements

The *content model* is defined for each element. A more complex example of a content model is :

```
<!ELEMENT EXAMPLE (A, B?, C*, D+, E|F)>
```

which means that an EXAMPLE element must contain an A element, followed optionally by B, followed by zero or more C, followed by one or more D, followed by E or F.
The element content can be defined by the reserved words :

`#PCDATA`	parsable character data
`EMPTY`	empty element
`ANY`	any content (should be avoided because it disables content checking in that element)

Attributes

The attributes can be of the following types :

- CDATA
  any text (not to be confused with the CDATA *section* : this text is parsed)
- ID
  a name to identify the element; all ID values in a document must be different
- IDREF or IDREFS
  an IDREF(S) is a reference to ID(s) in the document
- ENTITY or ENTITIES

the name of one or more entities

- NMTOKEN or NMTOKENS

  one or more words
- a list of names

  the attribute's value must be one of the indicated names

There are four predefined notation names for the attributes :

- #REQUIRED

  the attribute must be used each time the corresponding element is used
- #IMPLIED

  the attribute is optional
- "value"

  the attribute is optional, and has the default value specified
- #FIXED "value"

  the attribute is still optional, but must have the specified value if it is used

Entities

The entities can be either internal, external, or parameter entities. Internal entities contain a block of text. External entities are references to files that may contain textual or binary data. If an entity's value is text, the text is parsed when the entity is parsed, possibly replacing other entities by their values.

There are some predefined entities which do not need to be declared, for special characters (they come from HTML) : "<", ">", "&", "'" and """.

Notation

Notation can be used to identify specific types of external binary data, e.g.

```
<!NOTATION GIF87A SYSTEM "GIF">
```

DTD specification

A valid XML document must include a DTD specification. For the above example, this can be written as follows :

```
<?xml version="1.0" standalone='no'?>
<!DOCTYPE LYRICS SYSTEM "lyrics.dtd">
<LYRICS>
...
```

This tells a validating parser that the DTD file can be found at the same location as the XML document, under the name "lyrics.dtd". Specific markup definitions for the document can also be added after the DTD filename.

Related standards : HyTime, XSL, XPointer, XLink, Namespaces, RDF

HyTime

HyTime (Hypermedia/Time-based Structuring Language) (ISO/IEC 1997) extends SGML by defining a standard set of facilities for doing hypertext and multimedia presentations. It is an SGML application conforming to ISO8879. HyTime is not directly linked to XML, but it had a strong influence on the definition of XML and its extensions.

HyTime provides the notion of “Architectural Form” to SGML. An architectural form is a syntax template around which a document author can build semantic constructs for linking and coordinate space addressing.

Like SGML, HyTime addresses only the issue of interchange of hypermedia information, as opposed to the standardisation of presentation. Two specific issues are well supported by HyTime : the representation of hypertext links, and the synchronisation of static and time-based information.

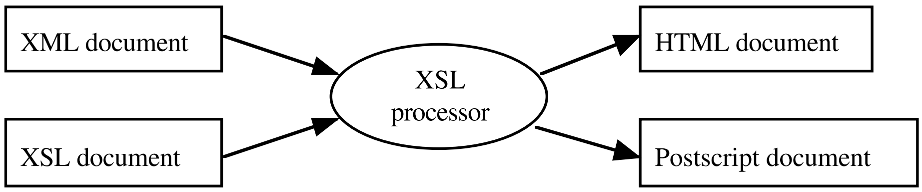
HyTime supports cross-referencing facilities to uniquely identified elements in external documents. It also extends SGML’s reference capability to accommodate elements with no unique identifiers in the same document. It provides pointers or location addressing schemes that contain the necessary information in order to locate cross-referenced data. HyTime supports addressing by name, by position in the document, and by semantic construct. Links can be established to documents that conform to HyTime as well as those that do not.

XSL, a layout for XML

XML documents are content-oriented, which means that there is no interface specification. A mechanism is thus required to display XML documents, and this is possible with stylesheet languages. These stylesheets are separate documents describing how an XML document can be transformed into a rendered format (this can be graphical or any other media support, like speech).

XSL (W3C 1999j) is a stylesheet language designed for XML. It is derived from CSS (Cascading Stylesheets), a simple stylesheet language focusing on visual characteristics, and DSSSL (Document Style and Semantics Specification Language), the standard stylesheet language for SGML, allowing much more complex processing although not well suited for the web.

An *XSL processor* is used to transform an XML document into a rendered format, with the use of the XSL document for the DTD used by the XML document.



Example with the current XSL working draft (specification of XSL may still change) :

```
XML :          <ChapterTitle>My Chapter</ChapterTitle>

XSL :          <xsl:template match="book/ChapterTitle">
                 <H1>
                 <xsl:apply-templates/>
                 </H1>
              </xsl:template>

HTML result : <H1>My Chapter</H1>
```

XPointer

XPointer is a language used to refer to a part of an XML document, using its internal structure. Its definition has been influenced by HTML, HyTime, and the Text Encoding Initiative (a project having the task to develop and disseminate guidelines for the interchange of machine-readable texts among researchers, and to make recommendations for the encoding of new texts).

An XPointer is composed of a list of *location terms*, each specifying a location in the XML structure relatively to the previous one. A location term is composed of a keyword (such as id, child, ancestor) and possibly of parameters related to the keyword. For instance, `child(2,CHAP)` refers to the second child element whose type is `CHAP`.

An XPointer can be associated to a URI (Uniform Resource Identifier) identifying the XML document.

XPointer is still a W3C working draft, and is subject to change. It is not described further here because it hasn't been used in this study, but more information can be found on the W3C web site (W3C 1999b).

XLink, XML linking language

XLink (W3C 1999c) defines a syntax to describe explicitly relationships between XML documents or parts of them. Its definition has also been influenced by HTML, HyTime, and the Text Encoding Initiative.

An XML link is defined by an element in the DTD : its syntax is defined by a DTD, but it must conform to some syntactic rules, like attribute names, defined by the XLink standard. The simplest case is like an HTML A link, with these limitations :

- the link is one-way, and is declared only in the origin document
- the link behaviour with the user interface (e.g. when a user clicks on the link in a browser) is not defined by the link itself, it is chosen by the browser
- the link is only linking one document to another

XLink goes beyond all of these limitations, and also adds semantics to the links. Note that the possible effect on the user interface is an exception in XML, normally separating the content from any user interface.

Example of a *simple* link :

```
<mylink xml:link="simple" title="Citation" href="http://www.xyz.com/xml/foo.xml"
show="new" content-role="reference">as discussed in Smith(1997)</mylink>
```

Declaration :

```
<!ELEMENT mylink (#PCDATA)>
<!ATTLIST mylink
      xml:link          CDATA                   #FIXED "simple"
      href              CDATA                   #REQUIRED
      content-role      CDATA                   #IMPLIED
      title             CDATA                   #IMPLIED
      show              (embed|replace|new)     #IMPLIED >
```

Example of an *extended* link (linking more than two documents together) :

```
<commentary xml:link="extended" inline="false">
  <locator href="smith2.1" role="Essay"/>
  <locator href="jones1.4" role="Refutal"/>
  <locator href="robin3.2" role="Comparison"/>
</commentary>
```

Namespaces

The Namespaces specification, which became a W3C recommendation in January 1999, is a syntax to use different DTDs within the same XML document. It extends the element names from a local name specified in one DTD, to global names adding a DTD URI to the element names.

Example :

```
<?xml version="1.0"?>
  <!-- initially, the default namespace is "books" -->
  <book xmlns='urn:loc.gov:books'
        xmlns:isbn='urn:ISBN:0-395-36341-6'>
      <title>Cheaper by the Dozen</title>
      <isbn:number>1568491379</isbn:number>
      <notes>
        <!-- make HTML the default namespace for some commentary -->
        <p xmlns='urn:w3-org-ns:HTML'>
            This is a <i>funny</i> book!
        </p>
      </notes>
  </book>
```

In this example, three DTDs are used in the same XML document : “books”, “isbn” and “HTML”. The elements “book”, “title” and “notes” are declared in the “books” DTD, “number” is an element declared in the “isbn” DTD, and “p” and “i” are declared in the “HTML” DTD.

The usefulness of such a syntax is evident : this is a standard way to combine DTDs, and still allows DTD validation. The only drawback is the additional complexity to parse XML documents.

However, this can only be used with compatible DTDs (a structure from a DTD can be enclosed in an element of the other DTD), and preferably with DTDs having no overlap in their element semantics (otherwise the resulting document can have ambiguous semantics). Namespaces are not very compatible with XML 1.0 and the DTD syntax, because their is no standard way to validate a document using more than one DTD. The future "XML schema" should solve this problem, and a temporary solution is to create a new DTD combining the DTDs used with namespaces.

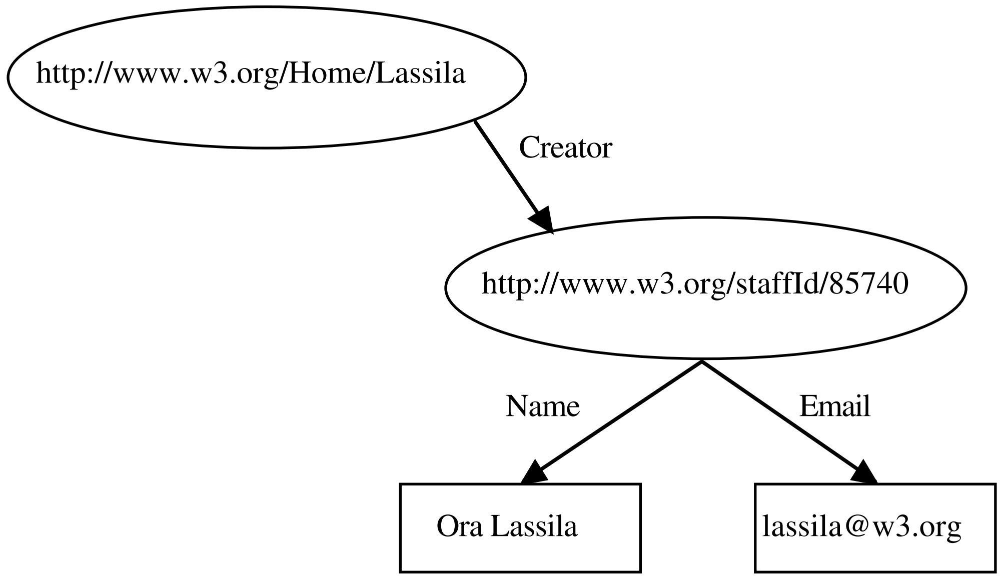
RDF

RDF (W3C 1999d) is a model for representing metadata, and one possible syntax (using XML) for expressing it. At the core, RDF consists of *nodes*, and attached *attribute/value pairs*. Nodes can be any web resources or other instances of metadata. Attributes are named properties of the nodes, and their values are either atomic or other resources or metadata instances. The instances of RDF data models, the RDF Schemas, are entity-relationship diagrams, and two RDF expressions are equivalent if and only if their data model representations are the same.

RDF in itself does not contain any predefined vocabularies for authoring metadata, but its point of view is broad enough to be applicable to most (if not all) metadata information.

Example :

The sentences "The individual referred to by employee id 85740 is named Ora Lassila and has the email address `lassila@w3.org`. The resource `http://www.w3.org/Home/Lassila` was created by this individual." result in the RDF model :



and a possible syntax for it is :

```
<rdf:RDF>
    <rdf:Description about="http://www.w3.org/Home/Lassila">
      <s:Creator rdf:resource="http://www.w3.org/staffId/85740"/>
    </rdf:Description>

    <rdf:Description about="http://www.w3.org/staffId/85740">
      <v:Name>Ora Lassila</v:Name>
      <v:Email>lassila@w3.org</v:Email>
    </rdf:Description>
  </rdf:RDF>
```

Here is an example of the Dublin Core used with the proposed RDF syntax. The following Dublin Core properties are used : “Title”, “Description”, “Publisher”, ”Date”, “Subject”, “Type”, “Format”, and “Language”.

```
<rdf:RDF
    xmlns:rdf="http://w3.org/TR/1999/PR-rdf-syntax-19990105#"
    xmlns:dc="http://purl.org/metadata/dublin_core#">
    <rdf:Description about="http://www.dlib.org">
      <dc:Title>D-Lib Program - Research in Digital Libraries</dc:Title>
      <dc:Description>The D-Lib program supports the community of people
       with research interests in digital libraries and electronic
       publishing.</dc:Description>
      <dc:Publisher>Corporation For National Research Initiatives</dc:Publisher>
      <dc:Date>1995-01-07</dc:Date>
      <dc:Subject>
        <rdf:Bag>
          <rdf:li>Research; statistical methods</rdf:li>
          <rdf:li>Education, research, related topics</rdf:li>
          <rdf:li>Library use Studies</rdf:li>
        </rdf:Bag>
      </dc:Subject>
      <dc:Type>World Wide Web Home Page</dc:Type>
      <dc:Format>text/html</dc:Format>
      <dc:Language>en</dc:Language>
    </rdf:Description>
  </rdf:RDF>
```

(from : the W3C Proposed Recommendation for RDF, 5 January 1999)
RDF is now a W3C Recommendation : http://www.w3.org/TR/REC-rdf-syntax/
The schema specification for RDF is still a proposed recommendation.

The RDF definition is rather large (about 33 pages), and has many features to allow its use for all kinds of metadata. Its generalised use would be useful to get the structure of any metadata and the semantics of the structure, but its complexity might slow down its use. Actually, it could even be used to describe resources that are not really metadata : it would give universally understood semantics to the documents. Implementing an RDF parser is probably not very simple, but fortunately there are already free parsers available.

Practical considerations : using XML today

Creating and Editing

Many tools have been created recently, to :

- create and edit DTDs
- create and edit XML documents (validating or not against a DTD), usually for general XML document with no feature specific to a DTD. Some publishing applications previously using SGML allow the editing of a text with XML formatting.
- create XML documents automatically from databases
- convert SGML or other hierarchic formats with no interface to XML

Visualisation

A simple text editor can be used to look at an XML document. Various programs can also display the XML tree graphically and let a user browse it, like Jumbo (Murray-Rust 1999) (also allowing a modification of the document), Internet Explorer 5, or a simple "XML browser" applet designed during the thesis.

However, a specific interface for the DTD is usually required. There are many ways to create a user interface for a DTD :

- with a direct transformation into HTML :

  this works for a given DTD, but a programming language has to be used for that, and this solution is not very flexible. Also, the features of the interface are limited to the HTML interface (with no support for XLink, for instance).
- XSL :

  currently, XSL definition is not finished, and the few programs pretending to support it are often buggy (like Internet Explorer 5). The best solution to use XSL today is probably to transform an XML document with its XSL stylesheet into an HTML document with a CSS stylesheet. Current browsers are (nearly) able to display HTML documents with CSS stylesheets, so no plugin would be required for the users.
- CSS :

  It is possible to use CSS with XML documents, to specify a user interface (again, this is nearly working with the latest versions of Netscape and IE). However, CSS is not, in my opinion, adapted to XML : for instance, it is not possible with CSS to add a text to the XML document before it is displayed. CSS is mainly used to add text styles, fonts or colours, but it cannot change the whole structure of an XML document to present it independently from its initial structure (this is possible with XSL).
- DSSSL :

  this complex stylesheet language was used with SGML, and there are a few tools able to use it to display SGML or XML documents, like Jade (Clark 1999). But these tools work only on some platforms, and they sometimes have compatibility problems due to the differences between SGML and XML : although XML is in theory compatible with SGML, in practice it is not, because all the features of SGML have not been implemented in the SGML parsers, and XML is using some rare features, and a specific syntax for empty elements.
- Java :

  a Java applet is currently a good solution to create an XML browser, especially if it requires specific features only available with a programming language, like fast interaction with the user or graphical capabilities. When XSL will be defined and implemented, it will be better to use it instead of Java for applications that do not require a complex user interface. Jumbo (which has been designed for CML (Chemical Markup Language) and the AML browser are both Java applets.

Representing XML documents in a programming language

DOM

The Document Object Model (DOM) (W3C 1999e) is an API (Application Programming Interface) for HTML and XML documents, created by the W3C. It provides a standard set of objects for representing HTML and XML documents, a standard model for combining these objects, and a standard interface for accessing and manipulating them.

One of the origins of the DOM is the browser war between Netscape and Microsoft, for Dynamic HTML : the two browsers were using different interfaces for manipulating HTML documents, causing the developers to program their programs twice over or abandon their web site compatibility. The W3C, having Netscape and Microsoft as two of their (important) members, had the opportunity to define a new, standard API that would be accepted by the two browser vendors. SGML Grove (Graph Representation of Property Values, also representing the result in memory of a parsing) and HyTime have also influenced the definition of the DOM.

The current DOM version, the DOM Level 1 specification, became a W3C recommendation on October 1998. It is divided into two parts : the Core, for any XML documents, and another part specific for HTML documents. Applications dealing only with HTML documents do not need to implement all the XML interface, and applications using XML with no HTML documents do not need to implement the HTML interface.

DOM is *not* a binary specification, or a set of data structures : no restriction is made on the way the objects are implemented, DOM only describes the interface (a set of procedure names with parameters, and property names for the various “objects”). It can even be implemented in a non-object-oriented language.

The API is limited, but extensive. It is difficult to implement only a part of the Core, so DOM implementations are rather “heavy”, which is an issue on the web. Sun has expressed the will to implement the DOM in the Java API, which would solve this problem : the clients would have the code locally, so it wouldn’t have to be transferred at an applet’s loading time. More information on the DOM can be found at `http://www.w3.org/DOM/`.

A simplified interface

Another API for XML documents was developed for the thesis, before the DOM specification. There was no standard API at the time, and a number of people were implementing different interfaces with common features but no compatibility.

The interface developed is much lighter than the DOM, and it voluntarily restricts XML. It is implemented in Java with 2 classes, while the DOM recommendation for Java implementation includes 18 classes. In fact, this interface does not really describe XML documents, but only the tree in memory : there is nothing to represent a forest (a set of trees), for instance. There are, however, all the basic functions to access the tree, together with a few more complex functions, and one to output an XML document corresponding to the tree. Clearly, this doesn’t meet all the requirements for a large application using many XML DTDs, but it is sufficient for an XML application using one or two simple DTDs.

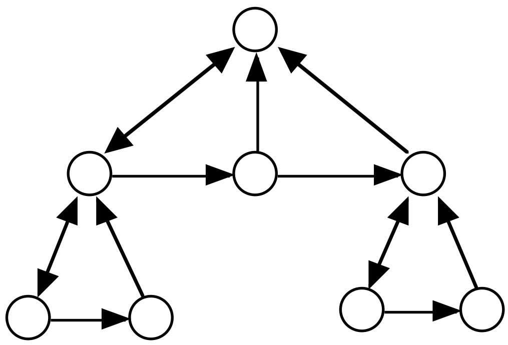
The two classes and their attributes are : (the complete interface is in Appendix 1, “Class xml.XMLTree”) :

XMLTree

tag :	String
val :	String
attr :	Vector of Attributes
father :	XMLTree
next_brother :	XMLTree
first_child :	XMLTree

Attribute

name :	String
val :	String



links between the nodes in this implementation

Implementation of an XML parser

SAX

SAX (Simple API for XML) (Megginson 1999) is an event-based API for XML parsers. It is not a W3C standard, but it has been widely used since its definition, in May 1998.
The “event-based” specification means that a parser following this API has to generate events (by calling procedures) when it parses an XML document. For instance, the document :

```
<?xml version="1.0"?>
<doc>
<para>Hello, world !</para>
</doc>
```

generates the events :

```
start document
start element: doc
start element: para
characters: Hello, world !
end element: para
end element: doc
end document
```

DOM, on the other hand, is a tree-based API, always referring to the tree in-memory to access an XML document. It might be better, for a very long document that doesn’t fit in memory, to use only SAX. Many DOM implementations (like the Docuverse’s DOM SDK) are actually based on top of SAX : the DOM tree(s) are created with the events generated by an SAX-conformant parser.

The SAX interface is separated into four interfaces that have to be implemented by a parser : the “document handler”, the “DTD handler”, the “Entity resolver”, and the “error handler”.
More information on SAX can be found at http://www.megginson.com/SAX/.

A simplified XML parser
An event-based parser has been created by us, before the SAX definition. It is used to parse XML documents and create a tree in memory of the XMLTree class. The idea is the same as for SAX, although not all the features of XML are implemented : there is no "DTD handler" and no "Entity resolver" (the entities are not supported).

The advantage is that it's very "light" and fast : if we add the implementation of the parser and the tree in memory, it is still smaller than the *specification* for the DOM !

The javadoc-generated documentation is available in Appendix 2, "Class xml.Parser_XML".

Thoughts and perspectives on the use of XML

Adding content to a structured document, or adding markup to a text ?
More and more metadata is required today to handle complex information on the web. One approach to solve this problem is to add company-specific tags in existing HTML pages. As the browsers don't recognise the new tags, that don't display them and they are hidden from the user's view. This kludge is opposed to the XML approach, because the new tags are defined nowhere, and can be in conflict with other company-specific tags. The HTML documents created in this way are far from conforming to the current HTML DTD, thus slowing down the interoperability efforts.

The adepts for this technique can argue that only HTML is really available today, and that it's a simple solution to add metadata to existing web pages. But, by continuing in this direction, they forget about switching to a true solution.

The use of XML, with a DTD, is more rigorous and provides more flexibility. Even when a document has little metadata, it is better, for automatic processing, to format it rigorously. At the time when style and metadata information was written by hand, it was understandable, from a human point of view, to add some metadata tags only when needed, but this is history.

XML islands in HTML
This was another threat to true XML : Microsoft, probably afraid of rewriting the core of its browser, suggested to support XML only within HTML documents. At the time, they were trying to justify that IE could handle XML without handling real XML documents. But the XML defenders refused any compromise : this easy solution would have destroyed the idea(l) of DTD conformance (or at least rigorous structure) for all documents over the web. In the end, it seems Microsoft changed its mind because the new IE is trying to support real XML documents.

A mix of DTDs (like HTML and another DTD in a single document), more and more needed, can be achieved with the namespaces standard, without loosing the good properties of XML.

An XML Tower of Babel ?
HTML was a standard for any information on the web. Tomorrow, many different XML languages with various DTDs will coexist. Programs using all the web resources, for instance for indexing, need some uniform resource for all the documents, and this isn't quite ready for XML yet. The Dublin Core provides a minimum set of metadata that could be included in all XML documents with the namespaces standard. To add more metadata in a uniform way, RDF will probably be used. The whole mechanism is being built, but it will be years before it is significantly used.

Even with a standard system for basic metadata, there will still be the risk that several DTDs are written for the same applications. The result would not be worse than current HTML files, but it would reduce the advantages for using XML : automatic processing would still be possible, but it would require conversions between the documents. Sometimes the conversion would not be complete, causing the usual compatibility problems. Well, some companies seem to like proprietary formats causing incompatibilities because of their commercial advantages, so they will probably appreciate the fact that this is still possible with XML...

To avoid the “Tower of Babel” risk, DTDs can be organised in two ways. First, the base elements to build the DTDs can be common among many DTDs. An example of this is MathML, an XML language to write mathematical expressions : this language is not very useful alone, but can be used within other DTDs. Second, DTDs for distinct applications can be clustered together to create a new language able to address many different subjects.

The political side

The last section shows clearly that cooperation is an important factor for XML applications. Only through cooperation can interoperability be improved, and XML only simplifies the problem, it does not solve it by itself.

This doesn’t mean that official agreements have to be signed to design an XML language : most companies have now realised that cooperation was possible even with rivals, and that not only the customers benefit from a standard, open format. A striking example of this is the cooperation between Netscape and Microsoft within the W3C for designing XML, while they still have a number of ongoing court cases.

As more and more people realise they are using similar DTDs for some applications, they tend to think a unification of the formats is possible : XML also encourages cooperation.

XML and databases

Many DBMS (Database Management Systems) now have the possibility to create an external view of a database as an XML repository (a set of XML documents), and a few databases are actually based on real XML repositories. But the success of XML should not make one forget the usefulness of traditional databases : they provide many important processing features with an internal structure designed to optimise the speed.

It would be very difficult to provide the same features with a database using a collection of XML documents as its internal structure. As the databases become more and more oriented toward XML and other views lose of their importance, however, it could be interesting to find a solution to optimise the creation and use of XML documents without losing the traditional database features.

XML is a bridge between unstructured data and structured data with strong typing. Compared to plain text or HTML, it adds structuring information, but it still allows the use of unstructured data, and there is (yet) no standard mechanism in XML to specify strong typing as in databases. A proposed solution, to specify for example that an XML element content is a float, is to use processing instructions in the XML document. But this idea (and the associated syntax) is not in the XML specification, and it would be more logical to add this information in the DTD (actually, a new schema (DTD syntax) for XML is being created now (W3C 1999f), with such features (`http://www.w3.org/TR/NOTE-xml-schema-req`)).

The lack of strong typing in XML is an issue for databases, for instance to query databases. This problem can appear for the creation of an XML query language : will it be possible to use it to query databases? If a strong typing mechanism is provided in XML, it will add to its complexity, which is also an issue considering XML goals.

Is there a solution to use XML documents internally and still benefit from traditional database advantages? A problem for this is the tree structure of XML documents : the tree structure is not a usual structure/type in current databases, especially relational databases. In object-relational databases, a tree could be encapsulated into an attribute, but this approach would dismiss the representation of the tree from the database structure, thus not truly solving the problem. With object-oriented databases, it would be possible to create a class for an element in the DTD (or, if the DOM is used, simply use the same objects in the database as in the object-oriented code), but the instances would become complex and heavy for DTDs having a lot of elements and a deep structure. If methods are to be associated with certain elements in the DTD, object-oriented databases are a solution, but they could be a little slow for the processing. This leads to the question : at what point in the tree structure should the database representation stop? A type in the database could represent a whole XML document, or an XML tree, or an XML leaf (an element without any children). The indexing of the database should also be taken into account : some elements in the structure can be chosen as a key for the indexing to speed up a search, but this requires a representation of the tree in the database representation.

It has been suggested that hierarchical databases could be used to represent XML documents. They have not been abandoned by mere chance, though : they were useful for quick processing based on a fixed key, but they were not flexible and required knowledge of the physical representation. Maybe a new conceptual model could be created, more adapted to XML, but in the meantime the OODBMS (Object Oriented Database Management Systems) are probably the best suited to represent XML documents.

An experimental DBMS for XML and OEM (a language similar to XML but not supporting DTDs) documents is Lore (Goldman, McHugh & Widom, 1999, Stanford 1999), under development at Stanford University. Lore is focused on queries in XML repositories : it features a specific query language for XML, and a way to make a query with an XML document (as an example of the documents to look for). Its internal structure, originally developed for the OEM data model, was adapted to XML.

IV. SOFTWARE STRUCTURES

To use distributed process and exchange information in various ways, it is also important to use good software structures, with abstract interfaces. The object-oriented concept was a key evolution in software structures, but is still relatively recent. Encapsulation of "objects" with interfaces is a fundamental idea to perform tasks over distributed sources, and this choice deserved a whole chapter of explanations.

History

At the beginning of computer science, all the hardware required was very expensive, and the software was limited in size and complexity. The first systems were only taking care of dialoguing with the hardware, giving a small interface to programs. All these programs were written in assembler, and could not be used on different machines. To save up memory, the programs could modify their own code, and most data were a part of the program.

With the hardware advance in cost and capacity, it became possible to write bigger programs, and more structuring was needed to avoid bugs. The system became in charge of more and more processing needed by different applications, and part of the system was copied into read-only memory. Safe programs were possible thanks to more restrictions : a separation occurred between data and programs, with the appearance of read-write-execute flags for different segments dividing the memory. Different kinds of memory and storage of information appeared, each with a speciality.

Then, programs became divided into procedures, thus allowing collective work on the same software. And later, programs could communicate with each other, adding another level of complexity. The operating systems became composed of many layers separated by specific interfaces.

Each time a significative progress was made, it was with the help of structuring, putting new restrictions on the software : at first, it could have looked to be a hindrance for speed and overall performance, but because of our limited human minds and our limited communication possibilities, creating a new interface has proved to be a fundamental step to reach another level of complexity.

Abstraction is useful tool to reduce complexity. As software became more complex the ability to work with it using abstractions became more important. One such abstraction that became interesting and useful from research in the 70's is the Abstract Data Type. An ADT is an encapsulation of a data structure along with the procedures (operations) that manipulate that data. It is said to be encapsulated when its state is implied, meaning that it is only visible and accessible through the procedures. Languages that implemented ADT's were Modula2 and Ada.

Today, ADT's utility has evolved into a collection of languages said to be object-oriented. Such languages express ADT's as objects. To express an ADT as an object also requires something to dynamically create new copies, which have attributes (state) and behaviour (operations).

The object oriented concept and the cross-platform language Java were two other huge steps towards reusable software.

The object-oriented concept

The object-oriented approach is based on an intuitive correspondence between a software simulation of a physical system and the physical system itself. The analogy between an algorithmic model and a mechanical model leads to the common use of the word "object", although the use of abstract objects, not connected with the real world, is possible with the software model. The concept of objects in computer science was first used with Simula, where an object was thought of as a self-contained program having its own data and procedures (Birtwistle, Dahl, Myhrhang & Nygaard, 1973).

According to Brad Cox (Cox 1986), the main problems addressed by the object-oriented concept are the bulk of code, and the surface area. As a project grows, the number of individuals implied increases, resulting in an increased amount of effort for coordination. Dividing a project into independent sub-projects reduces this overhead. Brad Cox associates the idea of sub-projects' surface area with the following issues : information hiding, abstract data types, time sequence of operations area with the following issues : information hiding, abstract data types, time sequence of operations (the surface area increases when a consumer has to do things in order, as required by the objects), garbage collection, protection domains (like files or pipes used for communication between sub-elements), concurrency. These problems are well solved by modern object-oriented languages, although not perfectly.

To summarise, the good properties of the object-oriented concept are :

- encapsulation : objects form natural data abstractions boundaries and help focus a design on system structures instead of algorithms. As the objects are behaviourally autonomous, the reorganisations become easier.
- data abstraction : with the emphasis on *what* rather than *how*, and the generalised use of inheritance, every part of a program becomes reusable.
- dynamic binding : polymorphism is possible at runtime, giving more freedom to programming and enabling the use of more intuitive concepts.

These good properties can be obtained by other means than objects, but objects give an intuitive and uniform framework.

Although object-oriented programs have not replaced all the other software structures, they have developed a lot during the last ten years. Two elements have slowed its progress : using objects generally requires a maintenance overhead, which is not appropriate for some procedures having to be very optimised (but these procedures can be included into an object-oriented framework), and object-oriented programming is not as simple as procedural programming, so it is often not used for short programs. Compatibility requirements with older programs is another reason for not using object-oriented programs, although interfaces are possible.

A few studies have tried to measure the importance of the concepts underlying the object-oriented one. Linda Rising and co-authors defined a metric for information hiding in (Rising & Calliss, 1994). The most important criteria in their information hiding metric seems to be the number of design decisions for a given module. Another study (Abreu & Melo, 1996) tries to evaluate the following metrics : Method Hiding Factor, Attribute Hiding Factor, Method Inheritance Factor, Attribute Inheritance Factor, Polymorphism Factor and Coupling Factor. Unfortunately, this study is not based on a big dataset, and is not conclusive for some of these factors. As they state in their conclusion, “there is an obvious lack of conclusive studies in this field and further research is required”.

The foundation of the Internet

Internet history

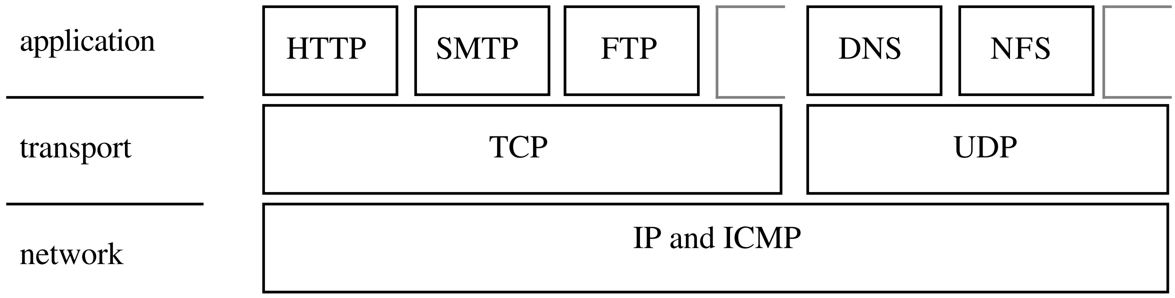
The Internet began as the ARPANET during the cold war in 1969. It was developed by the US Department of Defense's (DOD) research people in conjunction with a number of military contractors and universities to explore the possibility of a communication network that could survive a nuclear attack. It continued simply because the DOD, its contractors, and the universities found that it provided a very convenient way to communicate.

For the first decade that the Internet was in existence, it was primarily used to facilitate e-mail, support online discussion groups, allow access to distant databases, and support the transfer of files between government agencies, companies and universities.

During the early 1980s, all the interconnected research networks were converted to the TCP/IP protocol (this enables all of the networks of the Internet to send data back and forth), and the ARPANET became the *backbone* (the physical connection between the major sites) of the new Internet, which comprised all TCP/IP-based networks connected to the ARPANET. This conversion to TCP/IP was completed by the end of 1983 - and the Internet was born.

TCP/IP

TCP/IP is actually a layered suite of protocols, as described by the graphic below. IP (Internet Protocol) is the basis, with ICMP (Internet Control Message Protocol), is the basis : it provides a mechanism to send units of data “datagrams” on a physical network. Based on IP are two protocols providing a higher level interface to transmit the information : TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). UDP is the fastest and most simple, and doesn’t add much to IP, while TCP establishes a whole “connection”, and always ensures that the datagrams arrive in the order they were sent, with uncorrupted data. Based on TCP and UDP are many “application protocols” for various tasks over the network, most of them using a client/server system. For instance, HTTP (HyperText Transfer Protocol) is used to transmit HTML files, SMTP (Simple Mail Transfer Protocol) delivers electronic mail, FTP (File Transfer Protocol) is used for interactive file transfer between hosts, DNS (Domain Name Service) maps IP addresses to the names assigned to network devices, and NFS (Network File System) allows files to be shared by various hosts on the network as if they were local drives.



The success if this architecture over systems mixing the different layers clearly shows the importance of interfaces in the computer science development. Yet, it is not easy to decide of the interfaces. The OSI (Open Systems Interconnection) model has 7 layers, and not all of them are present in the TCP/IP model.

OSI layers

LAYER	ROLE
application	applications using the network
presentation	display, formatting, encryption
session	initiating, maintaining, terminate logical session
transport	providing end-to-end error detection and correction
network	managing connections across the network
data link	providing reliable data delivery across the physical link
physical	transmitting data bits

The World Wide Web

In 1990, HTTP, a hypertext Internet protocol which could communicate the graphic information on the Internet, was introduced. Each individual could create graphic pages (a Web site), which then became part of a huge, virtual hypertext network called the World Wide Web (WWW). The enhanced Internet was informally renamed “the web”, and a huge additional audience was created.

Many other applications have been created on top of TCP/IP, such as WAIS (Wide Area Information Servers), a keyword-based IR system, and Gopher, to access a system of interlinked informational menus. But, as the web evolved, it tended to replace the other non-multimedia protocols, and it has already killed WAIS and Gopher.

The web today is based on the already rich HTML standard, and also uses other technologies to improve the communications between clients (using a web browser) and servers, and to create interactive interfaces. XML, replacing HTML in many cases (and also changing the HTML standard into an "XHTML" language conform to XML), also allows the publication of more "data-oriented" information.

Interface-oriented data (HTML, HTML/CSS)

HTML is the basis for the web. The "web pages" are written in HTML, even though they can point to other files in other formats, like images. HTML, defined by a SGML DTD, has evolved a lot and its definition is much more complex than it used to be. It is now (in version 4.0) using stylesheets, like CSS (Cascading Style Sheet language), to specify the fonts, colours, and layout effects. CSS instructions associate HTML elements with styles, so that it is no longer required to copy the same style information each time a given HTML element is used, for instance for a title.

Communication with a server (CGI, Java applets and servlets)

HTML uses a mechanism to send information to an application on a server, returning another HTML file. The HTML description of the information to send is done with a *form*, associating variable names to a graphical interface (text field, check box, ...). Once a form is filled by a user, the choice can be send as name/value pairs to a CGI program. This system has the advantage of being very easy to use, to create a web user interface to a program. It is however limited in two ways : first, the user interface is restricted to a few HTML elements, fine for a form but not for a more complex user interface. In fact, with the success of the web, more and more users have been using application with a very restricted interface, while they expect a much better presentation from their local software. Secondly, the Common Gateway Interface can only be used to create a single input for a program, execute it, and get a single result as a whole HTML document.

These limitations led to the development of Java applets : these are short programs, downloaded in real-time by a user, to create an interactive user interface and communicate with a server. Because the code is downloaded locally, all the interface work is computed locally, avoiding the delays with the programs having their interface controlled remotely. In practice, unfortunately, for security reasons, unsigned applets are not authorised to communicate with other servers than the one they reside on. The process of signing an applet is very difficult today, because the two major browser vendors have chosen a different, incompatible solution. As a result, the servers sometimes have to do work that should be done locally, like calling other servers on the internet.

A new alternative to CGI programs are Java servlets, using HTML forms at the client side and a Java program at the server side. They are an intermediary solution between CGI programs and Java applets.

Interactive interface (Java applets, Dynamic HTML)

Another solution to create an interactive interface is to use Dynamic HTML. "Dynamic HTML" is a term used by some vendors to describe the combination of HTML, style sheets and scripts that allows documents to be animated. An example of it is JavaScript, a scripting language to modify interactively an HTML document. Dynamic HTML is usually not used to communicate with a server in other ways than CGI forms : it is mostly used to create simple interactive features to an otherwise static web page. It has also suffered a lot from the incompatibilities between Netscape and IE, but they have finally agreed on the DOM (Document Object Model, described in the XML section) interface to use the HTML trees with the same interface.

Data-oriented information (XML, XML/XSL, XML/Java)

XML adds a new dimension to web publishing, with the possibility to publish data-oriented documents that can be used both for automatic processing and for the creation of a graphic interface. However, XML alone does not provide any user interface, so it has to be combined, either with style sheets like CSS or (preferably) XSL, or with a programming language like Java. This is a radical change from the evolution that occurred with HTML and Java : the web was becoming more and more interface-oriented, with less metadata and other data that could be automatically processed. For instance, a Java applet can provide a whole user interface, with no indexable text at all. By using XML in combination with Java, one provides both a data with an explicit meaning, and an interactive user interface.

Distributed Objects

Changing to object-oriented programming brings many improvements, such as components reuse. It is very useful to be able to share clearly interfaced parts of code, but it is sometimes needed to have a server do some tasks at a client request. This was done with simple client/server programming, for instance with sockets, but the use of objects over a network adds many new questions.

A system to distribute objects over a network is usually required to let client objects use remote server objects transparently, as if the objects were part of the same program. This is the object-oriented equivalent of the RPC (Remote Procedure Call) : an object has the role of a server, and another one is the client.

Orfali, Harkey and Edwards wrote (1996) : “Distributed objects are the next big wave. We believe that distributed objects are essential for making the intergalactic client/server vision real” and “We believe that by early 1997, ORBs will become ubiquitous. And we won’t be talking about them any more”. The progress were not that fast, and in 1999 ORBs (the core of distributed object systems) are still not ubiquitous. But the needs for a universal interface between programs is more present than ever, and distributed objects still appear as a good solution.

The following issues have been well addressed :

- publishing : the objects providing public services should be known, and their public methods available
- security : the services should be protected against unauthorised intrusion
- transparency : the objects should be accessed nearly as easily as if they were local
- reliability : the use of the network, even over long distances, should not affect the methods reliability
- speed : the work load added to manage the distributed objects should be minimised
- granularity : the designers should be careful about the size of the objects they publish (for instance, it is not reasonable to create a shared object for a simple integer)

Various systems have been proposed to manage the communications between the objects, sometimes with different objectives (Jayson 1999). CORBA, presented here, has the advantage of being vendor-neutral.

The Common Object Request Broker (CORBA) (Keahey 1999)

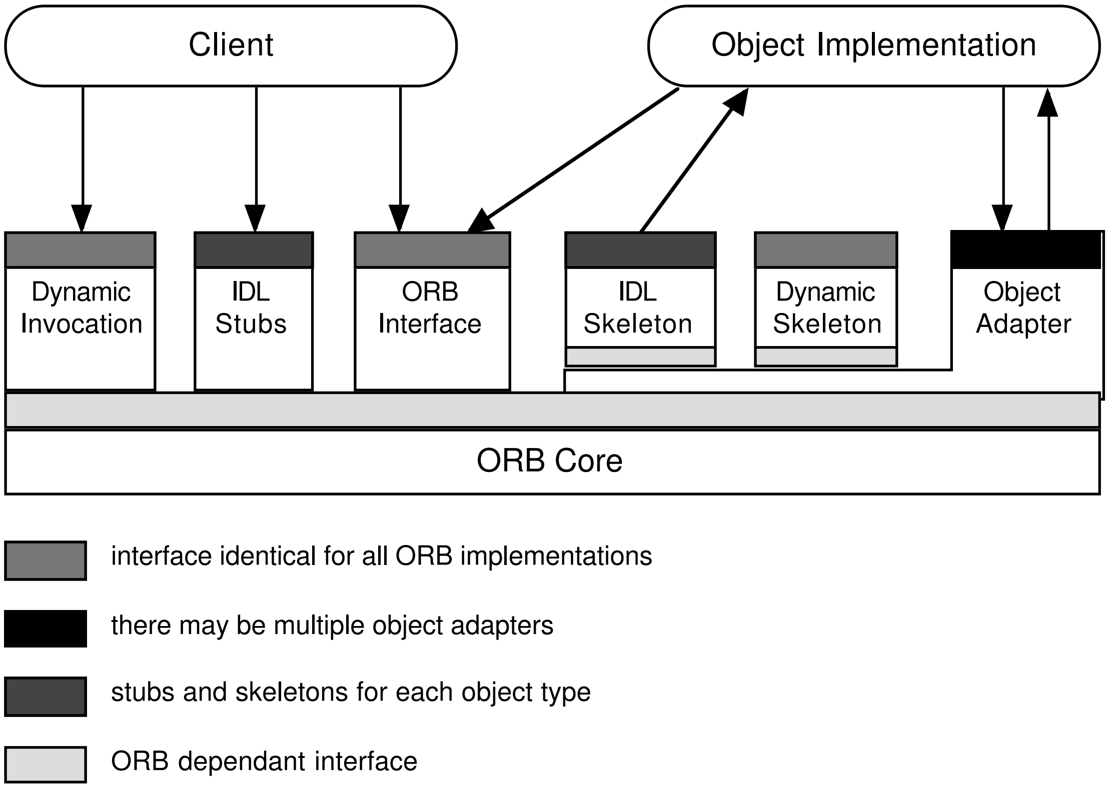
CORBA specifies a system which provides interoperability between objects in a heterogeneous, distributed environment and in a way transparent to the programmer. Its design is based on OMG (Object Management Group) (OMG 1999) Object Model.

The OMG Object Model

The OMG Object Model defines common object semantics for specifying the externally visible characteristics of objects in a standard and implementation-independent way. In this model clients request services from objects (which will also be called servers) through a well-defined interface. This interface is specified in OMG IDL (Interface Definition Language). A client accesses an object by issuing a request to the object. The request is an event, and it carries information including an operation, the object reference of the service provider, and actual parameters (if any). The object reference is an object name that defines an object reliably.

The Basic Mechanics of issuing a request

The picture below shows the main components of the ORB architecture and their interconnections :



The central component of CORBA is the Object Request Broker (ORB). It encompasses all of the communication infrastructure necessary to identify and locate objects, handle connection management and deliver data. In general, the ORB is not required to be a single component; it is simply defined by its interfaces. The ORB Core is the most crucial part of the Object Request Broker; it is responsible for communication of requests.

The basic functionality provided by the ORB consists of passing the requests from clients to the object implementations on which they are invoked. In order to make a request the client can communicate with the ORB Core through the IDL stub or through the Dynamic Invocation Interface (DII). The stub represents the mapping between the language of implementation of the client and the ORB core.
Thus the client can be written in any language as long as the implementation of the ORB supports this mapping. The ORB Core then transfers the request to the object implementation which receives the request as an up-call through either an IDL skeleton, or a dynamic skeleton.

The link between the object-oriented concept and data structuring

The use of XML documents, composed of trees of XML data, in object-oriented programs, leads naturally to the definition of associated trees of objects in memory. The DOM standard interface, and the SAX API for parsing, both described in the XML chapter, can be used to do this.

In object-oriented programs, a class defines an interface and procedures for the objects, and the instances of the class are the real objects, with their data, which can be manipulated in the programs. Similarly, a DTD provides an interface for XML documents, and the XML documents can be viewed as instances of the DTD.

The trees offer for data an improvement similar to the object-oriented programming for the

The trees offer for data an improvement similar to the object-oriented programming for the software. Objects and trees also share some common characteristics : they provide encapsulation, a fundamental property for any complex processing, and they both use a relation "element of" to link the different elements. Other relationships are found between objects and between XML trees, showing that both are suited to represent networks.

These ideas lead naturally to a direct mapping between objects and XML trees. An API for this, and the implementation, has been developed by JXML, Inc.. It is called "Coins". Here are a few extracts from the Coins web page (JXML 1999) (`http://www.jxml.com/coins/`) :

"Coins is an extensible system for processing XML documents, and provides a simple API for processing XML documents as trees of object. Coins supports the use of pre-existing Java Beans, providing several alternative approaches to connecting the data of an XML document with various types of Java objects."

"The tree of objects used by Coins to represent a document is defined by the Document Object Model (DOM) Specification. The DOM provides an API for navigating the document's object tree. Coins uses a SAX-conformant parser and the Docuverse DOM SDK to convert an XML document into a tree of objects, each object corresponding to an element in the XML document."

Usually, the data structures were determined by the structure of objects or other structures in memory. With databases, however, the data structures were defined before. With a direct mapping between the objects in-memory and the XML documents, the structure can be defined for both at the same time.

Artificial Intelligence Structures

Some neural networks have a very special software structure, that could be qualified as "fuzzy". For example, multi-layered neural networks with a back-propagation learning method, have a given static structure consisting of nodes and links between them, but the learning process changes the weights of the links, so that a new structure, resulting in a different process taking place in the network, is created automatically. The new structure resulting of the learning process, is usually unknown : the neural network is seen as a "black box". The reason why it can be called fuzzy is that there might be illogical associations at the end, and unused or unnecessary links (although the unused links can be removed automatically because of the low value of their weight). This structure is usually not optimum for the problem, but it has the advantage of having been found automatically.

The structures found automatically are currently very simple, and are used to solve simple tasks ("simple" in the sense that the task can be described with only a few words) with only a few inputs/outputs. In comparison, whole programs in high-level programming languages are much more complex. This is because there is an extensive use of human symbolism in traditional programs, which cannot (at the time being) be created automatically in neural networks. To solve a complex problem with neural networks, many of them can be used to solve basic tasks, with a high-scale structure using the traditional symbolism.

Expert systems, although using declarative programming, make an extensive use of symbolism and thus can reach higher structure levels than neural networks. Systems use more often neural networks to solve simple tasks, with the expert system processing on top of it, than the opposite (using a neural network to coordinate various small expert systems).

An interesting development of neural network is based on the use of different network layers working at different levels of abstraction, and linked together with interfaces . The usual weighted links between the nodes of two layers can be seen as an "interface", and compared to an API (Application Programming Interface). With this point of view, the existence of interfaces in neural networks is also an advantage because it allows the creation of complex structures, as it does for traditional applications.

Another interesting field of research is based on the use of small, specialised A.I. (Artificial Intelligence) modules (sometimes called “agents”), communicating together to reach a higher global organisation. This orientation of A.I. is often opposed to the building of a single big A.I. application solving all the problems. These agents can also create a “fuzzy” structure automatically, but it can be wondered whether they can reach a level of abstraction as high as an application with 5 clearly separated interfaces.

Interest for distributed frameworks, self-organisation and automatic classification

The success of distributed frameworks can be explained by two kinds of motivations : technical, and political/organisational.

The technical reasons are caused by limitations of the network performance (speed, reliability), and of the individual servers capacities for storage and processing. Distributing the work load and the information storage among different servers allows the achievement of efficient information delivery, fault-tolerance and security. With the clear interfaces between the modules, it becomes also easier to add a new part to the whole network.

Distributed frameworks can also improve the organisation for a large project, especially if there are many people working on it, for different tasks, with different data, at different locations. Not all these characteristics are required for a given project, however, to make a distributed framework useful. In fact, sometimes, the interest of having formal delimitations between groups with distinct responsibilities is so high, that a reorganisation can be decided before a change from a centralised framework to a distributed one. The goals for this reorganisation are an improved software framework, but also an improved intern organisation. With the improved flexibility, later changes in the organisation will be much easier.

Choosing a distributed network can be a political decision, motivated by a decentralisation will. However, a distributed framework can very well be controlled by a single authority, in which case it is technically distributed, but not “open” in the sense that anybody could add something to the existing network.

A distributed framework can also be used for mobile computing : each piece of software can be located anywhere, and move independently from one another.

The drawback is an added complexity : the network often needs to be managed, with synchronisation between different operations. Its conception also requires the definition of common protocols. Not all applications are efficiently distributable : some problems cannot be decomposed into parts that could be independent enough to improve the performance if they are distributed.

Some networks have the ability to auto-organise : a new knowledge based on the links between the network nodes can be created without a supervised processing. This is the case, for instance, of neural networks : even when the learning is supervised, the network is a “black box”, and the connections between the neurons are not specified explicitly. But this is also true for a network such as the web : the organisation is very loose, but the links created independently can create a new knowledge automatically. More generally, any network can become a complex structure, even when it has very simple bases. Because of its possibly complex structure, a network can represent many things, and it is useful for unknown knowledge representation (as with A.I.).

When huge data volumes have to be classified, a human work is no longer possible, and an automatic classification is needed. A network is a good way to classify data when the classification is complex. Machines can already do nearly anything a human is capable of when they have all the needed metadata to represent explicitly the “background” usually reserved to human use. This background information can include examples and rules, from which a network can be trained.

Processes can also be distributed, along with their associated data. An encapsulation of a small data set and associated procedures is usually called an “object”, or an “agent”, especially when many objects or agents have the same interface. A whole network of agents can be created, to do even more things. This is often used for simulation, or to represent a person on the internet, but is not limited to these tasks. Agent systems are distributed by nature, and inherit of the advantages and drawbacks of distributed systems. A network of small agents can be represented as a bigger agent, for a higher level computing. Actually, the same concept is behind all these “new” technologies : it is the not-so-new object-oriented concept.

Networks can be used for automatic classification, especially when the nodes of an existing network are to be classified. In this case, a partitioning algorithm can be used efficiently, as demonstrated here.

Software and data structures proposed

Today’s information retrieval on the web is “document-oriented” : most of the time, search engines and directories are used to find HTML documents. However, documents can contain several “objects” with different meanings, which could be searched individually. Also, a single object, identifiable by a given semantic, can be represented by several documents with different interfaces. For example, a report can be available in different text formats, and a single chapter could be searched for individually even if it is written in a part of an HTML document or if it is contained by several HTML documents.

Humans are inclined to search for objects with a semantic instead of documents. It is not satisfying, for instance, to get 3 times the same document, from different locations or written in different formats, in a list of 10 documents retrieved. It would be better to get just one entry for the object, and to get the different documents (in various formats and locations) after a click on it. Also, a relevant object can be missed because it was spread on several documents (no single document had all the search keywords), or because it was only a part of a document (and the object keywords were lost in the document keywords).

The object-oriented concept was successfully applied to databases, where the search is a very important part, and it could also be profitably applied to semi-structured data publishing. XML is a good way to store semi-structured data in this regard : it allows the representation of objects without any interface specification, and XML documents can be broken down into sub-entities. The sub-entities can be accessed directly with the XPointer standard.

There is no standard information retrieval protocol for XML yet, but the W3C was submitted a note for an XML query language, XML-QL (W3C 1999g) (`http://www.w3.org/TR/NOTE-xml-ql/`). This would allow complex searches in XML repositories, but is not suitable for other kinds of data. In fact, there is a lack of a general mechanism for object-oriented information retrieval.

A good protocol for information retrieval today is Z39.50 : this client/server protocol lets a client do an attribute-based search in a collection of records. A transformation into an object-oriented protocol is thinkable, for instance by having methods in some attributes, but this system would suffer from the same shortcomings as ORDBMS. Because Z39.50 was conceived to handle only record searches, its core cannot handle an image-based search (for instance requiring real-time zooms in a vectorial graphic), or a hierarchical data (like XML) search. In practice, because it is not using HTTP, Z39.50 is difficult to integrate on the web. (more information on Z39.50 in the *Emerge* section)

An API for object-oriented information retrieval, letting each object class define a method for information retrieval, with a common interface, would provide a more general solution for information retrieval. This could be implemented with any object-oriented language, and CORBA could be used for the information exchange.

Data browsing is also improved with object-oriented APIs : each object-oriented data can be assigned a method for the user interface. This is not often used on the internet because of HTML limitations, but the situation should be reversed with XML.

Knowledge discovery also gains by using the object-oriented concept : a definition of the objects, with their semantics, prior to any knowledge discovery, is better than a purely syntactic study. Semantic links between the objects are also an advantage for the object-oriented approach. “Object-oriented knowledge discovery” does not necessarily mean “distributed” : this is a new possibility, and the rapid growth of the *agents* on the internet is a good demonstration of its importance, but a centralised approach is still possible with a collection of objects, and it is already better than using a collection of documents with no clear semantic or mixed semantics. The agents are usually focused on a task, and do not always use object-oriented data : they would gain by using object-oriented data together with their object-oriented processes.

The object-oriented concept is not new, but has not been applied to all the fields in computer science yet. However, there is no reason to stop this evolution now. It is already applied a lot for data archival and browsing (although not on the web), but is not often used for knowledge discovery and information retrieval. A proposal of the thesis is simply to apply this concept more, and especially on the web. Using it at all the information processing stages : archival, browsing, IR and KDD, will greatly simplify all the tasks, thus allowing more complex processing. In practice, it is not required to use the same object class for all the tasks, but using the same semantic for a given object at all stages should be very useful, for a system conception and its implementation. Inheritance can also be useful, especially because object classes are often exchanged freely on the internet and form the basis for more complex development.

V. THE ASTRONOMICAL MARKUP LANGUAGE

Built on top of XML, AML is a language unifying different kinds of data in astronomy. Such a unification provides a strong basis for knowledge discovery over distributed sources. It transforms a nightmare (using different file formats with different semantics and try to use links between them) into a dream (the end of the Tower of Babel). It can also be used to create new user interfaces using information that was usually displayed separately, useful for cross-discipline work.

AML, a scientific application of XML

Science requirements for data formats

Scientific data is sometimes similar to data used in the industry, but has also some specificities happening more often with scientific data than with other data. These data specificities, leading to specific requirements for the data formats and the software, can be summarised as the following :

- huge data volumes : often, databases are close to the technical limits, and an improvement in the maximum storage capacities is always followed by new surveys using all the new capacities

- separate management of quantifying and qualifying data (Rafanelli & Shoshani, 1990, Teschke, Ruf & Lehner, 1995). Quantifying data describes the measured data, while qualifying data are used to address quantifying, for instance by date or number.

- aggregate-based and read-only data access : data analysis is often conducted by the creation of new data sets created with the initial ones, and data processing on these data.

- multidimensional data (this has led to the creation of new database systems, multidimensional database systems), or tables (like astronomical catalogues; the RDBMS are more suited for these data). Other data, well represented by objects/attributes/methods, are better stored in OODBMS. In all cases, the data are mostly numerical.

- for physical observations, the data are time-stamped : as the world is changing, all the observations are unique and should often be preserved. (King 1998)

- more and more multimedia data becomes associated with the traditional data : they are usually the result of a work on the initial data, and add for instance thumbnail images or other visualisation-related data, graphs, hyperlinks, or comments.

Is XML suitable for scientific data ?

The XML standard has not been designed to handle big datasets. XML documents are text-only, and contiguous : the whole document has to be written again for a modification. The standard solution to use binary data in XML is to use links, within an XML document, to external binary files. This situation, however, does not please many scientists, who want to have a single file containing binary data with the associated metadata described in XML. Jonathan Borden reports in the XML.com journal (Rein 1998) :
“In the medical world information takes many shapes. Text-based information including histories, descriptions, laboratory data and opinions are naturally represented in XML” “The only problem is that not all things can be expressed in a vector format. In most cases, medical images have a natural raster/bitmap format and so there is a real need to deal with the bitmap data type in XML. For interchange, we need a precise graphic but we also want to include additional structural information for use by other applications such as databases.”
There are a number of solutions to solve this problem :

- use another format to add metadata in binary files, and also describe the metadata separately in XML documents. The idea of separating the metadata from the data for large data files has already been successfully tested with HTML. With this solution, attention will be paid to the synchronisation between the metadata in the binary files and in the XML documents.

- encode the binary data with the Base64 encoding, in the XML documents. This is good solution only if the binary data is not too large, otherwise the files will be wasting space and it will slow down the reading and writing of documents.

- compact the XML document and the associated binary files with the raw data together, for instance in a zip file, and use software able to use the compacted files directly. This is possible only if the data can be broken down into a header file (which will be written in XML), and one or more raw data files (which will become the binary data files, possibly large). Scientific data is often separated into a header and the raw data, so this would work in most cases. The XML documents can be indexed to allow quick searching through the repository, on the metadata.

- use the multipart/related MIME type to join an XML document with the associated binary files. This solution, similar to the previous one, was chosen by J. Borden. The MIME type solution is however not standard for XML documents, and another syntax will probably be chosen as a standard in the future, to join XML documents and binary files.

- use a database to store all the data, design it to be able to export XML documents easily, and use XML documents with their associated binary files as an external view for the database. More efforts are required for the conversions, but this would allow database operations on the data while still providing a XML view directly usable on the web.

As a conclusion, XML can be successfully used for scientific data in most cases, but care must be taken as to the solution chosen to solve the problem with binary data. In some cases, it is better to use a database instead of a repository of XML documents to store the data, for instance when the initial data is described as a set of tables, or when a search must be conducted on the binary data.

AML : Astronomical Markup Language

AML is an XML language describing various kinds of object-oriented data and metadata useful in astronomy, and is aimed at being an exchange format for astronomical data over the Internet.

The AML language is organised as a metadata section (about the AML document), and six types of “AML objects”. An AML document is composed of a metadata header and a collection of AML objects, describing different types of information. The metadata header is mostly based on the Dublin Core metadata set. AML objects may contain links to other AML objects, and to external objects such as data, images, or documents. These links are written in the XLink standard, providing a uniform linking system. The AML language can be easily extended, for example by adding a “Set of images” object. The whole AML DTD is included in Appendix 3, and the AML web site gives more information and examples : `http://monet.astro.uiuc.edu/~dguillau/these/`.

The objects defined by the Astronomical Markup Language :

Astronomical object

This describes information about an astronomical object, with the identifiers (the names), the coordinates, the object type, other information, a list of measurements, and 2 lists of links to other AML objects. The first list of links is using an identifier to specify an AML object, which can lead to different AML documents depending on the server used to get the information about the AML object. The second list is using URLs to specify AML documents directly : this can be useful in the case of an AML repository, with many files existing on a given server. Examples of links are a bibliographic reference, or a link to an image of the object. These lists of links are also included into all the other

AML objects.

Article

This part describes information about an article. A unique identifier is used for astronomical articles, and this specificity of the astronomy field is very useful to link the AML objects.

Table

This includes metadata for a table, and a link to the content of the table.

Set of tables (catalogue)

A set of tables is a list of tables linked together, with information about the set.

Image

Again, only the metadata is included in this part, with all the information about the image and the way it is stored, and there is a link to the raw data of the image. The link can also point to a FITS file (an old standard format for images in astronomy).

Person

This describes information about a person, usually an author of astronomical articles. Attention has been given to the way the name of the person is stored, with a clear tagged separation between the lastname and the firstnames.

Project

This AML object describes a project, and is usually linked to many other AML objects

It is tempting to describe AML as a “metadata standard”, but this leads to the question of what is metadata compared to data, and the answer is not clear and absolute. Usually, data is called “metadata” when it is giving information on other data; with this definition, a “Person” AML object would be called data (because the object it is referring to is not a data file), and a “Table” AML object could be called metadata (because it is giving information on the binary file with the table content). It is not always easy to choose, though. For instance, an “Image” AML object can be seen as metadata for the binary image file, but it can also be seen as observational data, the image binary file being a part of the whole observational information. So, the meaning for “metadata” depends on the semantics, which can lead to confusion. The point of view for AML is to describe data on “objects”, sometimes with links to binary files or other objects, regardless of the “meta” level. As most scientific data use different levels of metadata, it is important to use a format handling data and metadata in the same way : some “data” can become “metadata” when some data is added to a file, and conversely.

The following features are specific and important for AML :

- *Integration of objects*

AML combines information about several distinct objects in the same format. The objects have common elements, described in a consistent way (this “common” part is explicitly separated from the other elements in a new AML DTD). Combining the different objects is very important for applications using more than one : if there was a DTD for each object, the definitions would not be very consistent, and an application would have to mix file formats transformations with data processing instructions. Another option, with independently defined objects, would be to use a common internal structure in a program or an intermediate file format, but this implies doing the transformations at the end of the chain between the source of the data and the application. Since the transformations can be difficult or impossible because of differences in the semantics, it is better to unify the data as early as possible.

- *IR/KDD features*

The links between the AML objects are important for information retrieval and knowledge discovery. These links have a role (an explicit meaning) and a title. They can be defined by URL or URN, and they can be links to virtual objects, only created on-the-fly when needed. They allow meaningful browsing of the files, as well as more elaborated data processing like the AML maps, described later in this report.

• *scientific features*
AML is representing astronomical data, and is using science-specific features with this aim in view. For instance, in the "Astronomical Object" AML object, the measurements can be written in a uniform way : the DTD defines an element for a list of measurements for a given property, and an element for a measurement, with the attributes unit, uncertainty and bibliographic reference. The same attribute names are used throughout the DTD for the same purposes, for instance the unit will always be defined by a "unit" attribute.

• *customisation possibilities*
Institutes may want to carry their own data in AML files, even if they don't have an interest for most people outside. An element "SPECIFIC" can be used in this regard, which is to be used with the namespaces standard for XML : another XML DTD can be used inside the "SPECIFIC" elements, with elements not defined in the AML DTD, without risking to loose the documents interoperability.

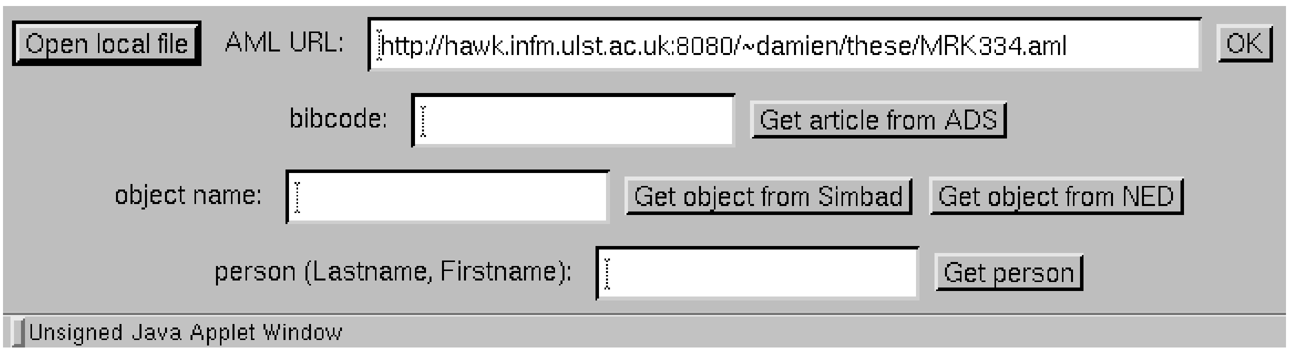
The AML browser

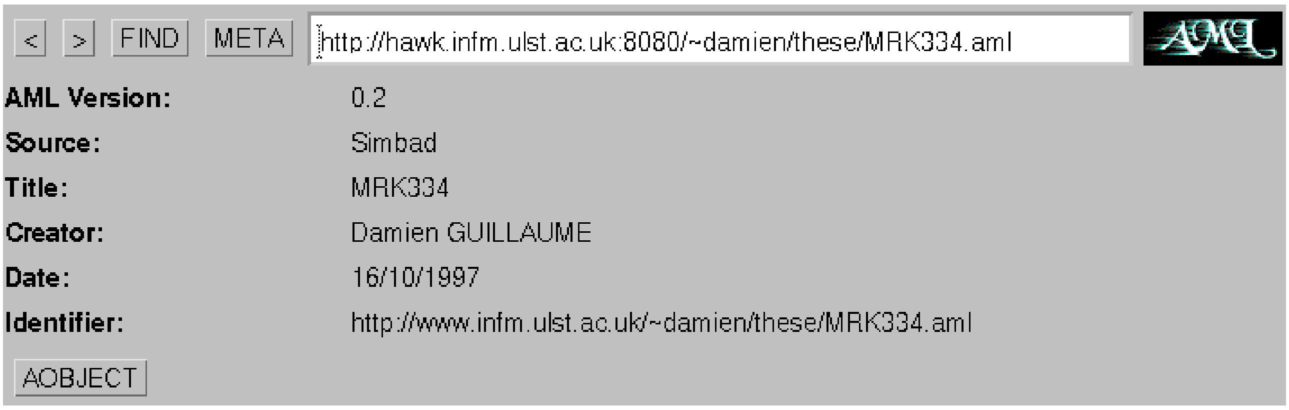
The AML browser is a Java applet to visualise and use AML documents, based on the JDK (Java Development Toolkit) 1.1.

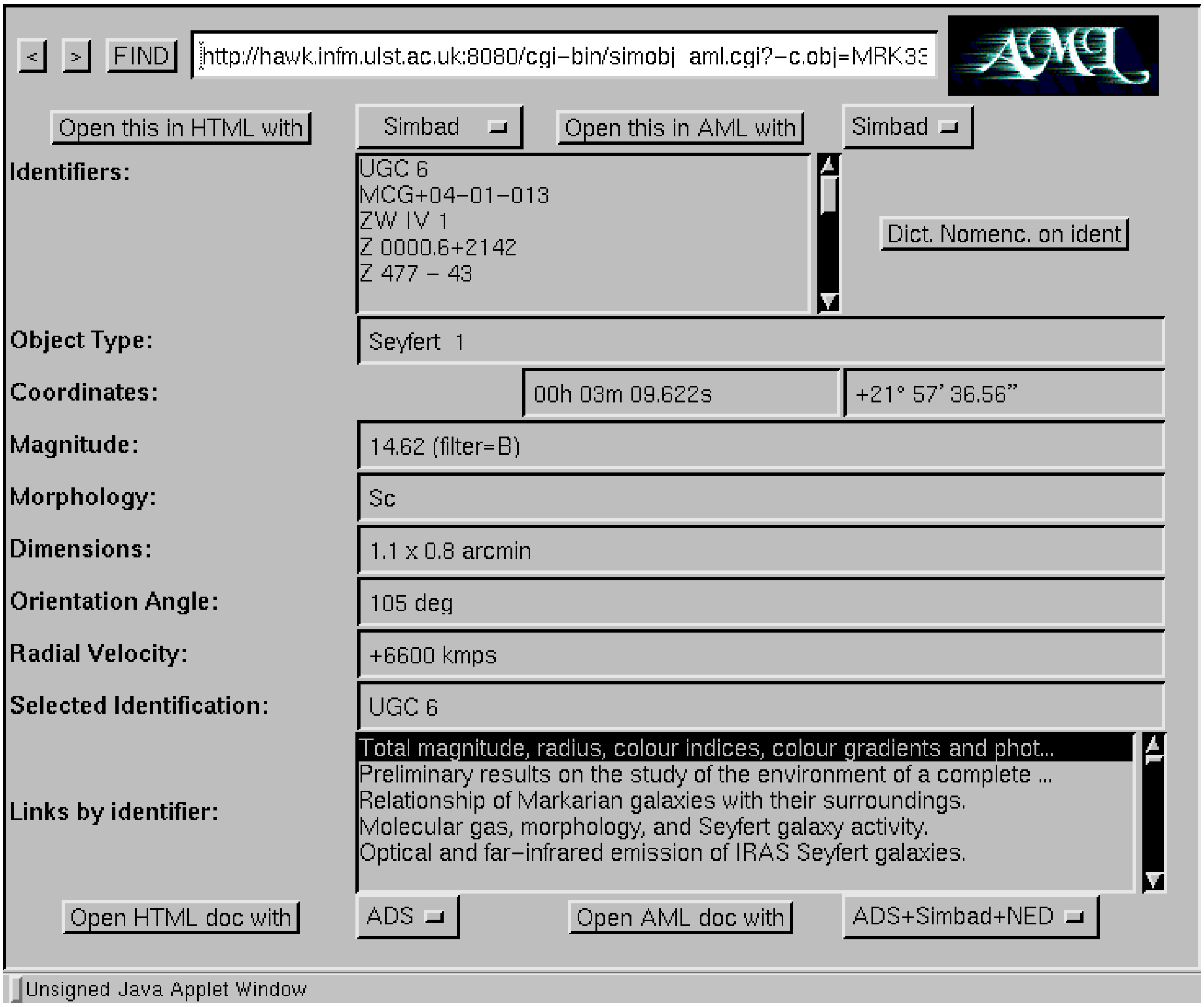
User Interface

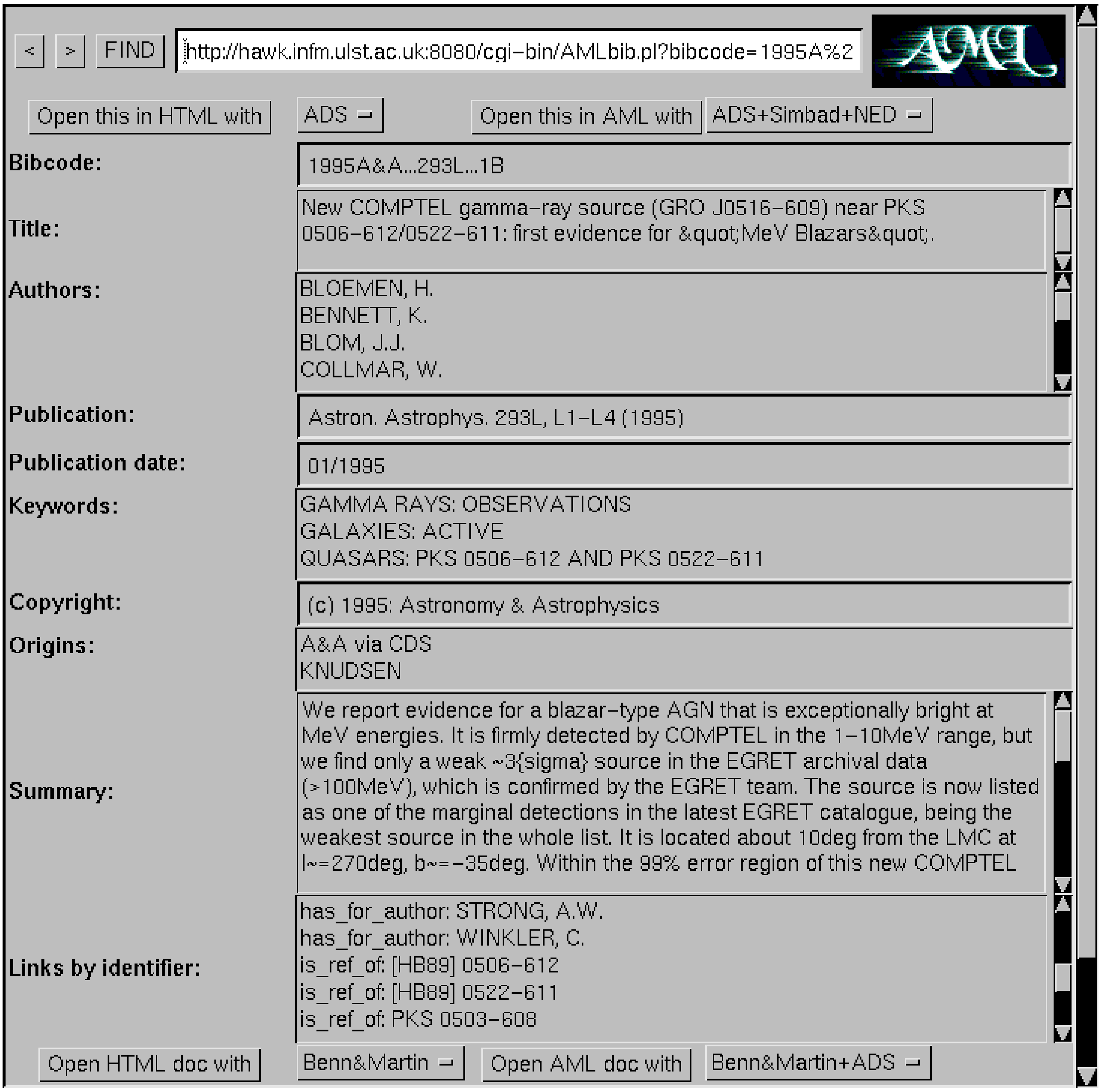
The user interface is similar to a web browser interface : the URL of the AML document is displayed on the top, with buttons to go back to the last document or move forward in the history, and a button to start a search. There is one more button, one to display the document metadata. Because HTML documents describe an interface while AML documents describe the data in itself, there can be any user interface resulting from an HTML document, but only a limited number of user interfaces from an AML document, corresponding to the different AML objects.
Currently, there are 6 different AML objects, with a different interface for each of them : astronomical object, article, person, table, set of tables, and image. With an additional interface for the metadata, and another one for the search, there is a total of 8 different pages handled by the AML browser.

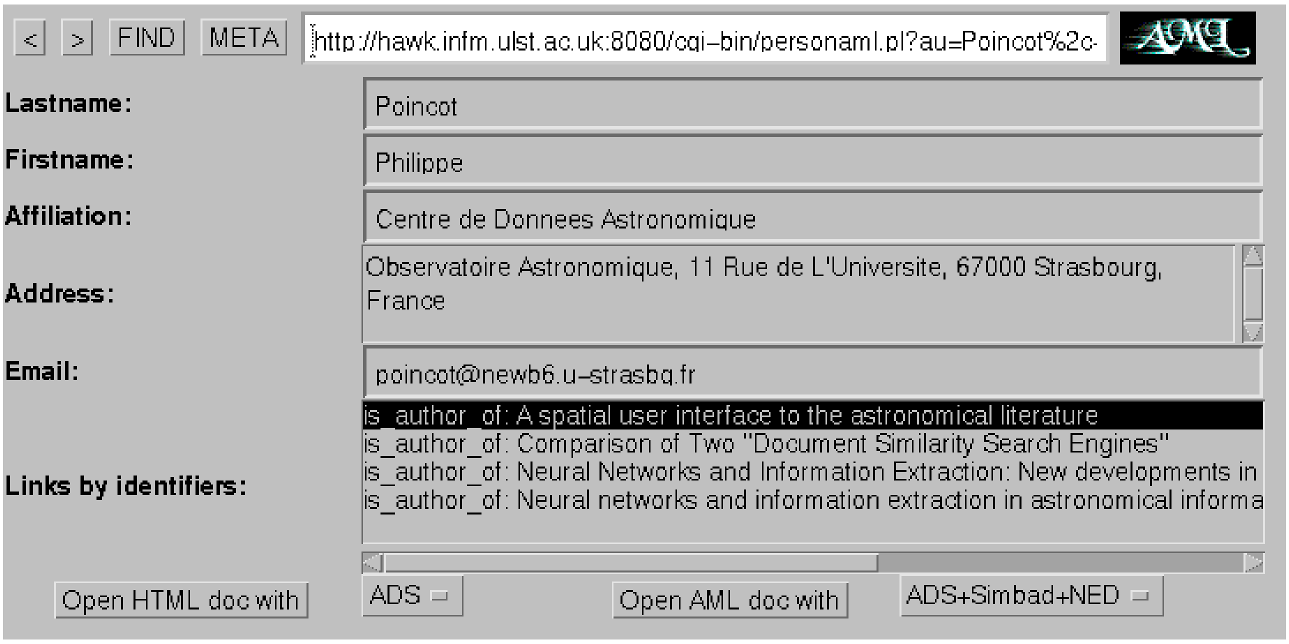
The interface is self-explanatory enough to show here the screenshots without further detail.



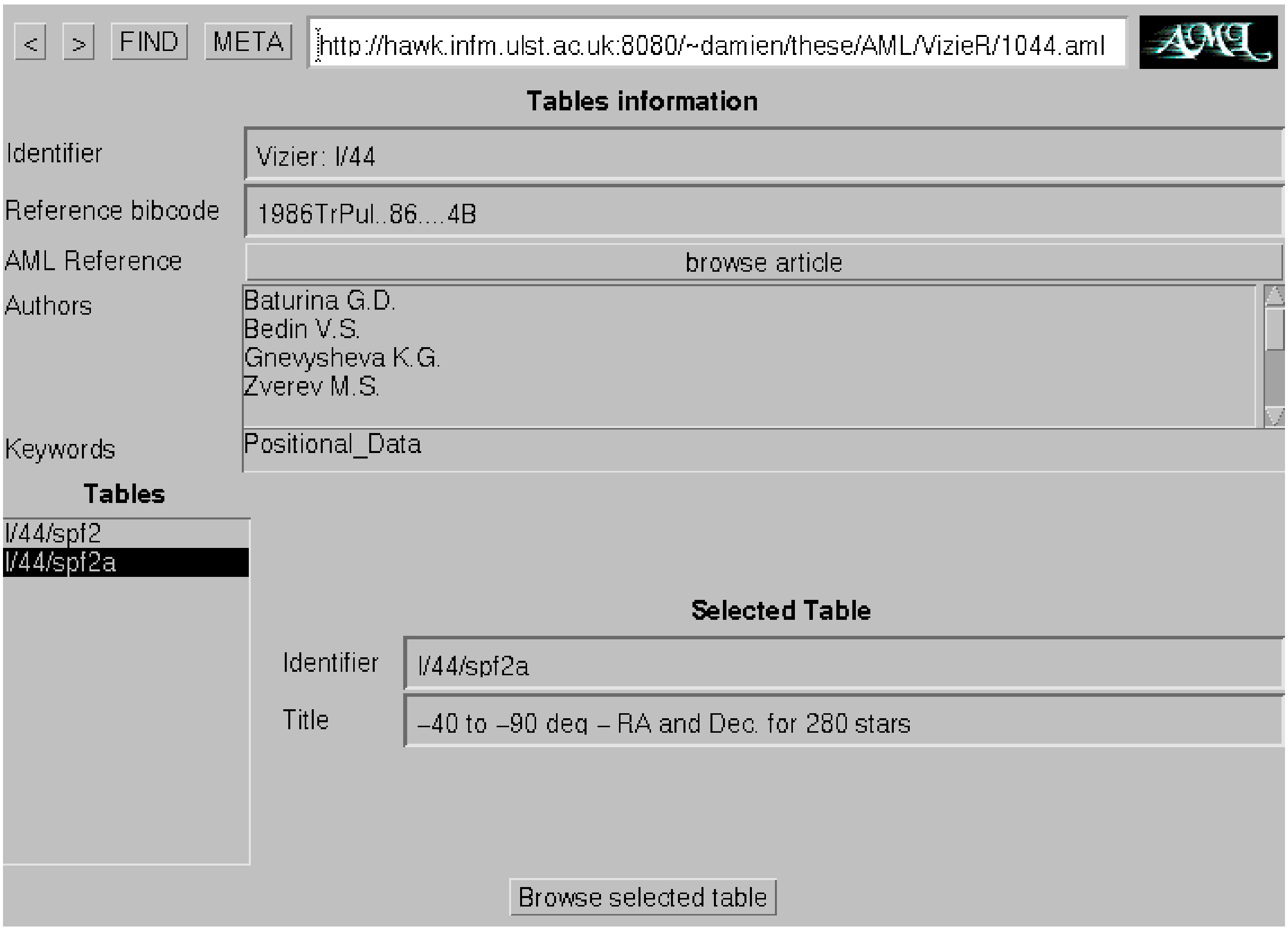

< > FIND META
http://hawk.infm.ulst.ac.uk:8080/~damien/these/MRK334.aml
AML Version: 0.2
Source: Simbad
Title: MRK334
Creator: Damien GUILLAUME
Date: 16/10/1997
Identifier: http://www.infm.ulst.ac.uk/~damien/these/MRK334.aml
AOBJECT


< > FIND
http://hawk.infm.ulst.ac.uk:8080/cgi-bin/simobj aml.cgi?-c.obj=MRK33
Open this in HTML with
Simbad
Open this in AML with
Simbad
Identifiers:
UGC 6
MCG+04-01-013
ZW IV 1
Z 0000.6+2142
Z 477 - 43
Dict. Nomenc. on ident
Object Type: Seyfert 1
Coordinates: 00h 03m 09.622s +21° 57' 36.56"
Magnitude: 14.62 (filter=B)
Morphology: Sc
Dimensions: 1.1 x 0.8 arcmin
Orientation Angle: 105 deg
Radial Velocity: +6600 kmps
Selected Identification: UGC 6
Links by identifier:
Total magnitude, radius, colour indices, colour gradients and phot...
Preliminary results on the study of the environment of a complete ...
Relationship of Markarian galaxies with their surroundings.
Molecular gas, morphology, and Seyfert galaxy activity.
Optical and far-infrared emission of IRAS Seyfert galaxies.
Open HTML doc with
ADS
Open AML doc with
ADS+Simbad+NED
Unsigned Java Applet Window

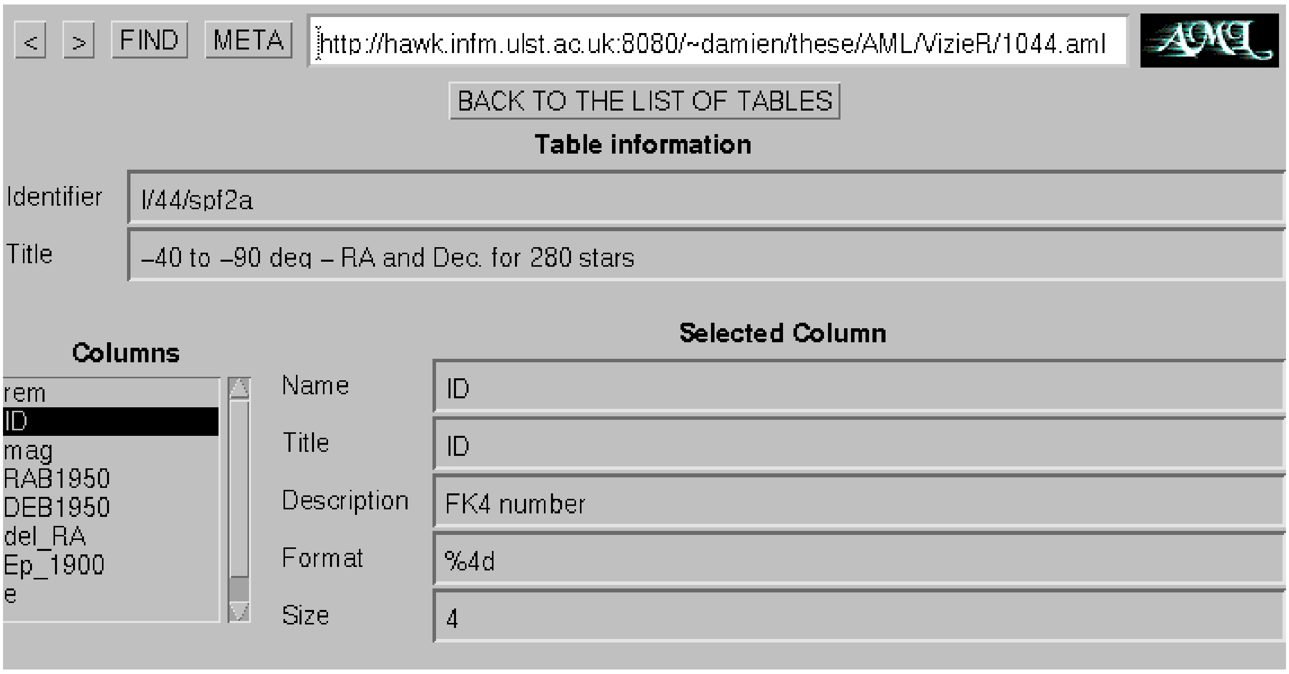

< > FIND
http://hawk.infm.ulst.ac.uk:8080/cgi-bin/AMLbib.pl?bibcode=1995A%2
Open this in HTML with
ADS
Open this in AML with
ADS+Simbad+NED
Bibcode:
1995A&A...293L...1B
Title:
New COMPTEL gamma-ray source (GRO J0516-609) near PKS 0506-612/0522-611: first evidence for "MeV Blazars".
Authors:
BLOEMEN, H.
BENNETT, K.
BLOM, J.J.
COLLMAR, W.
Publication:
Astron. Astrophys. 293L, L1-L4 (1995)
Publication date:
01/1995
Keywords:
GAMMA RAYS: OBSERVATIONS
GALAXIES: ACTIVE
QUASARS: PKS 0506-612 AND PKS 0522-611
Copyright:
(c) 1995: Astronomy & Astrophysics
Origins:
A&A via CDS
KNUDSEN
Summary:
We report evidence for a blazar-type AGN that is exceptionally bright at MeV energies. It is firmly detected by COMPTEL in the 1-10MeV range, but we find only a weak ~3{sigma} source in the EGRET archival data (>100MeV), which is confirmed by the EGRET team. The source is now listed as one of the marginal detections in the latest EGRET catalogue, being the weakest source in the whole list. It is located about 10deg from the LMC at l~=270deg, b~=-35deg. Within the 99% error region of this new COMPTEL
Links by identifier:
has_for_author: STRONG, A.W.
has_for_author: WINKLER, C.
is_ref_of: [HB89] 0506-612
is_ref_of: [HB89] 0522-611
is_ref_of: PKS 0503-608
Open HTML doc with
Benn&Martin
Open AML doc with
Benn&Martin+ADS



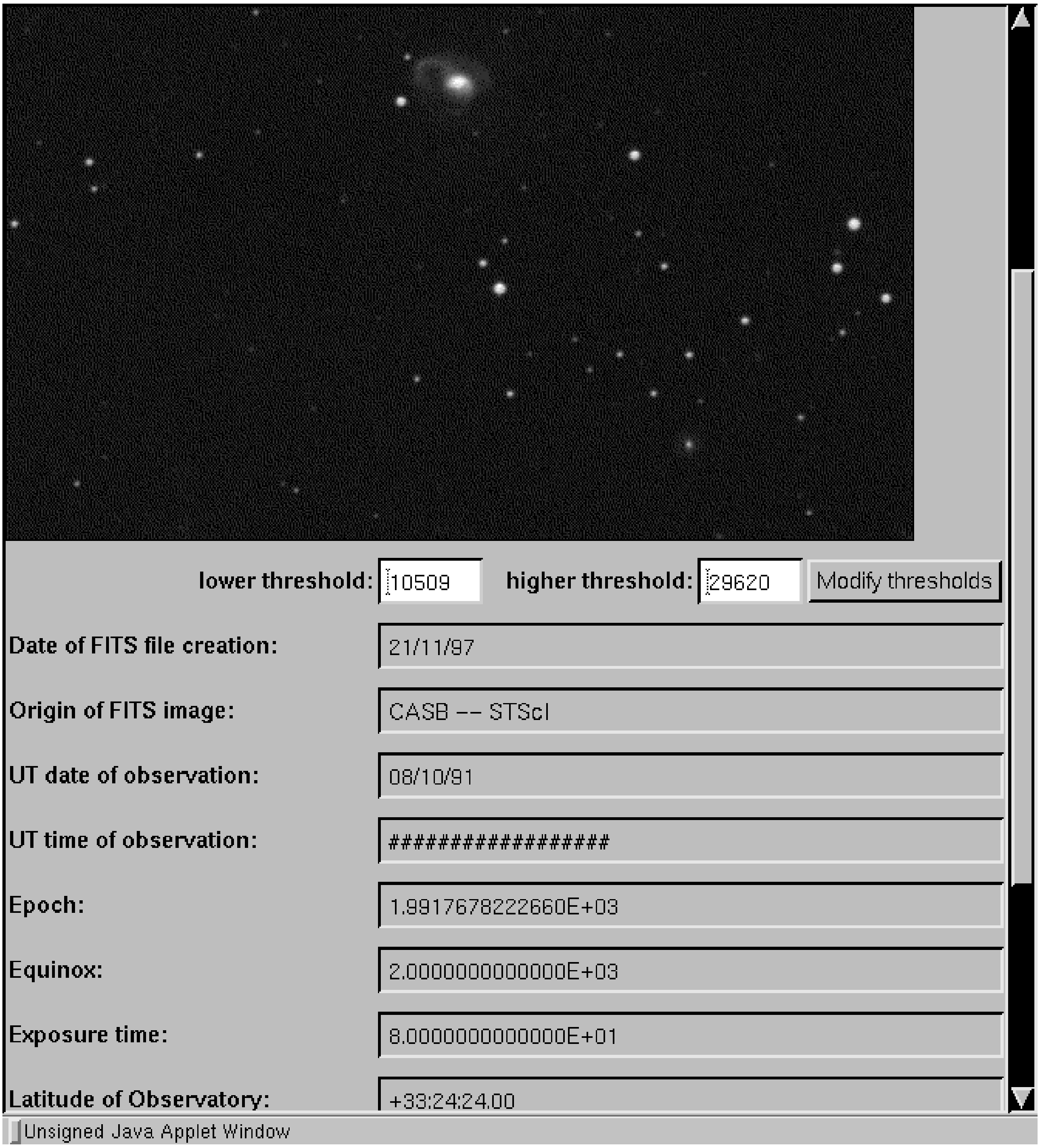
< > FIND META
http://hawk.infm.ulst.ac.uk:8080/cgi-bin/personaml.pl?au=Poincot%2c
Lastname: Poincot
Firstname: Philippe
Affiliation: Centre de Donnees Astronomique
Address: Observatoire Astronomique, 11 Rue de L'Universite, 67000 Strasbourg, France
Email: poincot@newb6.u-strasbg.fr
Links by identifiers:
is_author_of: A spatial user interface to the astronomical literature
is_author_of: Comparison of Two "Document Similarity Search Engines"
is_author_of: Neural Networks and Information Extraction: New developments in
is_author_of: Neural networks and information extraction in astronomical informa
Open HTML doc with
ADS
Open AML doc with
ADS+Simbad+NED



< > FIND META
http://hawk.infm.ulst.ac.uk:8080/~damien/these/AML/VizieR/1044.aml
Tables information
Identifier Vizier: I/44
Reference bibcode 1986TrPul..86....4B
AML Reference browse article
Authors
Baturina G.D.
Bedin V.S.
Gnevysheva K.G.
Zverev M.S.
Keywords Positional_Data
Tables
I/44/spf2
I/44/spf2a
Selected Table
Identifier I/44/spf2a
Title –40 to –90 deg – RA and Dec. for 280 stars
Browse selected table



< > FIND META
http://hawk.infm.ulst.ac.uk:8080/~damien/these/AML/VizieR/1044.aml
BACK TO THE LIST OF TABLES
Table information
Identifier
I/44/spf2a
Title
–40 to –90 deg – RA and Dec. for 280 stars
Columns
rem
ID
mag
RAB1950
DEB1950
del_RA
Ep_1900
e
Selected Column
Name
ID
Title
ID
Description
FK4 number
Format
%4d
Size
4



Implementation

The AML browser is composed of the following Java objects :

- 1 for the XML parser, from which a AML parser is derived
- 1 representing an XML tree
- 1 representing the browser frame, derived from the Frame class. It also contains the interface for a search and for displaying the metadata
- 1 derived from the Applet class, which can also be used as an application, and is used only

to open the frame

• 1 for each of the AML objects (using a common class interface). This is where the object terminology takes its sense : the whole object's data is described in the AML file, available as a tree in the Java object, and the interface is described as methods of the same Java object. This is why it is easy to create a new AML/Java object : the same interface is used for all the objects (both for the data with AML and with the programming interface with a Java interface), but the objects are clearly separated and independent.

The Java objects corresponding to the AML objects use some common elements (apart from the XML parser and the XML tree) :

• the top part of the browser, displayed by a procedure in the Frame object

• the fields of various sizes, and the lists of fields : a normal field is supposed to contain text, and a textbox with suitable dimensions is created in the interface. If an XML tag end with the letter 'S' and contains other tags with the same name without the 'S', it is considered as a list by default, and displayed as such.

• the AML links are also common for all the AML objects. They can be either URLs, or identified by the AML object type and the identifier (with the meaning of the link and a title). In the latest case a list of servers able to give the result for the given AML object is displayed as a choice for the user. A simple URN system has been created for this : given a list of servers in a configurable XML file, it can transform a URN into a URL. The URN can be composed of the AML object type, the object identifier, the output format (AML or HTML), the server name, the location (to use the closest mirror), and other parameters.

Some functions are specific to certain AML objects (and are implemented in the corresponding Java objects), for instance :

• for an astronomical object : display of astronomical coordinates

• for an image : image display and image processing (currently, only the ability to change the image thresholds; more image processing is planned)

• for a list of tables : a description of each table, and a link to each of them

• for a table : the table display (this calls a cgi program to read the rows as the user scrolls down the table). Some search functions are planned.

Integration with other software

Although this has not been done yet, a module of the AML browser could make use of CORBA for some specific tasks executed on different servers. This could be especially useful for resource-intensive processing, such as complex image processing. This system could become widespread if the software industry becomes a service-oriented system on the internet.

The AML browser is composed of different modules, easy to modify separately. But it can also be used as a single module in a bigger software. This is what has been done with the NCSA EMERGE architecture : this architecture provides a search, retrieval and display environment for scientific data. The advantages of AML for information search and retrieval are a good reason to use it within EMERGE, and the browser is called to display any AML data. This is done simply by calling a method in the browser frame class to initialise it, and a method to open a URL or a Stream.

VI. THE AML MAPS

Having finally reached the point where knowledge discovery was getting easy, it was time for an application. The AML maps gather AML documents, cluster them and display them with a nice user interface. Things that were thought impossible before, like clustering articles, authors and astronomical objects together, become quite natural from the end user point of view.

Clustering of AML documents

Discovering Knowledge by Using the Links : A Problem of Visualisation

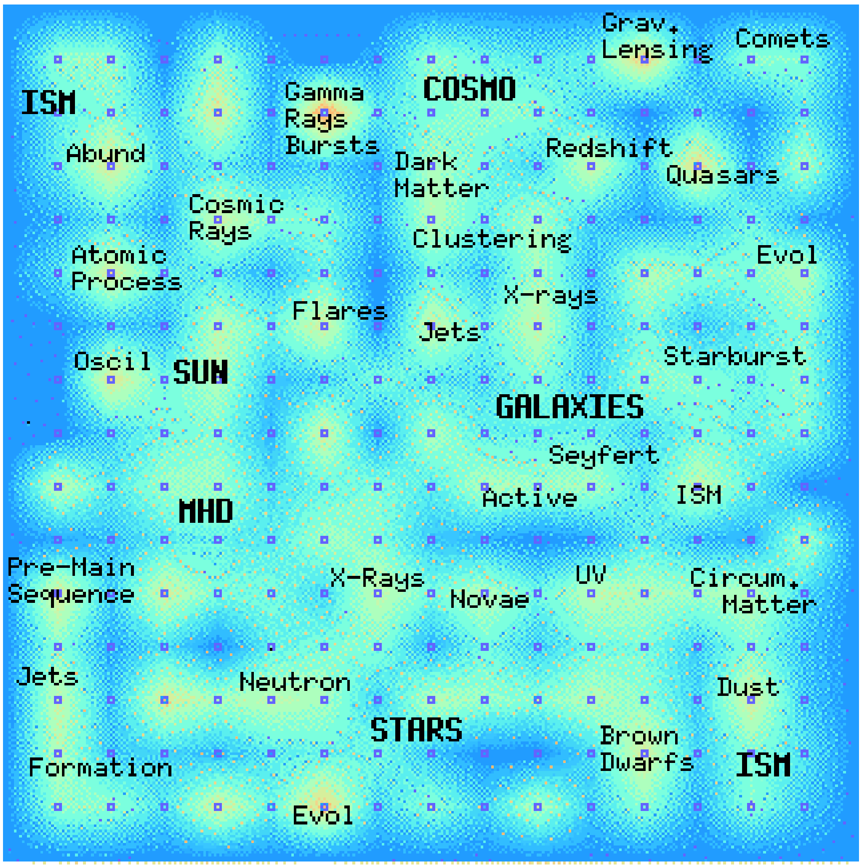
Different existing astronomical information servers make documents available for internet users, e.g. ADS, NED or Simbad, and one can use the links between the astronomical objects and the articles to surf from one server to another. However, the documents have more and more links, and when a document has hundreds of links, it becomes impossible for the user, unless (s)he is very patient, to look at all the links and to summarise the information.

It would be possible, from one document, to retrieve all the documents linked to it, and present a large rooted tree to the user. However the tree would be so big that, even if the problem of displaying it is solved, the user would become quickly lost. Also, there would be links between the retrieved documents which would be difficult to show on a tree. Some proposals have been made to display non-nested clusters, like the extended tree (EXTREE) (Corter & Tversky, 1986) or multiple trees (Carroll & Pruzansky, 1975, Carroll & Pruzansky, 1980), but they still use trees for the display and therefore are not an ideal solution for sets of many documents.

A Java applet for representing the links of HTML documents and clustering the documents was implemented in 1996 by Greg Linden, and is called Web View. This applet works well when exploring the close neighbourhood of an HTML document, but the graph is really displayed as a graph, with circles for the nodes and lines for the edges, and it is difficult to use more than 30 documents simultaneously.

Kohonen's Self-Organising Feature Maps (SOM) (Kaski, Honkela, Lagus & Kohonen, 1998) displayed as density maps are increasingly used as a convenient display solution for information retrieval. In the astronomy field in particular, Kohonen SOMs have been used to display summarised information on the content of astronomical bibliographic databases and allow the user to search for documents by subject and get similar documents at the same time (Poinçot, Lesteven & Murtagh, 1998, Poinçot, Lesteven & Murtagh, 1999).

The next density map shows clusters of similar documents, with a colour corresponding to the density of documents in each square (pixel) of the map. For each square, a title (difficult to compute automatically) indicates the subject of the documents in that region of the map.



CDS Kohonen map for the Astrophysical Journal, based on approximately 12000 documents for the years 1994-1999 (this map's opposite borders are linked : "Comets" is close to "ISM")

In this particular case, the clustering is based on the keywords associated with the articles. In the case of our work, however, we seek instead to use the relationship between the documents as expressed by the links between them.

Graph-Partitioning Algorithm and Its Visualisation

Clustering the documents using their links is a a graph-partitioning problem : we have a graph of connected nodes (the documents), with weighted edges (links) between them, and we want to partition this graph so that it can be displayed on a density map (Fig. 2).

This problem of clustering documents based on their links is more interesting with XML and XLink than with HTML, because we have weighted links, the weights corresponding to the semantics of the links : the semantics are specified with XLink but are not supported in the case of HTML.

Many proposals have been made related to this NP-hard problem (Garey & Johnson, 1979), such as the simulated annealing algorithm (Shield 1987, Johnson, Aragon, McGeoch & Sheron, 1989) (solving the local minima problem by accepting non-improving solutions with a probability based on the *temperature* which is allowed to slowly decrease in value), and noising algorithms (Sharon & Hudry, 1993) (solving the local minima problem by adding noise to the data, decreasing with the number of iterations).

A recent proposal for a noising algorithm for the graph partitioning problem by Sudhakar and Murthy (1997) provides a good solution to this partitioning problem. This noising partitioning algorithm takes a graph as the input, and returns the associations between the nodes and a given number of clusters so that a cost is minimised. We used this algorithm, which is rather fast, and gives good results. This algorithm is described in Appendix 4, “Modified noising algorithm”.

We used two different ways to get the initial documents : they can be issued from a repository of linked documents, or they can be retrieved by querying servers. In the later way, an initial query is executed to get a first document, and its links are used recursively, until a chosen depth is reached, to get all the related documents.

Two important problems the algorithm addresses are to obtain balanced clusters, and to minimise the number of cut links between the clusters. A constant (α) is used to provide a trade-off between these two constraints, so that the cost function can be expressed as : $T = I + \alpha * C$, I being the total imbalance between the clusters, and C the cost resulting from the edges cut for the vertices that are not in the same cluster (the definitions of I and C are further looked at below).

PARTITIONING THE GRAPH AND CREATING A DENSITY MAP

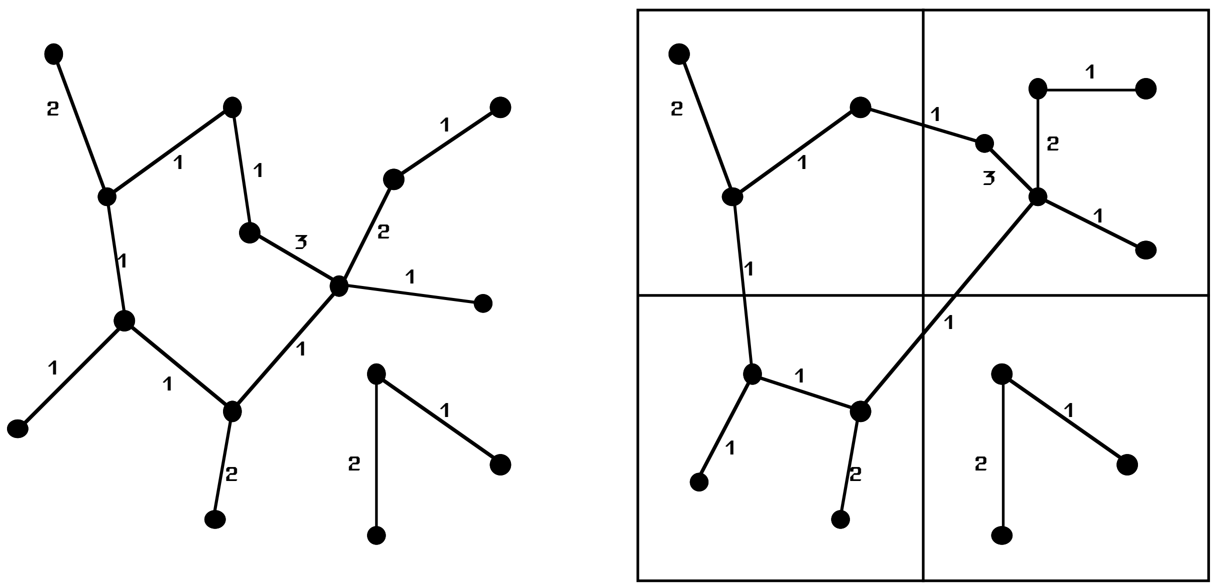


Fig. 2. Link weighted graph (left), and one partition (right).

In this particular implementation, the vertices are AML documents, and the edges are the links between them. The clusters are fixed on a grid of a given size at the beginning, and the cost C depends of the distances between the clusters :

$C = \Sigma$[for all the documents i, j]($w_{ij} * d(c(i), c(j))$)

w_{ij} being the weight of the link between documents i and j

c(i) being the cluster containing the document i

d(a, b) being the Euclidian distance, on the grid, between clusters a and b

We went through different tries before we obtained the cost function for the links mentioned above, C. Initially, we were calculating the clusters first, independently from the grid, and then we were trying to map them on the grid, using the links between the clusters for that: the noising algorithm was called a second time, with the clusters as vertices. But the result was not optimum, because another partition of the documents could have resulted in a better positioning of the clusters on the grid. The solution was to mix all the constraints together and to give them as an input to the graph-partitioning algorithm.

The original function for I was : $I = \Sigma[\text{for all the clusters } k](|L_k - L_{av}|)$

L_k being the number of documents in cluster k

L_{av} being the average number of documents per cluster

With this function for I, the result is not completely satisfactory for the user in the case where the documents are retrieved recursively from an initial one, because we get one cluster (with the first document queried) with a lot of associated documents, and all the other clusters have exactly the same number of documents. The result is interesting because of the associations we can discover, but if the cluster sizes were not so well balanced then the associations could be more relevant. If we reduce the value of α, we only get fewer documents per cluster (apart from the original one), and always the same number of documents for the other clusters. This is because of the imbalance cost function, I : the cost is increased by one if the number of documents in a cluster is increased by one, whether the current number is close to the average number or very far from it.

We improved this function, just for display reasons, to a function in which the imbalance is calculated with a square function centred on the average value :
we use $I_k=|L_k-L_{av}|^2/L_{av}$ instead of $I_k=|L_k-L_{av}|$. In this way, the imbalance cost grows quickly : it is not important if a cluster has just one more document than the average, but we want to avoid getting a cluster with no document at all, or a cluster with too many documents. The result is a nice density map with many colours, showing reasonably balanced clusters.

Ultimately, the graph-partitioning algorithm works like an "intelligent" program : it is given the declarative constraints of the problem to solve, and it is not necessary to look into the "black box" until it returns the (if not perfect, always good) result.

Titles can be chosen for the clusters, based on the links : for each document in a cluster, the sum of the weights of the links between the document and the other documents in the same cluster is computed, and the document with the best score is chosen as a title for its cluster. This might not be the most "representative" document, but it certainly is the most important document, from the clustering point of view.

Examples

Any AML objects could be used for clustering (currently : astronomical objects, articles, people, images, tables and set of tables), but we need to have access to a database giving AML documents as a result of a query. At the time of this study, the only servers supplying AML documents were : Simbad, NED and ADS. We could also use the widely-available database of astronomers created by C. Benn (La Palma) and R. Martin (Royal Greenwich Observatory, Cambridge) to get people-related information. For our prototypes, we used three different AML objects : astronomical object, article and person. We therefore have a three-dimensional space of relations between these objects.

With progress in information retrieval enabled by use of the internet, other works are available relating to collaboration between scientists : J. Grossman and P. Ion have worked on the collaboration graph for the most prolific mathematician, Paul Erdös. They report (Grossman & Ion, 1995) problems in identifying people's names, such as confusing Peter L. Erdös (also P. Erdös) with Paul Erdös, and we had the same difficulties. They suggest that every author should pick as complete an author name as possible for his or her publications and use it on every item. We would add the request for database managers to use a standard syntax, like "Lastname, Firstname1, Firstname2,..."

to specify author names and allow web robots to get the last name and the first names separately (one shouldn't forget each of the names can be composed of one or more words, and hence the space is not a good separator).

The collaboration graph Grossman and Ion use, however, contains just one type of information : authors. In our case, we add more information, and possibly more relations, with the articles and the astronomical objects.

Two map examples, for Professor Jean Heyvaerts (astronomer, Strasbourg Astronomical Observatory) and MRK334 (an astronomical object) are shown in Figs. 3 and 4. In these examples, the map is composed of 16 sets of documents. If the user clicks on a location in the map, the corresponding list of documents appears below the map, with the full title of the article in case of an article. When the user clicks on a document in the list, the content is shown in the AML browser.

Another interesting feature of this program is the selection of the documents, from the whole set of the documents retrieved, which are the most similar to the first document. The comparison is based on the content of the files, using DTD-specific functions related to the XML formatting. In Section 2, we have already touched on this aspect of added semantics in XML. For instance, the "most similar" AML document to the one representing Jean Heyvaerts is the AML document for Silvano Bonazzola, because they wrote many articles together. The most similar document for MRK334 is the one for NGC7214, because they are both Seyfert galaxies and many articles have been written about the two objects together. This is a typical "knowledge discovery" application, and it goes much further than what was previously possible with HTML : instead of comparing single words using their frequencies in the texts, it becomes possible to do complex quantitative comparisons.

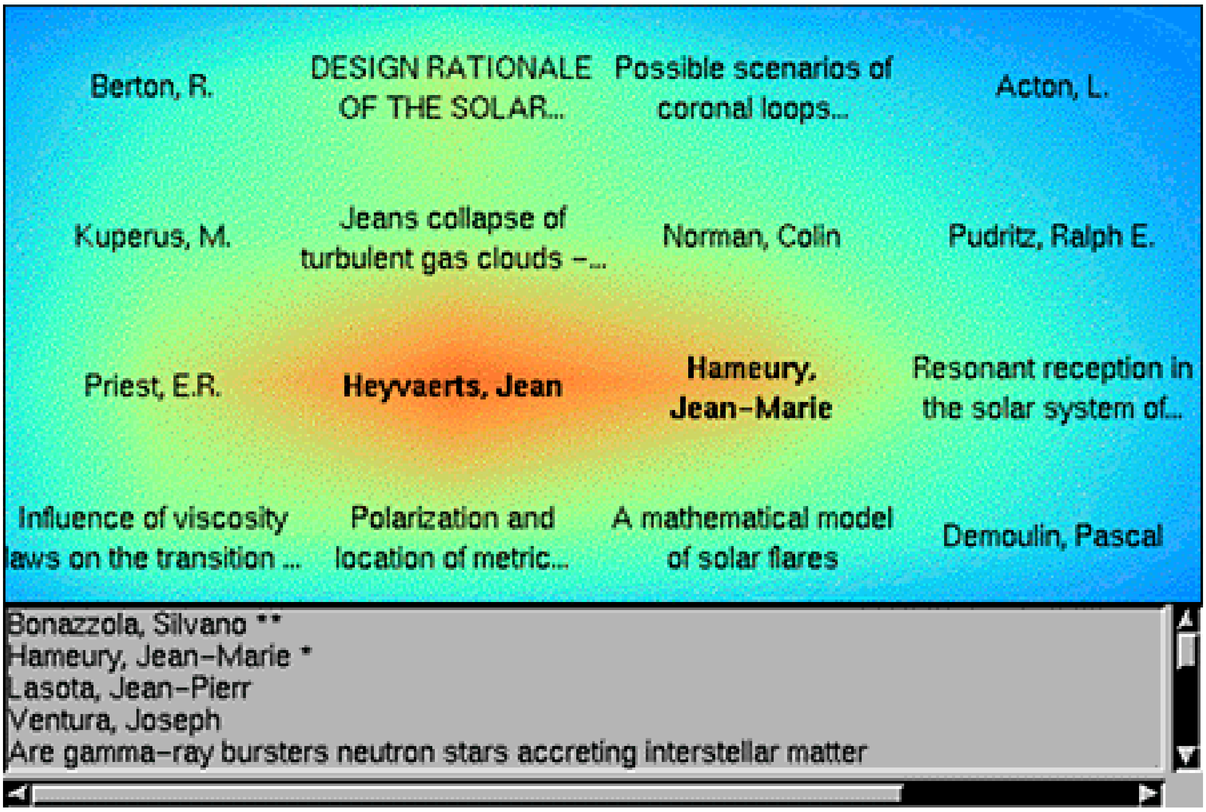


Fig. 3. Map for astronomer Jean Heyvaerts.

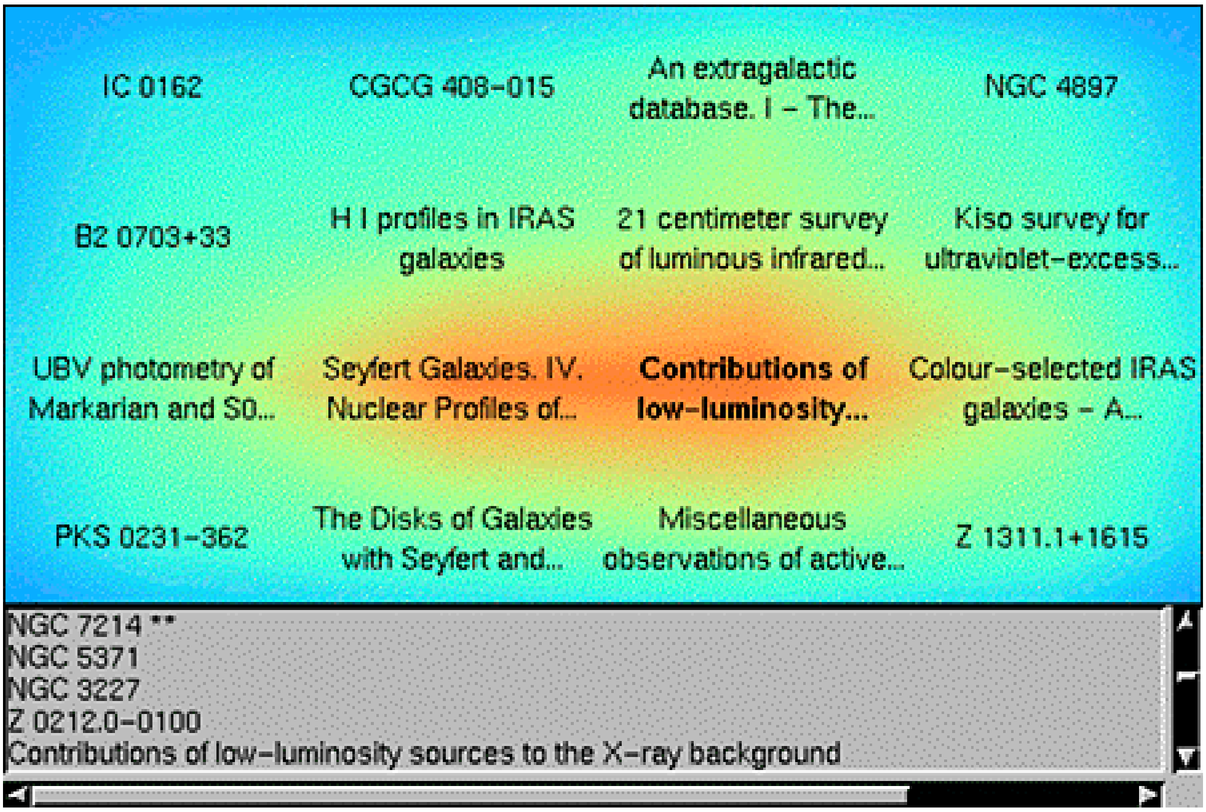


Fig. 4. Map for astronomical object MRK334.

Enhancing the Clustering by Adding Keyword Links

The prototype can be improved by availing of keywords associated with the documents : when some documents share the same keyword, new links are added between all these documents before the partitioning is carried out. These "keyword links" enhance the previous (XLink-related) links with the use of keywords associated with the graph nodes.

A simple way to implement this would be to link each document having a keyword with all the documents having the same keyword. But there would be n*(n - 1)/2 links added for n documents having the same keyword, and the partitioning would become more computationally demanding. Instead, an alternative is to create a new node for each keyword, with a null weight so that it doesn't change the balance of the clusters, and linked to all the documents having this keyword. This requires a small change in the partitioning algorithm, to allow the use of weights for the nodes (or simply a 0/1 weight).

Sometimes, previously isolated documents become part of a cluster with the use of keywords, because they share the same subject. While the maps not using the keywords are often dominated by the article/author associations, the maps using the keywords are more about the subjects in the articles. The weight of the keyword links can be changed accordingly to the desired result, based on whether the user prefers a map based on subjects or on author associations.

The clusters can be annotated with the keywords : the most representative keyword for a cluster becomes its name. To choose the best keyword for a given cluster, a score is calculated for each keyword, and the keyword with the best score becomes the cluster's annotation. The formula used is :

$$score = \frac{nb\_kwd\_docs}{total\_kwd\_docs} \cdot \left[1 - \frac{\left(f(nb\_docs\_in\_cluster - total\_kwd\_docs)\right)^2}{(nb\_docs\_in\_cluster)^2}\right]$$

$$where \;\; f(x) = x \;\; if \;\; x > 0 \;\; and \;\; f(x) = 0 \;\; otherwise$$

where *nb_kwd_docs* is the number of documents in the cluster having the keyword, *total_kwd_docs* is the total number of documents having this keyword (with the other clusters), and *nb_docs_in_cluster* is the total number of documents in the cluster.

This formula is in two multiplicative parts : the first part ensures that rare keywords are more important than very common ones, and the second part gives less weight to the keywords which are used less frequently that the number of documents in the cluster. The best possible keyword, for a given cluster, is used by all the documents in the cluster, and no other documents.

This formula can be combined with the previous one, using the links to choose a cluster annotation : the best score for the links is compared to the best score for the keywords divided by the number of documents in the cluster. If the score for the links is higher, the cluster title becomes the document title, and otherwise the best keyword is chosen. This gives a broad choice of titles for a cluster, reflecting the various meanings for the clustering, and the result is a better synthesis for the whole set of documents.

Some keywords in the articles are astronomical object names, and these keywords are not used here because the objects are already represented and taken into account by the AML documents representing the astronomical objects. This is better than doing clustering with these keywords and not using the astronomical objects as AML documents, because there can be 30 different names for one object, so the keywords using astronomical object names would not be correctly associated if they were used. Also, all the object names could not be included in the indexing vocabulary for the articles.

Ultimately, the keyword links add new organisation-related information to the map, and provide enhanced and extended meaning. This is very interesting, but two practical problems arise : many documents do not have any keyword, so they tend to be excluded from the ones having keywords, and there is currently no unified indexing vocabulary between the documents.

The first problem is partly solved by the use of the other links : the documents are always connected to others in the graph, so different clusters can have different origins and meanings. Some keywords can be extracted from other fields, such as the title or the abstract. But to extract the relevant words would require something like a list of important words.

The second problem, the indexing vocabulary, can be solved by the creation of a list of synonyms between the keywords, either by hand or automatically. Lee, Kurtz and Dubin are for instance working on this problem with a spreading activation network (Lee, Kurtz & Dubin, 1999a, Lee & Dubin, 1999b).

We have begun creating such a list by hand, and it appears that different keywords often represent different expressions of the same term, which is an issue close to problems addressed in natural language processing. But it also requires knowledge of the field of application, here astronomy. As this list is often composed of natural language words or expressions, it can also be used to extract keywords from plain text.

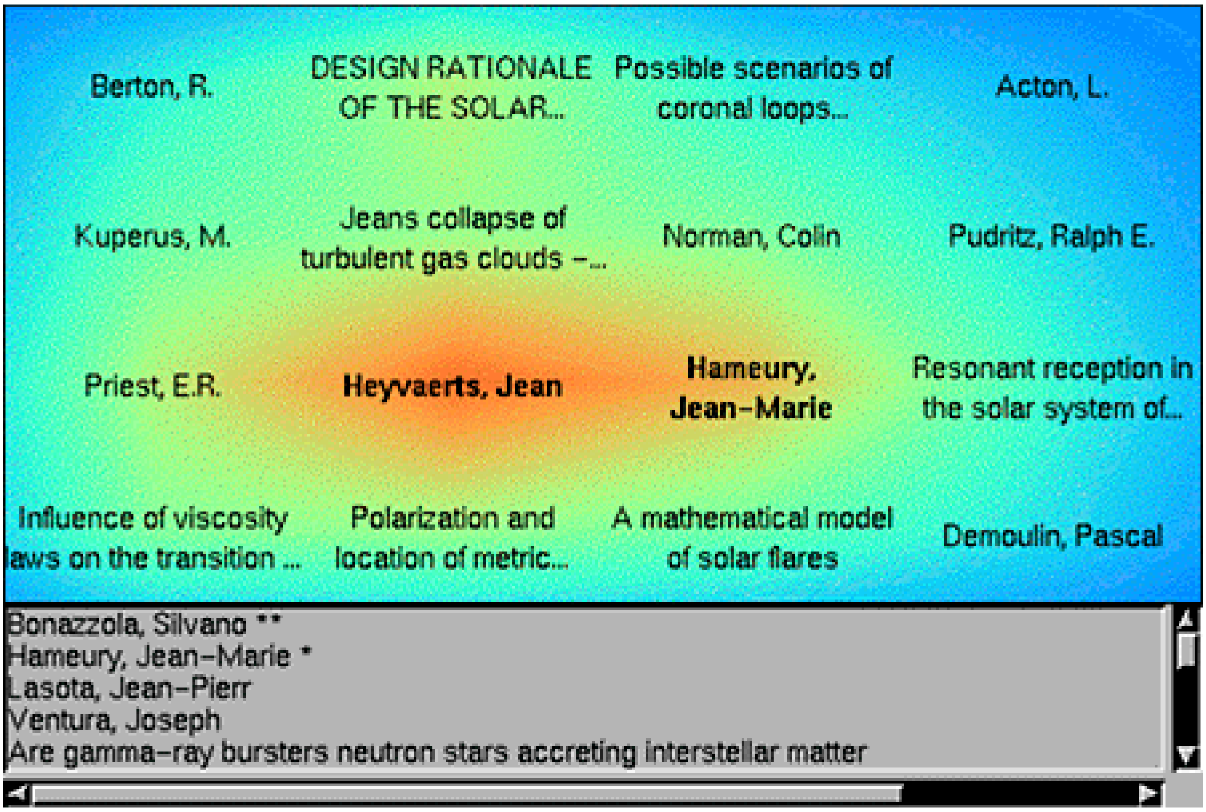


Fig. 5. The same map as Fig. 4 for Jean Heyvaerts, but with the use of keywords.

Map Implementation Issues

The time required to build a map with 65 documents can be divided into the time to download the documents, usually more than 12 minutes, and the time to do the partitioning, about 15 seconds. The first delay is due to the resources being distributed at different sites, in Europe or in the USA. The result is that the tool to build the maps cannot be used in real-time, and this is an issue because it would be very useful to get such maps instantly.

Different solutions can be considered to reduce access time :

- a site gathering together all the servers, using mirrors, and local processing of the documents,
- an index for all the servers, with only information about the links between the documents on all the servers : this solution is good only if the size of the links is much smaller than the size of the contents of the documents, and
- the use of threads to query for multiple documents at the same time.

We have tried to use threads, and this reduced the downloading time by a factor of 3 : we have found that a minimum of 4 minutes is required to download 65 documents, which is much better, but is still far from allowing real interactivity. This technique is limited by the various bandwidths, by the servers and by the local machine : after a certain number of threads, about 5 for us, the time to download the documents is increasing again.

The clustering time is very low when only the XML links are used, because the relationships matrix is sparse. But when a few thousand articles are used, with their keywords, it can take 15 hours to do a clustering of a similar quality. This is due to the keywords used in most documents : they create a huge number of new links, and the algorithm is not well adapted to this type of graph. In this case, the time to download the documents does not dominate the processing time.

AML maps : comparison with other IR and KDD systems

To compare the AML maps with other systems, we choose the case of astronomical articles. The AML maps can mix various kinds of documents together, for example about articles, authors and astronomical objects, but the other systems only allow a search within a set of documents of the same type, with a list of documents of another type linked to a found document.

Four different systems are studied here : the ADS search for articles by keywords (or any search system by keywords), the Kohonen map for astronomical documents (it has been created with A&A articles and ApJ articles, separately) , the AML map created with a collection of AML documents (like A&A articles), and the AML map created by using the links from a given starting document, recursively.

Four different tasks for these systems are studied : a simple search of a known document, a search for similar documents, a search for documents on given subject, and the exploration of the “surroundings” of a given document.

Search by keywords

A search by keywords, or by field (title / authors / keywords) is especially useful if the document is known : either one of the authors is known, or part of the title. In this case, the document can be found immediately with any of the different systems (they all provide a more or less good search by keywords), provided it is contained in the indexed collection. So, in this case, the most important is to have the biggest collection of documents, and this is easier if the search is only by keywords : the Kohonen map or the AML map are usually not created with more than about 10000 documents, because it requires too much time to create the maps. An indexed collection of documents for a search by keyword has no such limitation.

A similar document can be found by looking for documents with the same keywords as the original document. This is however limited by the syntax of the keywords : if two documents have two different keywords with a similar meaning, these keywords will not contribute to the connection between the two documents (unless a complete dictionary of synonyms is used).

A subject can be searched by keywords if the user knows the exact keywords for the subject. The user usually knows a good keyword for its subject, but will not see at all documents with similar subject, or documents on the same subject but with the keywords expressed in a different way. Having a rigorous thesaurus for the keywords helps a lot if the user knows the thesaurus, because the user can then check the thesaurus to avoid forgetting keywords of possible interest.

Exploring the “surroundings” of a document (with the documents linked to documents linked to the original document, like the articles written by the original article’s author for instance) requires a lot of work for the user, because the links presented to the user are always of depth 1, for instance the authors of an article, or the articles written by an author. After a lot of browsing, the user can create a synthesis of the various documents linked to one, to get a global view. In practice, nobody does this work completely. This work could however be done automatically with the following systems.

Kohonen map

A Kohonen map is the result of a clustering executed by a specific neural network. The neural network is unsupervised, which means the clustering is realised without any human interaction : the result is a self-organising map (SOM). As an input for the neural network is a vector for each document. The result is a list of clusters, organised on a 2-D grid, containing documents with similar vectors. There are semantic associations between the documents and the clusters, so a Kohonen map is also a semantic map. A user interface is linked to the resulting map, with different colours for each cluster depending on the documents density, and titles for semantic regions on the map.

- *Search for similar documents*
Looking for similar documents is immediate with a Kohonen map : the documents are clustered together, so it is actually difficult not to see the similar documents. But the similar documents found on a Kohonen map are not the same than with the previous technique : new documents can be found because they have the same keywords as a document similar to the first one, and other documents are not found any more because they are "closer" to other documents somewhere else on the map. An advantage with this technique is that the similar documents are already clustered, so the user can look at an interesting cluster near a document, and forget about another cluster, even though the two clusters are equally close to the first document.

- *Search for documents on a given subject*
With a simple Kohonen map, with titles for the clusters, one can look for a cluster on a given subject, or, if the subject is not a title on the map, for a cluster on a related subject. A possible problem occurs when a subject has not been selected as an important one for the clustering, and the documents are split among various clusters at different locations. An interface like the one developed by Philippe Poinçot can however solve this problem : it is possible to search documents by keywords first, then look at the clusters concerned (the colours of the map are changed to reflect a density based on the selected documents only), and add other keywords chosen in the keywords used by the selected clusters.

- *Exploration of the surroundings*
With the map auto-organisation, it is easy to explore the surroundings of a given document. But this is limited to the documents clustered in the map, which must be represented with vectors. Usually, this restricts the set of documents to documents of the same type, like articles (and the keywords are used to create the vectors).

AML map on a collection of documents

An "AML map" looks like a Kohonen map, but is using other resources to create the map, and another algorithm : the clustering is based on links between the documents instead of a similarity between vectors representing the documents. The keywords can also be used, but it requires a transformation into "keyword links", so that the clustering is using only the links (with various weights). Although it is called "AML map" here, it could be used with any kind of document (not only AML or XML), provided links can be expressed between the documents. With XML documents, the links are explicitly expressed with the XLink syntax, so clustering using the links is natural when searching through a set of XML documents.

- *Search for similar documents*
This is like with the Kohonen map, except that other similar documents can be found because of the links between the documents : for instance, with a set of AML documents with articles, authors and astronomical objects, two articles with the same author, or citing the same astronomical object (regardless of the name possibly given as a keyword for the article), will be more likely to be

considered “similar” by the AML map algorithm. An astronomical object is a good example of something difficult to translate into a keyword, because it has usually many names, or “identifiers”, and different authors refer to the same object with different names.

- Search for documents on a given subject / Exploration of the surroundings
Again, this is like the Kohonen map, except that a “subject” becomes here a larger notion : it is no longer only related to a sample of keywords. For instance, a group of authors and articles can be clustered together because the authors wrote these articles together, even if the articles are on different subjects. A title for a cluster can be either a representative keyword for the documents in the cluster, or the name of an important document in the cluster (with many links to other documents in the cluster).

AML map from a given document

With this method, all the documents linked to one are retrieved automatically, until a certain depth is reached in the tree. Then, the same clustering as before is executed, except that a comparison is made for each document retrieved with the first document. The 2 most similar documents are flagged on the map, so that the user can see them quickly. Because the comparison is using the knowledge of AML, with specific comparison procedures, it is much more precise than a simple comparison by keywords. The set of documents is closely linked to the first one, with the AML links, and there are advantages and drawbacks : some documents very relevant can be found by this way, but other relevant documents which are on the same subject but are not directly linked will be forgotten. This is why it is complementary with the AML maps on a collection of documents (or the Kohonen maps) : the “collection of documents” can be all the articles written for a journal, thus giving a global view complementary with the local and in-depth view given by the AML map on a specific document.
The AML map from a given document is of course the best way to explore the surroundings of a given document, because it is designed for this specific task.

Comparison between the Kohonen map and the AML map

The Kohonen map uses a vector corresponding to each document and tries to associate documents with close vectors, whereas the AML map uses links between the documents to partition the graph of documents into equilibrated clusters with a minimum “cost” for the links cut between the clusters. The two problems are mathematically equivalent : the result is the same (a grid of clusters with the documents), and it is possible to translate a problem’s data from one algorithm to another.

If there is a collection of documents represented by vectors, it is for instance possible to create a fake document for each element of the vectors, linked to all the documents having this element non-null in their vectors. The fake documents can be given a weight of zero, and the links’ weights depend on the values of the corresponding vectors elements. The fake documents are deleted after the partitioning to get a clean result.

Inversely, from a collection of linked documents, it is possible to create associated vectors for the documents by adding a new term in the vectors for each link, with a non-null value only for the two documents linked.

In practice, though, the Kohonen map algorithm is best suited when the input data is composed of vectors with a small dimension, and the AML map algorithm is better when the input data is expressed easily with links between the documents, and not too many links per document (the matrix is big but hollow).

The big advantage of the AML map algorithm is that the cost function is declared explicitly, and hence the result is sure to be what we want (although the parameter balancing the imbalance cost between the clusters and the cost of the cut links has to be “tuned”). Also, it is easier to choose the trade-off between the computing time and the result’s quality (it is a simple parameter) : because of this, a very precise result can probably more easily be found with the AML map algorithm, although it might require a lot of CPU.

Combining both techniques

It is sometimes interesting to use both the AML links and the keywords in the documents, for instance to create a map of articles, authors and astronomical objects and taking into account the article keywords.

This has been done with an AML map. The most common keywords, although greatly slowing the partitioning, are useful for the global structure resulting, while the least used keywords and the AML links are creating the low-scale structure of the map.

A solution, to keep the global structure resulting of the most common keywords without slowing down the partitioning, would be to use a Kohonen map first, with the most common keywords, and to create an AML map starting from this result, without the most common keywords. Usually, an AML map starts from a random solution, and tries to enhance it and avoid the local minima. But it is difficult not to fall in a local minimum for the global structure with this algorithm; by using a Kohonen map first to create this global structure and enhance it locally, we would assure that both the low-scale and the high-scale structures are preserved.

Conclusion

The different systems studied here are able to solve different kinds of search, and the best choice depends on the search's goal :

- for a quick search of a known document, or on a well-defined subject, the best choice is a simple search by keywords, with a system like ADS. Because keyword searches are widespread on the Internet, the user interface is easily understood by any user.

- for a quick exploration on a subject, it is better to use a Kohonen map or an AML map on a big set of documents (it is supposed that the map has already been created, and is ready for browsing). The user interface with a semantic map gives an overview of the set of documents.

- for an in-depth search related to a given document, an AML map created from the document is best suited. It requires to start a query and wait much longer for the result, but in the end it is a work usually executed "manually" that is done automatically : the result is close a human synthesis.

Evaluation of the classification

The Alliance project

The Alliance project is a collaboration between a number of people in Belfast, Strasbourg and Toulouse, to evaluate different methods of classification, with financial support of the British Council. Philippe Poinçot, in Strasbourg, uses a Kohonen map to classify documents, while the Toulouse group at the IRIT institute (Josianne Mothe, Saïd Karouach and Ludovic Fontaine) uses a method based on vector comparisons in a system called Tétralogie (Mothe, Egret, Dkaki & Dousset, 1998, IRIT 1999).

Although the testing is not complete yet, we have defined the method for the tests, and we have executed the classifications on our different systems with the same dataset.

The common denominator

The common denominator

The user interface, allowing a browsing of the documents in the Kohonen map and in the

AML map, is an important part of these systems. However, it is not an objective for the algorithm used in Tétralogie, so we have not kept this element for the evaluation.

Also, the classifications for Tétralogie and the Kohonen map are based only on vectors of keywords, while the AML map is based on the links between the documents. Although the keywords can be used as links between the documents during the classification process, the list of keywords is not stored for a given cluster in the AML map. So, an evaluation for the results using the keywords is only applicable to the Kohonen map and Tétralogie.

Objectives of the evaluation

We had two objectives in mind for the evaluation of the three systems. The first one was to gain a better understanding of the classification methods, and their common points and differences. Many ideas are common to all classification systems, and are applicable to all or several of them, while the differences are useful to understanding what is unique in our systems.

The second one was to evaluate the performance of the classifications with various criteria. As the classifications are complex processing, a comparison with other classifications are a convenient way to evaluate the performance : even though the quality of the result for one classification could be discussed because of the many factors linked to the classification process, we can be sure of the quality of the relative results.

The evaluation method

The evaluation method we have chosen is based on human tests. A number of experts in astronomy have to evaluate the results of the classification, with some criteria. The only different part between the systems, for these tests, is the classification method : a new simple user interface has been created solely in order to provide a common environment for the tests. This means that only a part of the objectives, for some systems, is evaluated : there is no “browsing” allowed here.

Other evaluation methods have been used by other researchers to evaluate classifications. Some of them imply many classifications based on the same dataset. This was not possible for us because of hardware limitations : for the chosen dataset, the AML map and Tétralogie had to compute the classification during hours (the Kohonen map was much faster), so it was not feasible to compute hundreds of different classifications.

Common elements : dataset and thesaurus

We have been very careful to use the same dataset and thesaurus, so that comparisons of the results are made easier. The dataset we chose is composed of 3248 articles published in various astronomy journals for a ten-years period, written by French authors and which had the most citations during the this period. We made sure that the dataset was very consistent, and that our systems could read and interpret it in the same way.

We had two different thesaurus for astronomy at the beginning : one with a list of keywords used for Tétralogie, and another one for the AML maps, with another list of keywords, with synonyms for most of them. In order to improve the performance of our systems and to use the same thesaurus, we have unified the two thesauri, which resulted in a list of 1982 keywords with 1358 different meanings. As the Kohonen map system was not using a thesaurus, we have transformed the dataset with it instead.

A different choice : keeping the rare keywords

An interesting difference in the different systems is the choice to keep or not the rare keywords. The reasons to remove the rare keywords are the following :

- a cluster on a general theme can be split because one of its elements has a rare keyword and gets associated with another document in another cluster
- clusters become easier to understand with only keywords of their “size” : keywords used by about the number of documents in a cluster

But, keeping the rare keywords :

- doesn’t change the general structure of the classification (the number of rare keywords is high, but they are not used a lot)
- lets the algorithm use all the information (if needed, the rare keywords can be given a lower weight) (this is the representational perspective)
- results in a more precise classification : two very similar documents have a better chance to be in the same cluster

As we thought that our systems might react differently for this choice, the Toulouse and Strasbourg groups have decided to remove the rare keywords, while I preferred to keep them. From a discrimination based perspective, keeping or not the rare keywords results in different structures being highlighted, so the best choice may depend on the test set, and on the precise goals of the classification.

Tests and explanations to the testers

The following tasks have been defined for the evaluation :

Task 1 : search for similar documents

1. Each expert chooses a document in the field of expertise, without using a classification or a user interface specific to a system
2. Each system returns the list of documents in the cluster containing the first article
3. The union of the lists returned by the systems is presented to the expert for evaluation
4. The similarity of each document with the first one is evaluated
5. The results are interpreted in terms of recall and precision

Task 2 : clusters consistency

1. Each expert defines a theme for his cluster selected during Task 1
2. Each document’s subject similarity to the theme is evaluated
3. The results are interpreted in terms of precision only

Task 3 : terms consistency in a cluster (only for Kohonen maps and Tétralogie)

1. For the cluster selected during Task 1, each expert evaluates the consistency based on the theme defined during Task 2 and the document keywords

Results

Unfortunately, we have not gathered enough results yet to get statistically meaningful results.

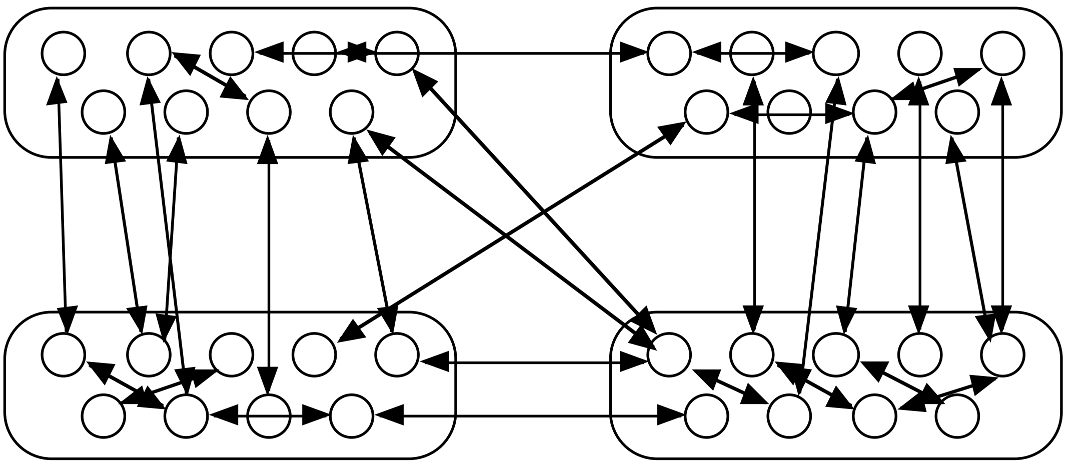
Is distributed partitioning possible ?

This section evaluates the possibility to create a graph partition of a number of distributed resources over a network, with a distributed algorithm. The resources are considered as the vertices of the graph, and the links between the resources are represented as the edges of the graph. Because the resources are distributed on many servers, the whole graph should never be created anywhere, and this is an important restriction for the resolution of the problem. Each resource is allowed to have a local view of the graph, and to communicate with its neighbours (resources having a link with it). To be efficient, the algorithm should restrain the communications between the resource repositories, because the network speed is the main limitation to solve the problem.

If an efficient algorithm could be found to partition the graph in a distributed way, it could be used on a huge scale, which would be very interesting for the many linked resources on the Internet. There is already a “web” of HTML resources on the Internet, but this web will become increasingly complex and meaningful with the generalised use of XML. A partitioning of the web could then be used, for instance, to create directories automatically with a much bigger scale than what can be done manually.

At this point, the centralist approach to this problem shouldn't be moved aside, even if it requires more and more power for supercomputers with the exponential growth of the web. The distributed approach and the centralist one are concurrent on some points, but have their own advantages and drawbacks, and they are complementary in some other cases.

The resources are stored in repositories, and the communication between resources in the same repository is not an issue, but they are supposed here to have mainly links to resources external to their repository. An example of this situation is, in astronomy, a server providing information on astronomical objects and another one specialised in astronomical articles : there are many links between the two repositories, and not many inside one repository (actually, when ADS, the main server of astronomical articles, will have information on the article references available, there will be more links between the articles, but this constraint is already realistic for today’s web).



Repositories and resources

Another reasonable assumption can be made about the repositories : each of them contains many resources. This means that communications between the repositories would not hinder the performance of the problem’s resolution, as long as the data transferred does not concern individual resources. However, it is very unlikely that a given cluster will be entirely contained by a given repository. To manage a cluster, it would be possible to associate a cluster with a repository, even if 90% of the cluster documents do not reside in this cluster.

An approach to distributed partitioning

An approach to this problem is evaluated here, based on the noising partitioning algorithm. The idea is to solve the problem with the same algorithm, but taking into account the fact that the resources are distributed : the communications between the repositories are minimised.

To process the partitioning, each repository needs to have a piece of software to manage the partitioning at a local level. A special server will coordinate the whole process, and this server must have a list of the repositories.

This algorithm is based on bidirectional links. If the links are unidirectional, as with most HTML documents, it is possible, at the beginning, to create bidirectional links from the existing unidirectional links : all the repositories would have to collect their resources' links to other repositories, and transmit the information to them.

At each iteration in the noising partitioning algorithm, there is a temporary solution, with a number of clusters (the first partition can be obtained randomly).

Each repository keeps track of the following information :

- the information for the clusters "centred" at a resource stored in this repository
- the number of resources for all the clusters

Each cluster keeps track of the following information :

- the number of resources in it
- a complete list of the links of its resources, with their weights

Cost evaluation

The noising partitioning algorithm tries to minimise the total cost I+alpha*C, of the imbalance $I_k=|L_k-L_{av}|$ for each cluster k, and of the cut links C_k for a cluster k.

For each cluster, recorded in a repository containing one of its members, the number of nodes is calculated, as well as alpha*(the number of cut links). Then, all the results (C_k and L_k) are collected by a server controlling the partitioning process by sending messages to all the repositories, the total number of resources is then calculated, and I_k for all the cluster, and the total cost.

After this first evaluation, the total number of nodes can be used to calculate I_k directly in the repositories.

The *descent* procedure has to calculate, for each node move, the difference created in the total imbalance between the clusters, and the difference in the cost of the cut links. This means that the list of the clusters, their number of nodes and their position should be known everywhere so that the cost difference can be calculated quickly.

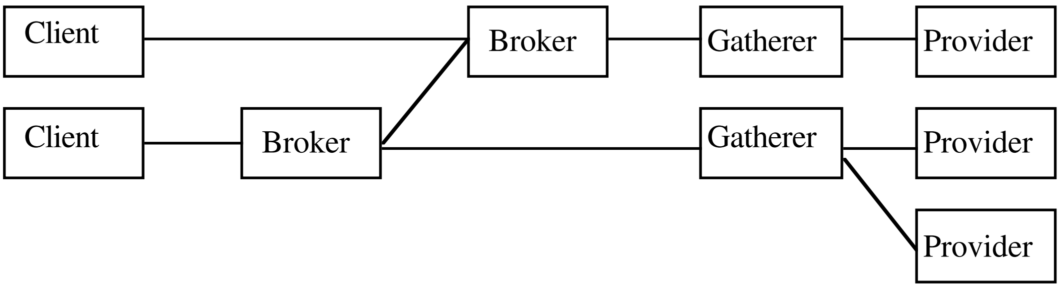
The problem

The algorithms to partition a graph are focused on the clusters, but the clusters do not relate with the repositories : they are composed of resources stored in different repositories. This means it is needed to store temporary solutions (nodes and links) in repositories where the information does not origin. The links between the resources are sometimes the biggest information in a database : for instance, the literature references to the astronomical objects sometimes take more disk space than the information on the objects. A distributed partitioning algorithm would move this information from one site to another a number of times, creating a huge overload on the network and slowing down the partitioning compared to a centralised solution. There is no interest to store all the information on different servers, far one from another with limited connection speed between them, so, a distributed partitioning is probably not a good solution to this problem, unless different assumptions are assumed.

VII. SEARCH FRAMEWORKS

Different frameworks have been proposed to let users search through internet documents. Two major issues are the number and the diversity of documents, and different frameworks have different solutions to solve these problems. Three frameworks have been studied during the thesis, and they are described here : Harvest, Ingrid and Emerge. Harvest has been created at the university of Colorado, Ingrid at NTT Labs, and Emerge at the NCSA. All three are distributed systems, in order to improve their scalability. Harvest and Ingrid are indexing resources to allow a quick access to the fields important for a query, while Emerge only transmits a query to the different servers, with the Z39.50 protocol, to allow more complex queries.

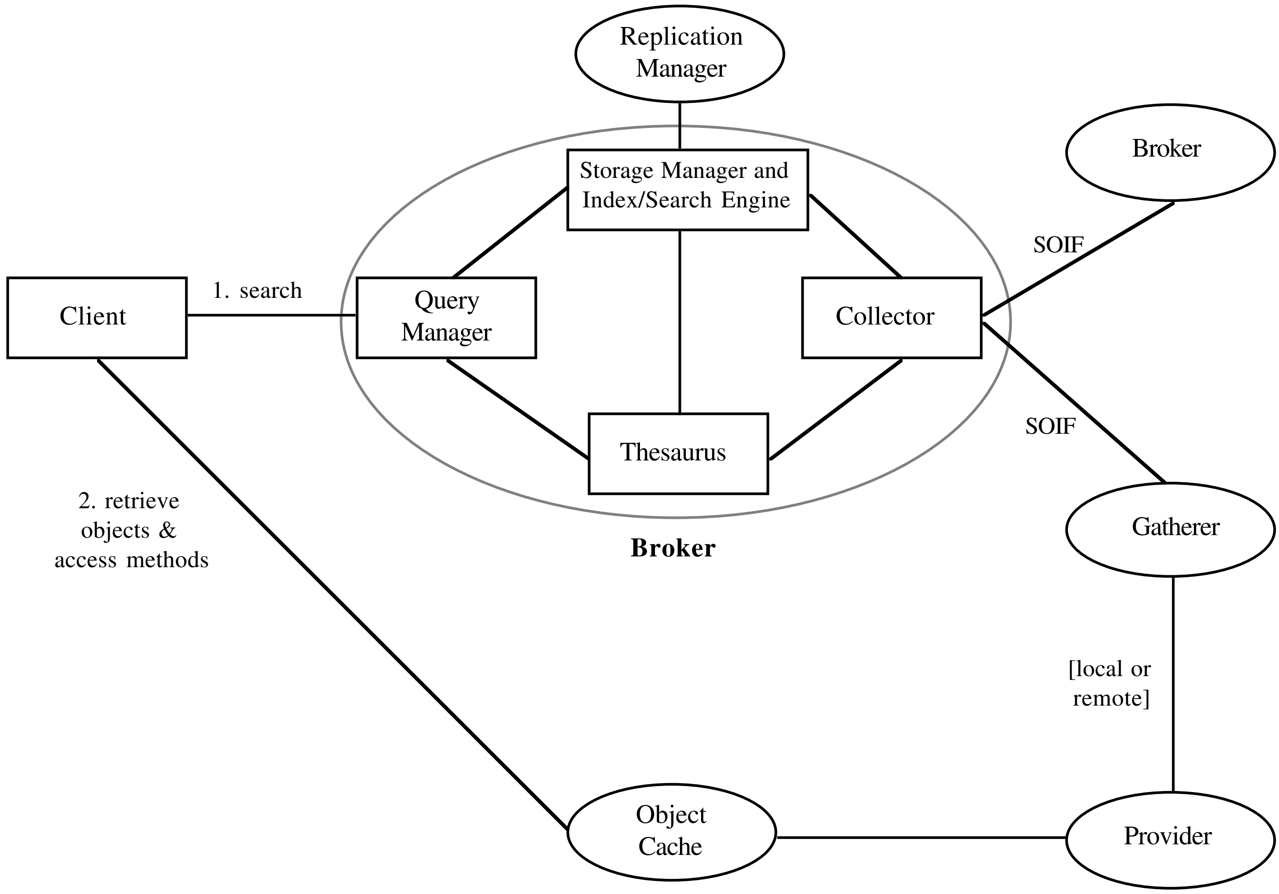
Although their principles are very different, the three system have a similar architecture, with 3 software modules : a query is initiated by a user with a software module called the "client" or "navigation server", the query is transmitted to a module called a "broker" or "gateway" which uses the information of the servers and other brokers to reply to a query or only translates the query for the servers, and a server (or "gatherer", or "publisher") handles the query or gathers indexing information. Each of the three software modules can exist in a number of instances, with various customisation : for example, a client can be configured to have a specific user interface and handle only a selection of data items.This same general architecture has been chosen in the three cases to provide a good flexibility.



Harvest (Bowman 1995)

Harvest is well represented by the above diagram (the names "broker" and "gatherer" are issued from it). A broker can be set to handle a selected vocabulary to avoid scaling problems of unfocused global indexes. Harvest uses SOIF (Summary Object Interchange Format) (Hardy & Schwartz, 1995) attribute-values to store the indexing information and exchange it between the gatherers and brokers. SOIF delineates streams of object summaries, and allows for multiple levels of detail. Each template (to summarise an object) contains a type, a URL, and a list of byte-count delimited attribute-value pairs. A number of attributes are mandatory or recommended with Harvest, to ensure a minimum metadata set.

Here is a more detailed diagram for Harvest architecture (from (Bowman et al., 1995)) :



With the different, well interfaced, software modules in the architecture, Harvest is very flexible. It allows the use of any indexing/search engine system, as long as it can handle boolean combinations of attribute-based queries, and supports incremental updates. Examples of indexing/search engine subsystems are Ingres or WAIS. Two particular subsystems have been developed for Harvest, each optimised for different uses : Glimpse (Manber & Wu, 1994) support space-efficient indexes and flexible interactive queries, while Nebula (Bowman et al., 1994) supports fast searches and views based on complex standing queries that scan the data on a regular basis and extract relevant information.

While the Harvest project has been terminated, some subsystems like Glimpse are still developed by other companies/universities, and it is still one of the best free systems for indexing and searching collections of documents on internet/intranet (Harvest 1999). Because it was designed to gather metadata from any SGML document, not only HTML, it can be applied to XML documents, which could give it a second life.

Application to AML

Harvest has been used to index collections of AML documents on a web server. A configuration file has been created to map AML elements to SOIF attributes, and, apart from minor problems due to the usual incompatibilities between SGML parsers and XML parsers, the AML documents were successfully indexed with Glimpse. The default web user interface was used to do boolean queries combining different AML elements to find AML documents. This worked well, although complex queries (for instance with regular expressions) did not always give consistent results. Glimpse still suffers from "finishing" problems, but the overall architecture seems able to reach its goals.

Example of SOIF attributes for an AML document

```
@FILE { http://strule.cs.qub.ac.uk/~damieng/these/MRK334.aml
```

```
update-time{9}:   935372990
title{8}:    MRK334
source{8}:   Simbad
ra{11}:      000309.55
keywords{29}:     iras
kug
mcg
mrk
pgc
ucm
ugc

identifier{54}:    http://strule.cs.qub.ac.uk/~damieng/these/MRK334.aml
ident{172}:
 UGC 6
 MCG+04-01-013
 Mrk 334
 ZW IV 1
 Z 0000.6+2142
 Z 477 - 43
 Z 478 - 15
 UCM 0000+2140
 KUG 0000+216
 [HB91] 0000+216
 IRAS 00005+2140
 VV 806
 PGC 207
dec{11}:     +215736.4
date{12}:    16/10/1997
creator{18}:       Damien GUILLAUME
amlver{5}:   0.2
amlobj{7}:  AOBJECT
md5{32}:    966f1b65b760d867832736d0f8d2dbdd
file-size{4}:     3075
type{3}:    AML
gatherer-version{6}:    1.5.17
gatherer-host{6}: strule
gatherer-name{13}:      strule server
refresh-rate{6}:  604800
time-to-live{7}:  2419200
last-modification-time{9}:    914090627
description{8}:    MRK334
}
```

Ingrid (Francis, Kambayashi, Sato & Shimizu, 1995) (NTT 1999)

The “broker” part of Ingrid is very different from the other search systems’ : each “Ingrid server” is storing a different set of “resource profiles” (storing the resources metadata), and these resource profiles are linked together among the different servers. While Harvest brokers focus on a specific vocabulary, Ingrid’s scalability is insured by the brokers focusing on the resources themselves, still allowing a quick retrieval by the use of the links between the resources. Ingrid was designed with the idea that scalability would quickly become a major problem for indexing resources on the internet, and has the most “distributed” architecture of the search systems studied here, to address this problem. So far, the centralised search engines have been able to keep up with the rapidly growing mass of information on the internet by using an improved hardware, but it is still not certain they will be able to handle all the internet metadata in the future.

What is Ingrid ? (this part is extracted from the Ingrid web site)

Ingrid is an architecture for a fully distributed, fully self-configuring information navigation infrastructure that is designed to scale to global proportions. Unlike previous designs, Ingrid is not a hierarchy of large index servers. Rather, links are automatically placed between individual resources based on their topic similarity in such a way that clusters of term combinations are formed. The resulting topology can potentially be searched and browsed by a robot efficiently.

The Ingrid project, of the NTT Laboratories, has been gradually abandoned, but mostly for non-technical reasons. It was successfully tested on IETF documentation, and other sites could have developed an Ingrid architecture. Ingrid was designed to be a whole-web search system, with scalability possibilities, so in the end it has not been tested for what is had been designed for.

Forgetting the sad fate of Ingrid, the rest of this section describes the topology design, but not the algorithms to create, use and maintain the grid (more information can be found on `http://www.ingrid.org/`).

The goals of Ingrid are :

- Medium-quality ("best effort") keyword-style searching of any text resources stored anywhere.
- High-quality browsing of the text resources (or, non-text resources described by text).
- Fully automatic operation (works in the absence of keyword or attribute selection and requires no manual placement of links).
- Fully distributed navigation (no dedicated single-database search engines are required).
- No limitations on the global number of resources or hosts.
- Adding value to resources by placing them near other relevant resources in Ingrid space.

Ingrid as Compared to HTML

Ingrid can be viewed as a web space parallel to (and complimentary to) the web space that exists by virtue of HTML links (see Figure 1).
The two web spaces are similar in that they are both composed of (URL-type) links between resources. The similarity, however, stops there.

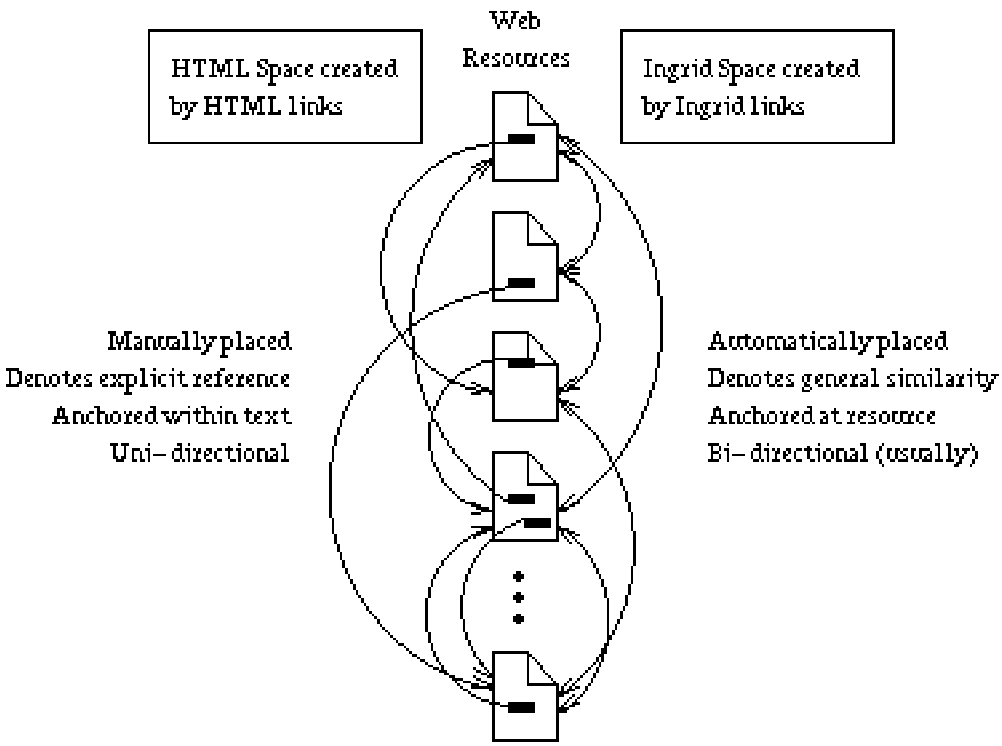


Figure 1 : HTML Space Versus Ingrid Space

HTML links are manually placed (by and large) and denote an explicit reference from one resource to another. Ingrid links are automatically placed and denote a general topical similarity between two resources. HTML links individually have a strong local meaning, but collectively (index and meta-index documents notwithstanding) do not contribute well to the global organisation of information. Ingrid links, on the other hand, have a somewhat weak local meaning, but collectively create a meaningful global organisation of information.

Browsing in HTML space can be considered as having two forms. One is browsing through index documents, and the other browsing among leaf documents (see Figure 2). Navigation among leaves takes place at the "micro" level, and tends to be from individual document to individual document. Navigation through indexes is at the macro level, and seems to be most often oriented towards organisations.
Sometimes it is oriented towards topic area, but to the extent that this is true, it is usually somewhat incomplete.

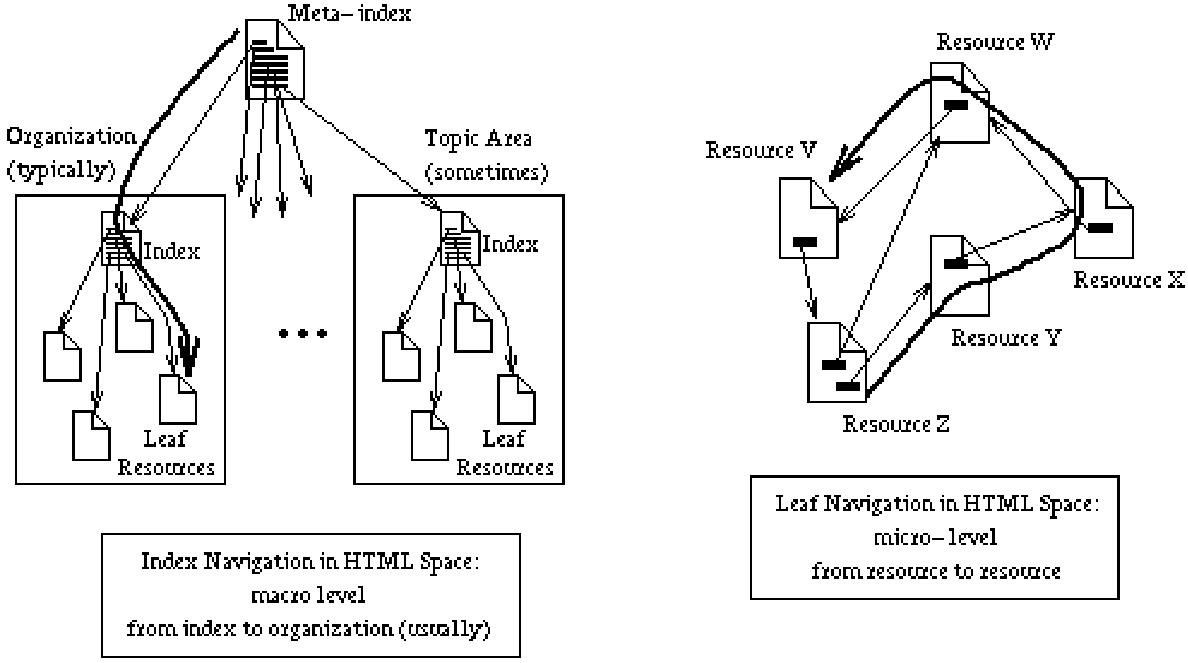


Figure 2 : Navigation with HTML

Navigation in Ingrid space is envisioned to take place almost entirely at the "macro" level, from topic area to topic area (where a topic area is composed of a group of documents, see Figure 3). In addition, the topic areas are interconnected. Thus, one can browse among related topic areas. Because of these differences in the two web (or information) spaces, they are highly complimentary.

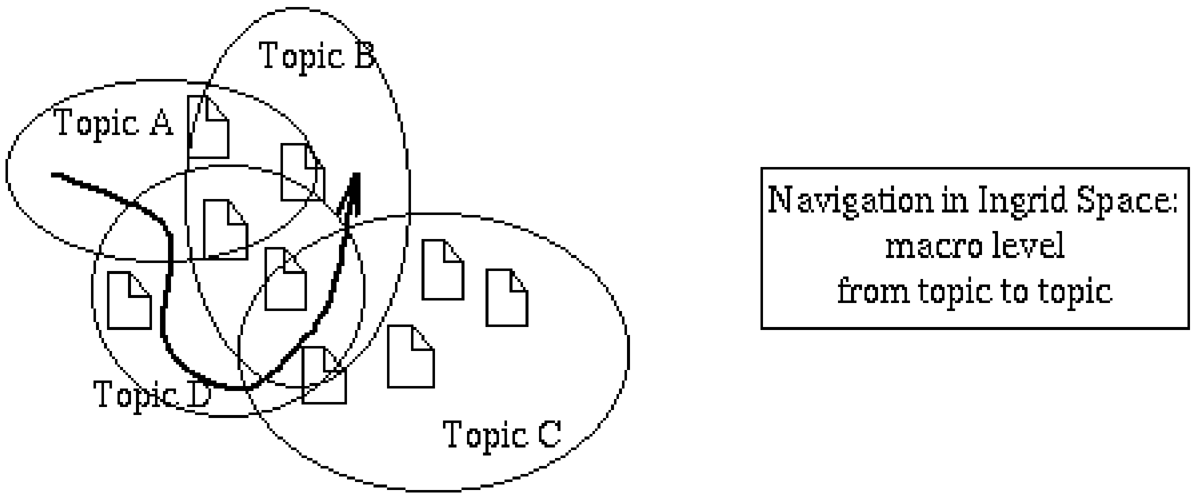


Figure 3 : Topic-level Navigation

Overview of the Ingrid Software Components

As shown in Figure 4, the basic component of the Ingrid Infrastructure is the so-called Resource

Profile. The Resource Profile is essentially a (text) summary of the resource that it represents (though the resource itself can be anything--an audio/video file, a physical object, or a service). The two required elements of the Resource Profile are :

1) Terms (or rather, term combinations) that describe the resource, and
2) A pointer (URL) back to the resource.

Other information may be available, such as resource title, authors, date, size, type, etc.

There are three types of Ingrid Software Components :

• Publisher : Retrieves documents from web servers, creates a Resource Profile (RP) from the document, and sends the RP to the Ingrid Server.
• Ingrid Server : Installs the RP into the distributed Ingrid infrastructure and answers queries.
• Navigator : Searches the Ingrid infrastructure and retrieves RPs for the User.

The Navigator comes in two types, the Single-User Navigator and the Navigation Server. The Single-User Navigator runs directly on the user's workstation and searches the Ingrid infrastructure from there. The user GUI is based on Motif. The Navigation Server runs on some server system, and can be accessed by many users via the users' browsers. There are HTML Forms and Java versions of the Navigation Server.

The following figure shows the Ingrid software components and their relationships.

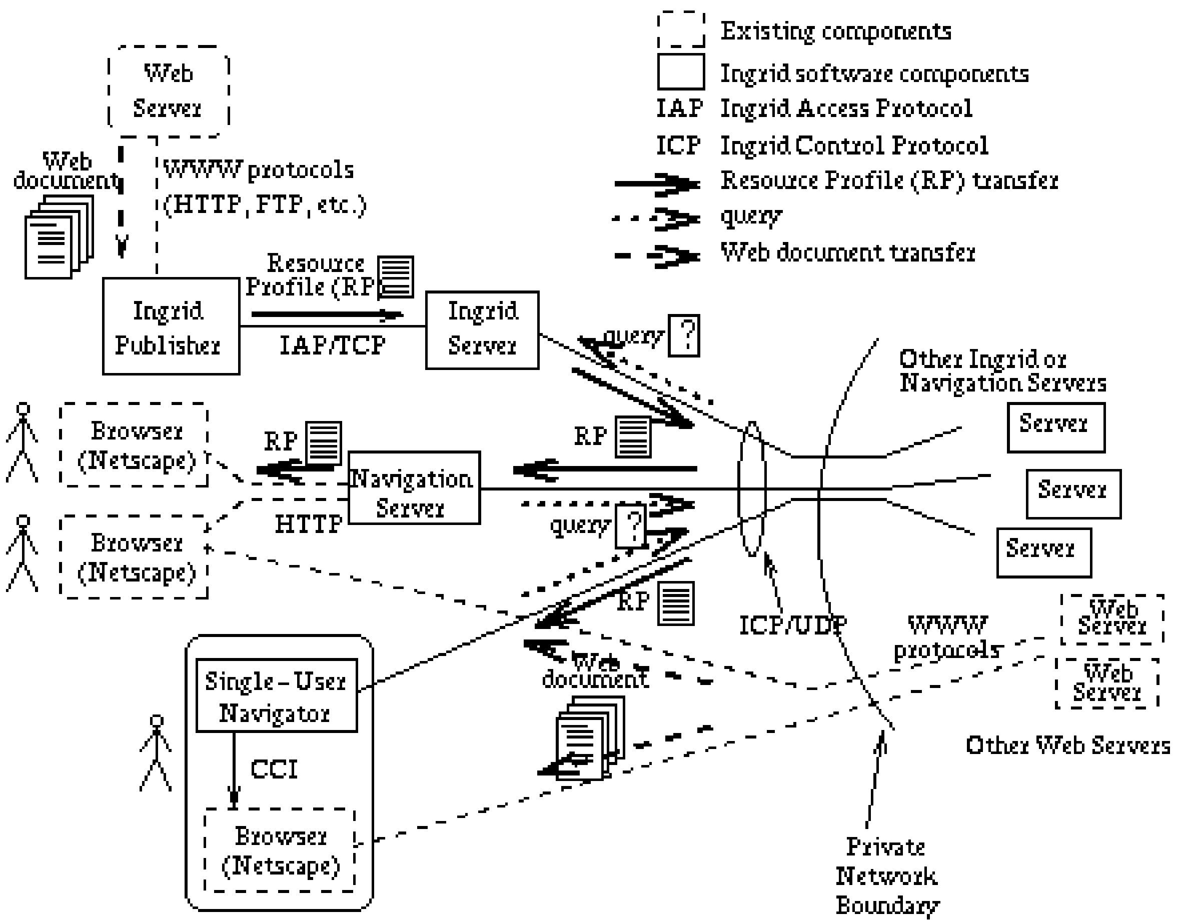


Figure 4

The publisher retrieves documents from a Web Server, either via the web using the publisher's robot, or if installed on the same machine as the web server, by directly reading from the directories holding the web documents.

The publisher isolates the terms in the web document and selects high-weight terms for insertion into the Ingrid infrastructure. Using these terms and the URL of the web document, plus other information such as title, author, etc., the publisher creates a Resource Profile (RP) and transmits it to the Ingrid Server using the Ingrid Access Protocol (IAP).

The Ingrid Server stores the RP. It inserts the RP into the Ingrid infrastructure by :

1. Searching the infrastructure for similar RPs.
2. Adding links to a certain subset of the found RPs.

The Ingrid Server does this using the Ingrid Control Protocol (ICP). ICP is a UDP-based (datagram) protocol. After the Ingrid Server has added links for its RP, it can answer queries from other Servers by using both link information and cached information.

Once an RP is inserted into the Ingrid infrastructure, other users can find it using one of two Ingrid Navigators. Both Navigators search the Ingrid infrastructure by sending ICP packets to Servers.

The modifications that were begun for Ingrid, and what was considered

XML

The first objective was to make Ingrid XML-aware, so that XML documents could be indexed. A first program was realised to create an RP from an AML document : some fields were directly translated into name/value pairs for the RP, and other fields were used to get keywords (with more or less weight, depending on their importance). It became soon clear that this program couldn’t be generalised for all the XML DTDs : to be efficient, the indexing has to use the specificities of the DTD. It also depends if we consider the XML files as data or metadata : here, AML was considered as metadata, as the RP, which is why some fields were simply copied. But if it had been considered as data, new fields would have been created for the RP, like the creation date for the AML file.

Another idea was to format the RP as XML documents, because of the many advantages of XML (especially for metadata), and because the creation of RPs for XML documents would have been easier this way.

Maps

It was for Ingrid that it was first thought of creating AML maps, to improve the search of the first resource given some keywords. This global, simplified view of the grid could have been more efficient than the simple caching that was used, but it would have required some kind of centralisation (or a one-time look at the whole grid), which wasn’t in the first direction of Ingrid.

The first results of an indexing of AML documents

A first result was obtained with the use of Ingrid on a small collection of AML documents. The indexing with the publisher, the organisation of the grid with the server and the creation of the interface with the navigator were fully automatic. A search with the resulting system was showing interesting results, but sometimes no result when there could have been one : this is because the metadata in the RP has some limits : there are a few simple information like the date of the file (as name/value pairs), and a size-limited list of keywords, and the limitation in the representation of a document is reflected in the grid. Also, the links created between the documents are optimised for a search by keyword, and do not have a semantic : sometimes, a part of a keyword is used as a link, so that new links can be created between the documents, but this link has no specific meaning.

Because the links had no semantic, unlike the links existing now in AML, an AML map created from an Ingrid grid would have been very different from the current AML map : it wouldn't have been a "semantic map", and thus wouldn't have been human-readable, but it could nonetheless have been used by an Ingrid server to locate a document faster.

Although the first results obtained with AML were not at all disappointing, the project was abandoned, for various reasons, most of these reasons being non-technical : a reorganisation in NTT has led to all the people but one being moved to another project, and the last one (the project manager, Paul Francis) had an idea for a new project. Also, Paul Francis wanted to improve Ingrid's performance and stability before adding other features.

Emerge (NCSA 1999a)

Emerge is a framework for information search and retrieval, focusing on the needs of scientific data. Scientific data is often complex, requiring complex metadata. For example, an astronomical object can be identified by names or coordinates, but there are often more than ten names per object, and the coordinates require specific conversions before they can be used.

When object types are used for a search, it is difficult to use different heterogeneous databases, because each database is using a different way of encoding this information. Even when the information is written with the same format, it is usually stored under different names. In the Emerge context, a set of names of metadata for a given server is called a "profile". A mapping of the different search profiles, with the help of a "meta-profile", is needed in order to do a search on several servers.

Once an object is selected, various kinds of data can be found about it, but usually in different formats.

When a search or a data processing is required on many objects, it can be less CPU-consuming for the server to have a user script run on it and returning the global results, instead of letting the user download all the files and process the data locally.

To summarise, a search for scientific data can be complex, and require specific processing using a specific profile.

Z39.50

Most protocols, like HTTP, are not adapted to the dialogue required between a user and a library. Emerge is using ANSI/NISO Z39.50 (Z39.50 1999) (ISO23950), a standard well established for bibliographic data, with scientific data. Z39.50 is a statefull protocol, allowing a real communication between a client and a sever. Because of its modular design, most features can be developed and added incrementally, and still maintain backward compatibility with older implementations. The standard was originally designed to use the ISO/OSI (International Organisation for Standardisation / Open Systems Connection) presentation layer protocol. However, nowadays, most implementations are using TCP/IP. The features include :

- initialising
- controlling access : per-session and/or per-operation
- searching : done with a boolean query, in a well-known search format, with many operators available
- presenting records : a client can request a number of records matching some criteria, and because whole records can be transmitted (not only URLs), it is possible to make use of their content
- maintaining multiple search results
- browsing : its is possible to browse a database using the index terms or specific fields

• sorting of results
• explaining server capabilities
• controlling resources : a client can cancel a search or present a request in the middle of an operation. It is also possible to get accounting information (like the number of searches performed), which is useful for fee-based online services
• extended services : persistent or periodic queries can be executed by the server on behalf of the client, database maintenance, etc…

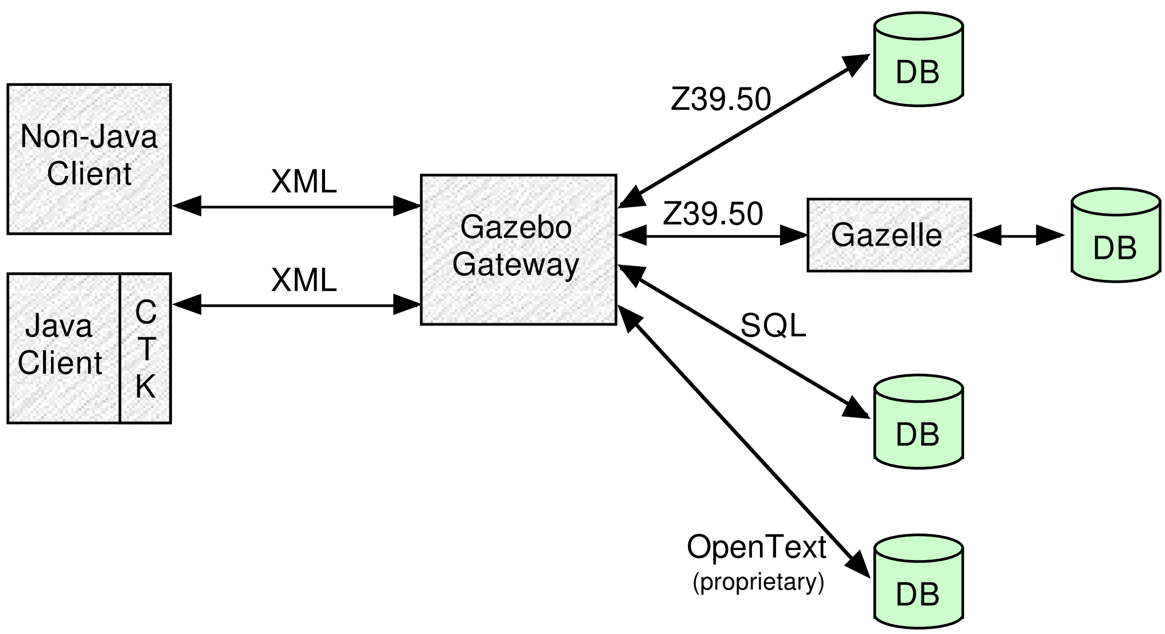
Many initiatives have been built on top of Z39.50 : GILS (Global/Government Information Locator Service, described in the Data Structures chapter) provides a uniform way of locating information (not only online) and is used to let people search the US government information. Isite is a search system implementing Z39.50 and providing a simple user interface to it.

Emerge Architecture

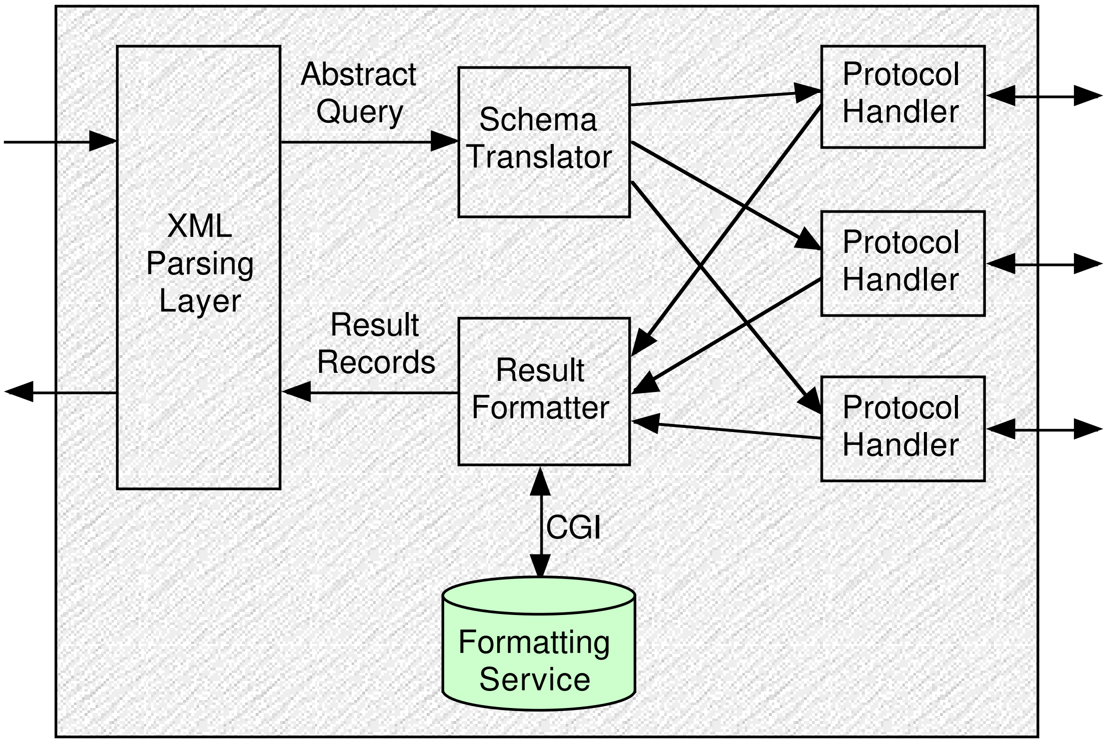
As Emerge does not use indexes, its architecture does not require a “gatherer” to index data locally. Instead, the databases can be queried directly from the “gateway”. Various query standards are supported by the gateway, but Z39.50 is the best supported. As implementing a Z39.50 server is sometimes difficult, a component called “Gazelle” (formerly known as “Pizzazd”) has been created to become an interface easy to implement between Z39.50 and any database. As Gazelle is stored on the server side and is an interface between the database and the gateway, it plays a role similar to the gatherer in the other search frameworks.

The client and the gateway communicate with a protocol specific to Emerge, using XML. To simplify the creation of new clients, a Java interface for this protocol, “CTK”, can be used to perform all the XML parsing and network communication.

Emerge Architecture



Gazebo Architecture



The Gazebo gateway is the link between the clients and the servers. It transforms a client query into queries understandable by the servers. Like Z39.50, it is based on an abstract data and query model, but unlike Z39.50 it is also based on an abstract model of search attributes, which map to parameters passed to the target data sources. This mapping is used to express semantic equivalences between different metadata schemas and thus make interoperable search across heterogeneous data collections possible. The schemas available in Gazebo can be obtained by the client with the client-gateway protocol in XML. The above graph shows an abstract query decoded with the XML Parsing Layer, and transformed by the Schema Translator into different queries for the servers, with their own schemas. The queries are sent to the servers with their own Protocol Handlers (although Z39.50 is recommended, other protocols can be used).

The client can request the results in short or full form (this is a Z39.50 feature), and get them asynchronously (the queries are executed separately but at the same time, and each result can be displayed by the client when it is returned). The records can be returned in any format, and scripts can be used to convert the formats returned by the servers into other formats (including binary formats), like HTML.

Finally, Gazebo is highly configurable, with an XML configuration format.

Emerge and AML

A new server has been created with Gazelle to query the servers providing AML documents : NED, Simbad, ADS, the Benn&Martin database, and the ADIL (Astronomy Digital Image Library, described in the following section). This lets a users query these servers, although the search possibilities are currently very limited : only an identifier can be used; a better search would require a direct access to these databases and a specific implementation for each.

When the gateway returns AML documents (as full records), they are displayed by the client with the AML browser. Other helpers are used for other documents, such as the ICE HTML (ICEsoft 1999) viewer for HTML documents, or the Horizon Image Browser (NCSA 1999b) for FITS images. From a returned document, a user can still navigate with the AML browser, using the links between the AML document, thus using a mix of query/result search and navigational search.

Possible extensions and future frameworks

Unifying Emerge and Harvest

The main problem with Emerge is the difficulty to create a new server, especially for simple collections of documents without any associated software. As shown in the graph of the Emerge architecture, it is designed for “DB” (databases) servers, as if all the scientific data was strictly structured and stored in a DBMS (Database Management System) having at least an SQL implementation. Unfortunately, it is not the case : many servers just have collections of documents, and the software is very limited : any implementation of complex boolean queries is difficult.

On the other side, Harvest can index automatically collections of documents in various formats (including not-well-structured formats like HTML), and execute boolean queries on the indexed fields. But Harvest queries are limited to simple boolean queries and is far from letting a client send a procedure to be executed on the server. Its protocol does not allow much interaction.

A better framework would use the best of Emerge and Harvest : it would manage complex queries, map attributes, and use a database of indexed information with a gateway/broker, and use gatherers or not on the server side. Some servers would require direct queries, possibly complex or specific, and others would just provide indexed information that can be transferred to the gateway/broker and used without querying the servers.

Using XML and XQL

When the servers provide XML documents, it becomes easy to build a “plug-and-play” server for a search framework. The XML documents can be searched in an XML database, or simply indexed automatically like other documents, with attribute/value pairs. The only information missing for that in the DTD is whether an element is “searchable” or not (i.e. useful for a search), but a simple software could let a provider create a configuration file from the DTD, with a graphic user interface if necessary (it would be the equivalent of mapping the SGML elements to SOIF attributes in Harvest, except that it could be done simply by selecting the searchable elements). Reducing XML data to simple attribute/value pairs is not the best solution, because the structural information can be lost in the process. Also, XML-specific queries can make a better use of XML documents, but unfortunately there is no standard query language for XML at the moment. When an “XQL” query language for XML will exist, it will be used to query collection of XML documents, and hopefully an extension of Z39.50 will support it; “flat” attribute/value pairs is just a special case of structured data, so backward compatibility will not be threatened by this evolution, and search frameworks should support the general case of structured data.

VIII. OTHER COLLABORATIVE WORK

The principles and tools previously defined can be applied to many projects. The ADIL, for example, is a good test case for the internal database structure, because it is using various kinds of multimedia information. Since exchange of information is also important for the ADIL, it is also a good test case for this. Other projects, like ISAIA, have a larger point of view and aim at standardising information exchange among different disciplines or sub-disciplines.

ADIL (Astronomy Digital Image Library)

Principles (Plante , Crutcher & McGrath, 1998)

The fast increase of instrument's data production rate, occurring throughout all observational astronomy, has led to a new challenge : processing and understanding the information at least as fast as it is produced. The ADIL has been created in order to solve this problem, by improving data gathering, access to the data, and links to high-performance analysis and visualisation tools.

The technical improvements, though, are only part of the answer : an evolution in the habits of researchers is also important, and the ADIL should help astronomers to understand how they can improve their manner of accessing and publishing information. In particular, shared processed data is the key to allow comparisons and reuse of other works : the processing converts the image data to physical units, so that knowledge of the instruments is no longer needed to understand the image.

Done as a professional courtesy, the practice of sharing processed data has traditionally been an informal one : an astronomer wishing to reuse processed data would contact the authors who had published the original results based on the data and ask them for copies. The original authors would then have to dig through their personal collection of archive tapes for the data and go to the trouble of making a copy; since most researchers do not usually have the time or resources to keep a well-maintained archive, there would be no guarantee that the desired data could be found and copied. When the exchange is successful, the researcher must credit the original authors in any subsequent publications, and she should consult the original publication to ensure that the data are used in a way consistent with the constraints of the original experiment.

Although some observatories can now provide a good access to the data on the web (for instance, the CADC, Canadian Astronomy Data Center, distributes data calibrated "on-the-fly"), there are still smaller observatories that can not afford to maintain their own archives. Also, images with physical quantities instead of just calibrated measurements of brightness, are usually not available on the web.

The goal of the ADIL is to provide access to valuable image data that are otherwise not available through other archives. Acting as an intermediary between scientists, it aims to not just ease the exchange of data, but to encourage it. To ensure the quality of the data, the ADIL requires that it is associated with a published article, and that the images are the fully processed versions which led to the published results.

Although it might require a cultural change, the publishing of the processed data at the same time as the article, together with efficient tools to access to the data, should greatly speed up the cycle of publication, thus accelerating research as a whole.

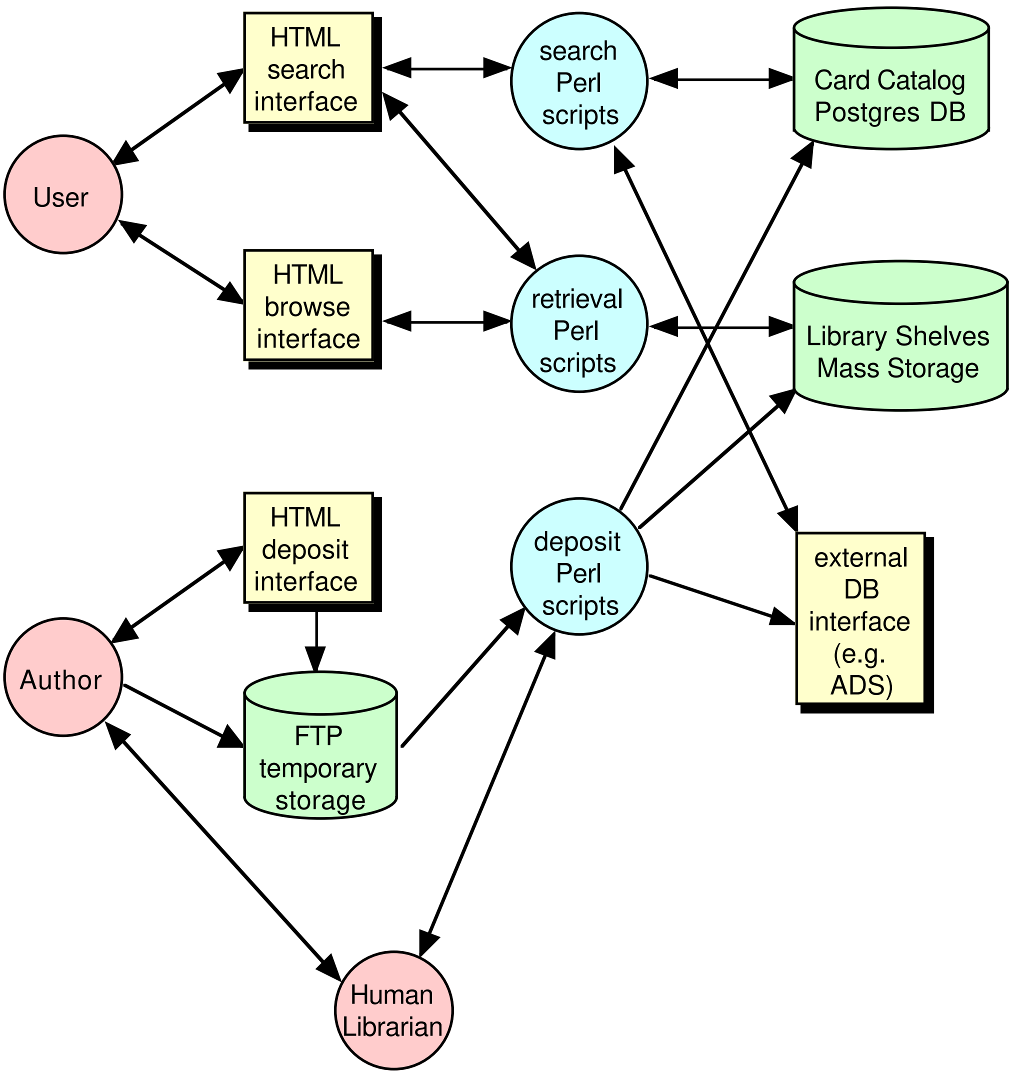
Current organisation

As most databases in astronomy, the ADIL is working on a UNIX system (Solaris™). It is based on an HTML interface, Perl CGI scripts, Perl processing scripts, a mass storage with a cache simply using UNIX file system mechanisms, and a Postgress™ database to do SQL queries on the metadata.

The data flows, from an addition to the ADIL by an author to the browsing of the data by a user, are represented in the following graph.

The ADIL stores information on a project in astronomy, the related article, and the fully processed images used for the project. An author gives this information on the internet, with an HTML interface to give a description, and via FTP to upload the data files. An “electronic librarian” can then transform the input automatically, extracting the metadata, to create a first set of documents to be incorporated into the database. The human librarian checks the resulting documents and can discuss with the author about them, before they are sent to the library shelves. The “library shelves” are actually a disk cache linked to the NCSA Mass Storage System (MSS, a tape-based system of fast drives) for long-term storage. A summary of the article can also be sent to ADS, so that both databases are in sync. The Postgress database, containing only the metadata used for searching the documents, is also updated at this time.

A user can query the database with HTML forms. The query is processed by CGI Perl scripts, which transform the query into SQL, send it to the Postgress database, and return a list of matches in an HTML document. The metadata about the projects, images (including a GIF preview) and articles can then be browsed by the user with other scripts accessing the library shelves. The HTML documents browsed have numerous links to other documents in the ADIL and to ADS documents. They also have links to the full FITS images, that can be retrieved via another Perl script.

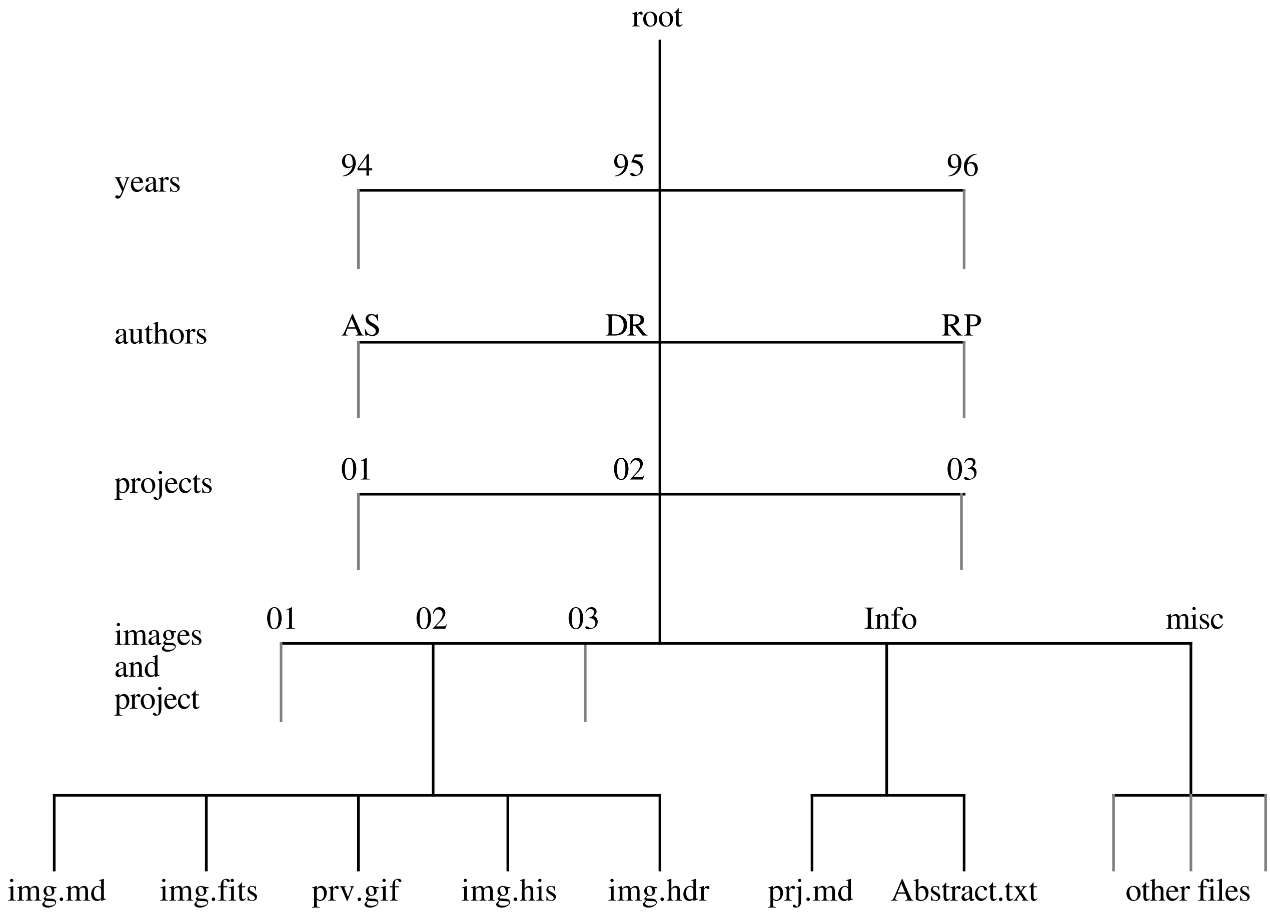


Study of the database in order to improve the structure

Data structures

The “library shelves”, containing the data and the metadata, are stored within a hierarchical directory structure on the disk, as shown in the next graph. The first level is a list of years. Each year directory contains a list of authors having deposited at least one project that year. Each author directory contains a numbered list of project. Each project directory contains folders for the images and information on project. The information on the project is composed of a metadata file with a specific format (prj.md), and an abstract describing the project (Abstract.txt). Each image folder contains the fits image or a link to the fits image on the MSS (img.fits), a metadata file (img.md), a file with the FITS header (without the history) (img.hdr), another one with the history of the FITS document (which can be very long and thus doesn’t mix well with the rest of the metadata) (img.his), and a GIF or MPEG preview (prv.gif and/or prv.mpg).

There is a redundancy between the metadata file and the FITS header (which exists both as a separate file and in the FITS file), but only the metadata file is used with the ADIL. The FITS header can be displayed to the user, because its format is known to most astronomers, and the FITS header in the FITS file is never used (the FITS file is never changed once it has been deposited, and its size is usually much smaller than the image, so it is not necessary to split it and combine it when the user downloads it).



The metadata files use the following format : they are composed of attribute/value pairs, with a type encoded on one character. The value can be a list of values, separated by a pipe character.

Here is a description of this format :

```
NOTES
  o  All lists delimited by |.
  o  Types: s = string
          b = boolean
          i = integer
          d = floating point

PROJECT METADATA
codename    :s: an ADIL identifier for a data object
author      :s: the list of authors.
                Format:   LAST:FIRST[|LAST:FIRST]*
title       :s: title of an ADIL project
reference   :s: list of literature references describing this project
                and its data; each given by a bibcode; non-valid
                bibcodes are preceded by a !.
issurvey    :b: if true, images are part of a sky survey
relproj     :s: list of ADIL projects related to this one, each given
                as an ADIL codename
contact     :s: The name of the person to contact about this project
email       :s: email address of contact person
depdate     :s: date/time of deposit of project
uauthor     :s: a representation of the author list to appear in
                online preview documents.
priority    :i: a measure of the data object's mortality in the disk
                cache; a high value means it will last longer in
                the cache before being purged.
pubinterest :i: a guess as to how interesting this image may be to the
                general public
copyright   :s: the copyright year and holder
crhandle    :s: an identifier for the the copyright (used to locate
                the complete copyright statement.
comment     :s: general comment

IMAGE METADATA
ra_min      :d: minimum right ascension covered by image
ra          :d: identifying right ascension of image (usually image center)
ra_max      :d: minimum right ascension covered by image
dec_min     :d: minimum declination covered by image
dec         :d: identifying declination of image (usually image center)
dec_max     :d: maximum declination covered by image
restfreq    :d: rest frequency of primary spectral line being imaged
freq_min    :d: minimum frequency covered by image
freq        :d: identifying frequency of image (usually image center)
freq_max    :d: maximum frequency covered by image
species     :s: chemical formula for spectral line species being imaged
transit     :s: the transition of the spectral line
objname     :s: list of names of objects included in image (should be
                   standard name recognized by SIMBAD whenever possible)
objtype     :s: list of object types included in image
telescope   :s: telescope name
uname       :s: name given to image FITS file by contributing author
filesize    :i: size of FITS file in kB
EPOCH       :f: value of FITS keyword EPOCH
BUNIT       :s: value of FITS keyword BUNIT
NAXIS       :i: value of FITS keyword NAXIS
NAXIS1      :i: value of FITS keyword NAXIS1 (when applicable)
NAXIS2      :i: value of FITS keyword NAXIS2 (when applicable)
NAXIS3      :i: value of FITS keyword NAXIS3 (when applicable)
NAXIS4      :i: value of FITS keyword NAXIS4 (when applicable)
prvannot    :s: annotation to display with preview image
```

```
prvplane    :i: the slice within the image shown in the preview image
prvscale    :d: the scale factor needed to produce the preview from
               original FITS
prvclrmap   :s: the colormap file needed to produce the preview from
               original FITS
prvmin      :d: the minimum data value needed to produce the preview from
               original FITS
prvmax      :d: the maximum data value needed to produce the preview from
               original FITS
prvsub      :d: the subregion shown in the preview image
description :s: a general description of the image
prvxscale   :d: the x scaling needed to produce the preview from
               original FITS (overrides prvscale)
prvyscale   :d: the y scaling needed to produce the preview from
               original FITS (overrides prvscale)
prvrotate   :d: the rotation needed to produce the preview from
               original FITS
```

Software structures

Most of the internal software for the ADIL is written in Perl (Songline 1999). Perl is a powerful and dense scripting language, very used for CGI scripts on UNIX. However, Perl version 4, which was used in the ADIL, is not object-oriented. Also, it is reliant on the operating system : it could be ported from one UNIX to another, but it would be very difficult to port a collection of Perl scripts to a non-UNIX system. Some people may argue that it is possible to write good object-oriented multi-platform software with Perl5, but in practice, a programming language like Perl incites people to write quickly a non-reusable code. Perl scripts are usually difficult to maintain if one doesn't pay extra attention about maintenance questions, especially for big projects.

The software structure for the ADIL is not very well defined : it is difficult to know which piece of software uses which other piece of software. However, there are a number of Perl library files, with precise tasks, and a defined interface (this precise interface can only be defined as comments, not in the Perl language). These libraries are generally used by CGI scripts.

Transformation into XML

Why XML ?

XML was chosen as a new format for the metadata for the following reasons :

- more flexibility, lower maintenance cost;
- it bring the internal structure closer to the published structure (which is multimedia information), without losing the good properties of structured information;
- it allows the publication of both a GUI with HTML, and reusable data with XML. XSL can be used to transform an XML file into an HTML one;
- less specific software is needed with XML and XSL : an XSL document is all that is required to create HTML documents, and the transformation is defined in a declarative way;
- to split the internal software and data structure into several modules with clearly defined interfaces. An interface can be a DTD to specify the content of XML documents exchanged between the modules.

How?

- with AML, to clearly separate the public information from ADIL-specific metadata
- the ADIL-specific data has a DTD, which is combined with the AML DTD to get the whole DTD for the documents. Since some metadata used in the ADIL are of general interest, they have been added to AML.
- The use of AML requires the separation of the metadata between several objects, with links between them. The following objects are provided by the ADIL, and thus can have their own

AML documents : PROJECT, IMAGE, ARTICLE. At the beginning of the study, it was not clear whether the PROJECT and ARTICLE objects were two different objects, but the transformation into AML helped us to understand better the data served by the ADIL, and its organisation. It also resulted in the creation of a new AML object : PROJECT.

- XSL stylesheets are defined for the transformation into HTML, as described below.

Here is the resulting DTD for the ADIL. Most of the metadata information is in AML (and not here), because it is of public interest.

```
<!-- for the ADIL only -->

<!ELEMENT ADILPROJ (ISSURVEY?, UAUTHOR, PRIORITY?, PUBINTEREST?, CRHANDLE?) >

<!ATTLIST ADILPROJ
    xmlns         CDATA                      #IMPLIED >

<!ELEMENT ISSURVEY      (#PCDATA) >
<!-- boolean(F/T); if true, images are part of a sky survey -->
<!-- default value: F -->

<!ELEMENT UAUTHOR       (#PCDATA) >
<!-- a representation of the author list to appear in
     online preview documents -->

<!ELEMENT PRIORITY      (#PCDATA) >
<!-- integer; a measure of the data object's mortality in the disk
     cache; a high value means it will last longer in
     the cache before being purged -->

<!ELEMENT PUBINTEREST      (#PCDATA) >
<!-- integer; a guess as to how interesting this image may be to the
     general public -->

<!ELEMENT CRHANDLE      (#PCDATA) >
<!-- an identifier for the copyright (used to locate
     the complete copyright statement). -->



<!ELEMENT ADILIM (OBJNAMES?, OBJTYPES?, UNAME, FILESIZE, PREVIEW?, IMCOORD?) >

<!ATTLIST ADILIM
    xmlns         CDATA                      #IMPLIED >

<!ELEMENT OBJNAMES   (OBJNAME+) >
<!-- the objects visible on the image -->
<!ELEMENT OBJNAME    (#PCDATA) >
<!-- an object name -->

<!ELEMENT OBJTYPES   (OBJTYPE+) >
<!-- the types of the objects visible on the image -->
<!ELEMENT OBJTYPE    (#PCDATA) >
<!-- an object type (classification?) -->

<!ELEMENT UNAME      (#PCDATA) >
<!-- name given to image FITS file by contributing author -->

<!ELEMENT FILESIZE   (#PCDATA) >
```

```
<!-- integer; size of FITS file in kB -->

<!ELEMENT PREVIEW  (PRVANNOT?, PRVPLANE?, PRVSCALE?, PRVCLRMAP?, PRVMIN?,
                    PRVMAX?, PRVSUB?, PRVXSCALE?, PRVYSCALE?, PRVROTATE?) >

<!ELEMENT PRVANNOT   (#PCDATA) >
<!-- annotation to display with preview image -->

<!ELEMENT PRVPLANE   (#PCDATA) >
<!-- integer; the slice within the image shown in the preview image -->

<!ELEMENT PRVSCALE   (#PCDATA) >
<!-- double; the scale factor needed to produce the preview
     from original FITS -->

<!ELEMENT PRVCLRMAP  (#PCDATA) >
<!-- the colormap file needed to produce the preview from original FITS -->

<!ELEMENT PRVMIN     (#PCDATA) >
<!-- double; the minimum data value needed to produce the preview from
     original FITS -->

<!ELEMENT PRVMAX     (#PCDATA) >
<!-- double; the maximum data value needed to produce the preview from
     original FITS -->

<!ELEMENT PRVSUB     (#PCDATA) >
<!-- double; the subregion shown in the preview image -->

<!ELEMENT PRVXSCALE  (#PCDATA) >
<!-- double; the x scaling needed to produce the preview from
     original FITS (overrides prvscale) -->

<!ELEMENT PRVYSCALE  (#PCDATA) >
<!-- double; the y scaling needed to produce the preview from
     original FITS (overrides prvscale) -->

<!ELEMENT PRVROTATE  (#PCDATA) >
<!-- double; the rotation needed to produce the preview from
     original FITS -->

<!ELEMENT IMCOORD (LONMIN, LON, LONMAX, LATMIN, LAT, LATMAX) >
<!ATTLIST IMCOORD
    csys           (galactic | ecliptic | equatorial)    #REQUIRED
    equinox        CDATA                                  #REQUIRED >

<!ELEMENT LONMIN     (#PCDATA) >
<!-- minimum right ascension covered by image -->
<!ATTLIST LONMIN
    format         CDATA                                  #REQUIRED >

<!-- LON already defined -->
<!-- identifying right ascension of image (usually image center) -->

<!ELEMENT LONMAX     (#PCDATA) >
<!-- maximum right ascension covered by image -->
<!ATTLIST LONMAX
    format         CDATA                                  #REQUIRED >

<!ELEMENT LATMIN     (#PCDATA) >
<!-- minimum declination covered by image -->
```

```
<!ATTLIST LATMIN
    format          CDATA                                      #REQUIRED >

<!-- LAT already defined -->
<!-- identifying declination of image (usually image center) -->

<!ELEMENT LATMAX     (#PCDATA) >
<!-- maximum declination covered by image -->
<!ATTLIST LATMAX
    format          CDATA                                      #REQUIRED >
```

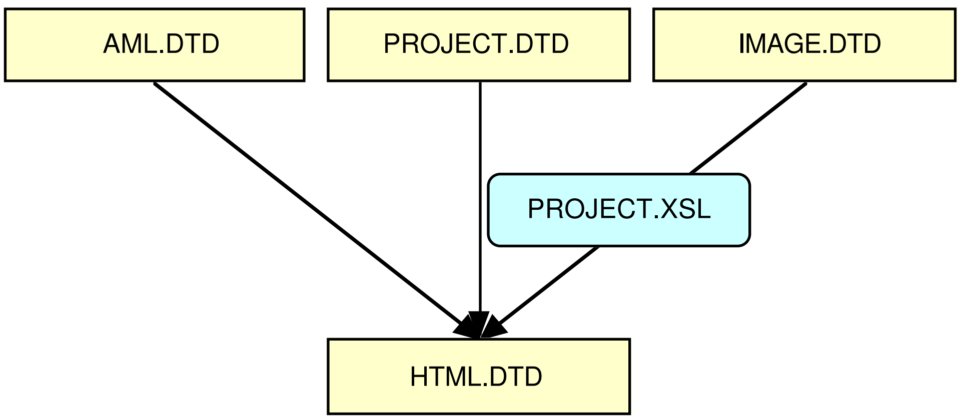
Use of XSL to create HTML documents

As of August 1999, XSL is said to be divided into XSLT, the XML language to transform a set of XML documents into another set of XML documents, and a formatting vocabulary. XSLT is using XPath (also used by XPointer) to address parts of XML documents. XT, a Java implementation of XSLT ©James Clark, has been used to create HTML documents from the ADIL AML documents. XT is based on the W3C working draft for XPath dated 13 August 1999 (W3C 1999h) (`http://www.w3.org/TR/xpath`) and on the W3C working draft for XSLT which came out at the same date (W3C 1999i) (`http://www.w3.org/TR/WD-xslt`).

As usual HTML documents are actually SGML documents and not XML, XSLT transformations into HTML documents actually creates XHTML documents, and then change the syntax automatically to output HTML documents. XHTML is the new XML language equivalent to HTML, but strictly conforming to the XML syntax.

In the previous implementation of the ADIL, an HTML template was already used, with a home-made markup to specify where to fill in the information, but filling it required Perl scripts. The XSL documents use the same idea : they are an HTML document, with the XHTML namespace, with added elements using the XSL namespace to specify elements to add from the input documents.

At the beginning, we tried to use an XSL document to convert the AML documents for a project into HTML files, with an XSL document "PROJECT.XSL". These AML documents are based on the AML DTD and on the DTD for a project specific to the ADIL, PROJECT.DTD. Their links point to AML documents for images, also used to create the HTML document. The whole transformation can be represented by the following diagram :

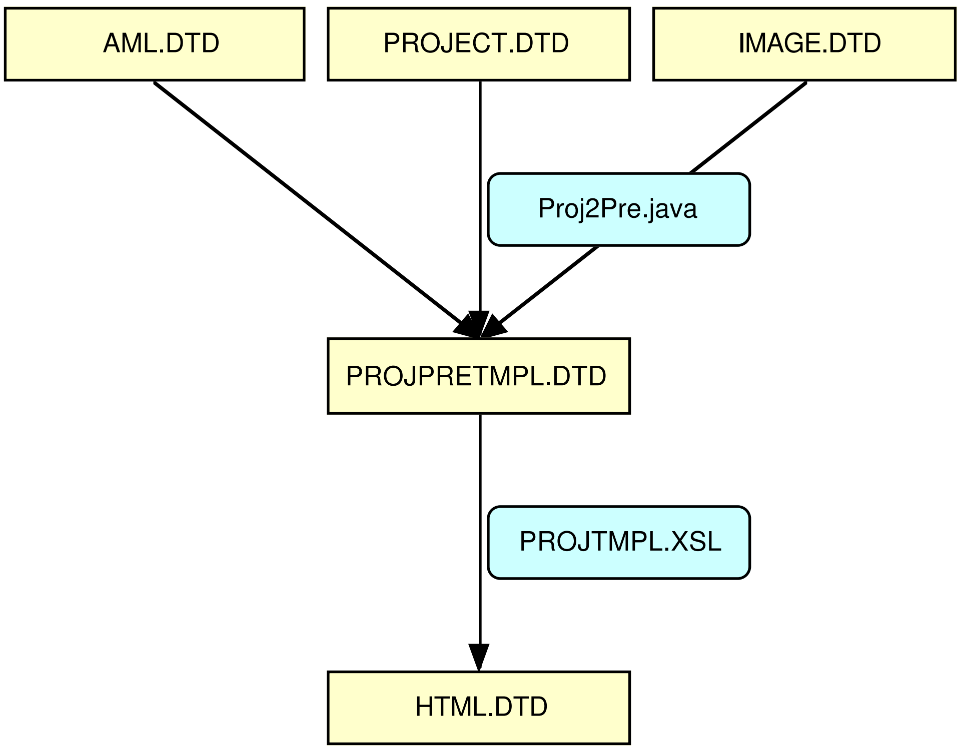


A problem encountered was the complexity of the transformation : as XSL does not support the use of variables, some transformations were very difficult or impossible to write in XSL (like the replacement of variables used in a table of images), resulting in an XSL document difficult to read. The advantage of using XSL documents is that they are configuration files instead of programs, and thus should be machine-independant and easy to modify. In this case, the XSL document was not easier to read than a program, which clearly showed the limits of XSL.

Another problem when applying this method was the time required to process the XSL documents. This version of XT is not optimised for speed, but the whole process is much more heavy in CPU resources, and will always be, than simply using a Perl script to create an HTML document. The XSL processor has to parse the XSL document and the XML document, then to interpret the XSL document. The XSL commands are sometimes complex : they are designed to easily specify XML transformations, which are not trivial operations.

The speed problem could be solved by creating programs from the XSL documents. As they are relatively static, one could imagine a program transforming an XSL document into a program able to do the transformation, and compiling it automatically. This would only require to start this program each time the XSL document is changed, and it would remove the level of interpretation caused by the use of XSL documents. This means that the speed issue is not a design flaw : it just requires an adequate software, which has not been created yet.

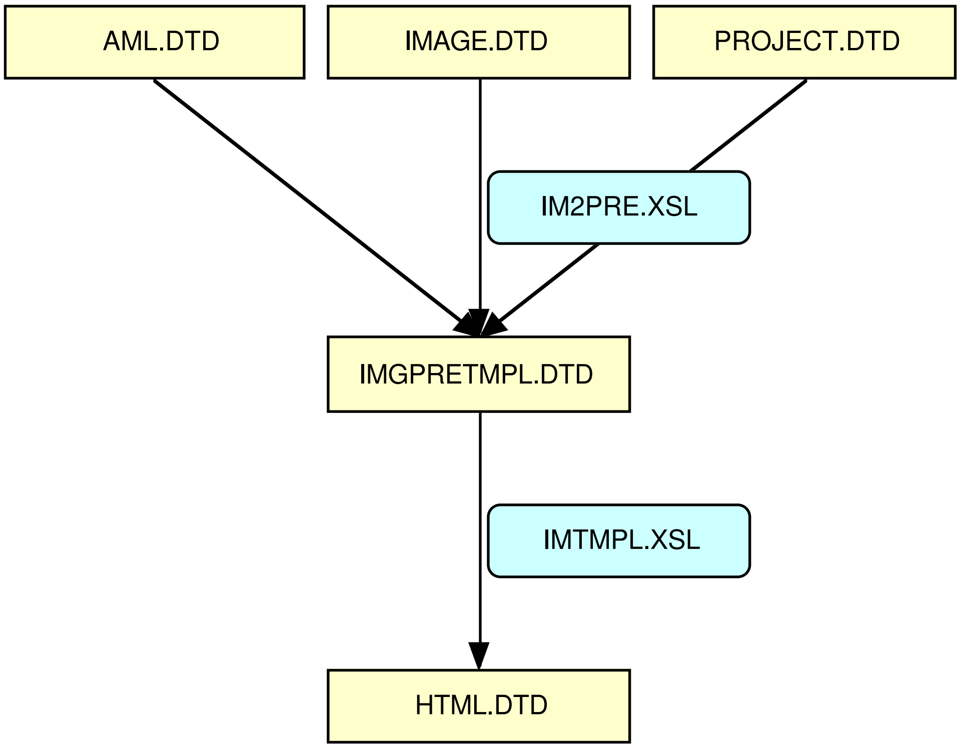
It turned out that XT was actually fast for simple XSL transformations. We decided to use an intermediary XML DTD containing all the elements to publish, and to create the intermediary files with a Java program. The HTML documents are still created with an XSL document, but only using the intermediary documents :



Although it seems more complex, this is a much better solution :

- the published information is clearly expressed in a DTD, independently from any formatting
- the process of creating the information to publish, and the one to format the document into HTML, are well separated
- the whole transformation is faster, because a Java program is used for complex transformations

A similar transformation was used to create the HTML documents about the images, except that an XSL document was used instead of a Java program to create the intermediary file because this transformation was much more simple than for the project.



IM2PRE.XSL, IMGPRETMPL.DTD and IMTMPL.XSL are in Appendices 5, 6, 7.

Towards a unification of astronomical metadata

Other developments for XML in Astronomy

A number of other projects in astronomy have recently started using XML, and they have defined new DTDs. A list of pointers, maintained by the NASA (Ames, Koons & Sall, 1999a), contains the following entries :

- The Astronomical Data Center (ADC) has developed a DTD for astronomical tables, to maintain table metadata in the archive. This DTD is more detailed than the one initially defined for AML. (Borne, Shaya & Edward, 1999, Shaya et al., 1999)

- NASA Goddard Space Flight Center and Century Computing (a division of AppNet, Inc.) have defined 2 DTDs to describe instruments : IML, Instrument Markup Language, and AIML, Astronomical Instrument Language, which is a domain-specific implementation of IML. (Ames, Koons & Sall, 1999b)

- ESO and CDS have defined another DTD for tables, for the JSky project. JSky mainly aims at being a repository of free software in astronomy, based on Java Beans. (Brighton & Albrecht, 1999)

- Gemini 8m Telescopes defined a DTD for observatories. Defined initially for the Gemini Observatory, it could now be used for other observatories.

- Caltech defined a DTD for scientific data objects in general, for the Digital Sky project. This DTD is surprisingly short (about 30 lines). (Caltech 1999)

Collaboration has begun to unify the DTD for tables, or at least sub-parts of this DTD. Two important meetings in this regard held at the workshop *From Information to Knowledge Using Astronomical Databases* in June 1999, and at the conference *ADASS IX* in October 1999, resulting in a first draft (Ochsenbein 1999). It would have been easier to defined the DTD together from the start, because we now have to rethink and discuss about all the decisions we already took for the DTDs. Some of these decisions, like choosing to use uppercase or lowercase for element names, cannot be defended easily, but we have to decide on one solution in order to have compatible DTDs. On the other hand, the separate development of the initial DTDs ensures that everybody has the best DTD for his own needs, and the new unified DTD we are trying to define is based on the best choices.

The ISAIA project

ISAIA (Interoperable Systems for Archival Information Access) (Hanish 1999) is a project to develop an interdisciplinary data location and integration service for space science. While location of documents in astronomy on the web is already achieved with AstroBrowse, a system that lets a user enter a coordinates query and use HTML links to look at the results on different servers, there is currently no way to use metadata going deeper than attribute-value information about documents with a URL (today HTML documents). As a result, the documents returned by AstroBrowse sometimes contain the text “no match found”, and the user has to open these documents to know that nothing has been found. Scientific metadata is usually not available as documents, as it is for library metadata for instance. Thus, a record-level access is needed to allow a good selection of scientific data : a user might want to make a program to automatically search for objects meeting certain conditions, with a specific procedure to do the test, on different servers.

The ISAIA project tries to define a common framework for search and information retrieval. It should allow complex queries, but also allow providers to use a simplified interface if they want to. Because of the advanced features required, the internal framework should be at least as advanced and flexible as, for example, the Emerge framework. And (as with the Emerge framework) different levels of interoperability must be supported. The minimum level of interoperability can be compared to the existing level with AstroBrowse, which can be achieved at a very low cost for the provider.

A meta-profile

Today, different profiles are used by data providers. A system like Emerge is able to map the different attributes of the profiles to a meta-profile, so that they are used in a unified way. However, some attributes are just impossible to map exactly one with another : they can have, for example, a similar meaning, but not exactly the same, which is an issue for scientific metadata. Of course, if all data providers were using the same profile, the interoperability would be greatly enhanced, but they would have to use the profile internally to avoid all the problems of profile mapping. The ISAIA project has begun the definition of a profile for all space science (thus for many different disciplines using usually different metadata). This big profile is composed of a generic profile for all the disciplines, and a number of sub-profiles defined by the different communities, trying to avoid redundancy. It could be used as the meta-profile in Emerge, so that the mapping is done easily by data providers : they would just have to create a configuration file to associate their attributes with the ones of the meta-profile.

Implementation of a profile

A language (or different languages) will be chosen to implement the meta-profile. Ideally, the profile should be independent of the implementing languages, so that it can be implemented in any language. The most important, for the interoperability, is the profile, not the language. In practice, however, a profile is often defined with the idea of a specific language to implement it. The attributes (or elements) do not have the same organisation (and sometimes meaning) for relational databases and XML documents : for instance, a short substructure would be implemented as a sub-element or a list of attributes for an element in XML, while it would be implemented in relational databases as a different table, with a key to make the link : these different organisations often lead to different choices for the meaning of the attributes.

XML is today the preferred language to implement metadata profiles.

Role of AML

AML was intended to serve as an example of the usefulness of XML for astronomical (and other) metadata. Now that XML is widespread, AML is promoting a unifying DTD for astronomical metadata : its applications (like the browser and the maps) are using the same DTD for different kinds of data. If providers were using incompatible DTDs, a user wanting to use data from different providers would have to define a new DTD (or use AML) and map the different elements between the DTDs : in practice, most users would abandon their project quickly. Various degrees of compatibility are possible, starting with a common syntax for a few elements. Whole parts of DTDs could also be the same, even if the whole DTD is different. However, most users (or programmers of applications using data from different sources) will want a single DTD. Providing AML documents does not mean using them internally : a provider can very well use another DTD, and transform it into AML for data exchange. But this transformation should be done by the people having the best knowledge of the semantics involved. If this is done by other people (for example, computer scientists instead of the initial researchers), the data must have a maximum of metadata to explain clearly the semantics. And the initial creation of the metadata would be best done… with an application creating XML documents : a format used for data exchange can become widespread for other tasks.

CONCLUSIONS

Many studies in astronomy require the use of data coming from different sources. Unfortunately, these data cannot be obtained in a homogeneous way, which limits the possibilities for such studies. Some specific aspects of astronomical data make IR (information retrieval) in this field unique, which leads to specific or flexible solutions. The massive volume of the data is one such specificity, and massive data need good metadata to be handled efficiently. Also, since universal identifiers rarely exist for data in astronomy, many IR techniques based on universal identifiers cannot apply here. Because of the peculiarities of scientific data, flexible data formats should be used to store or exchange them. Today, different archives tend to use different file formats, and enable access to the data (and sometimes the metadata) in different ways. Most archives contain large survey data, but, because it is still difficult for astronomers to bring their data online, it is difficult to find online data from small, specific projects.

The internet has become a uniform platform for accessing the information, and thus improved IR noticeably. However, there are still problems, due to the way the information is currently published and processed on the internet. The data are published with HTML, which mixes the data itself with changing user interface information: it is impossible to reuse automatically the data with HTML. Most processes are conducted with CGI programs, that only take name/value pairs as an input, and a document (possibly of any format) as an output: this extremely limits IR tasks (for instance, there is no standard way for a CGI program to reply that a result matching the query was found or not). Another lasting problem is the lack of standards to encode the information and the queries, and to handle distributed data. Because automatic IR is still limited today (one could say that HTML has even slowed it down), KDD is still in its infancy. Most applications in astronomy have used only articles, which are not scientific data. Because of its metadata complexity, scientific data has not been used much for KDD, although it should allow important research developments.

In order to improve information retrieval in astronomy, data structures and software structures were studied, with a particular focus on XML. After a brief review of existing IR techniques, a proposition was made to improve it. Following that is a description of AML, a markup language for applications in astronomy designed as a result of this proposition. Based on AML, a knowledge discovery application was described, the AML maps, to cluster AML documents. A few search systems have also been studied, and improved to make use of AML documents. More developments are ongoing, to unify metadata in astronomy.

Data can be divided into unstructured, structured and heterogeneous data. Traditional databases are well suited for structured data, but cannot be used to represent heterogeneous or unstructured data. XML can be used for all kinds of data, so it provides a unifying exchange format : databases can easily export their data into XML.

The limits of HTML (to represent information with semantics) and SGML (to be used easily on the internet) lead to the recent definition of XML, and a number of associated standards like SAX, DOM, XSL or XLink. To get all the benefits from XML, care should be taken to respect these existing standards, and new standards have to be created. OODBMS (Object-Oriented Database Management Systems) seem to be well suited for storing XML documents, and they allow various processing on the documents. However, a new database structure, based on the tree structure of XML documents, would be useful.

From the early definition to the widespread acceptance of XML, a complete revolution occurred, not only in the way information is published on the internet, but more generally in the support for explicit data formats and metadata. At the beginning of this thesis work, even though XML supporters were enthusiastic, it wasn't clear whether the industry (especially the small companies) would support a standard requiring a stricter attitute for the definition of a data format . In fact, many companies are now using XML without DTDs, which reduces its meaning and value (this is probably the down side of its popularity: XML is now sometimes being used because of its name rather than its advantages).

Thanks to hardware improvements, computer science has progressed by using more software structures, often requiring more CPU and memory. The latest developments are based on distributed objects : object-oriented pieces of software able to communicate on the internet with a standard protocol. The association between an object class and an XML DTD is particularly useful to encapsulate both the software and the data with a clear interface.

Distributed frameworks are an important idea to improve IR. Even better, distributed frameworks of objects _as opposed to documents_ let applications focus on the data content and the relationships between the objects, freeing them from formatting problems. With them, auto-organisation and automatic classification applications can be carried out easily. The object-oriented concept has already been successfully applied to data archiving and browsing, but was still to be applied to IR and KDD : the experiments realised tried to achieve this goal.

A limitation to XML is that binary data cannot be included into XML documents. Scientific data is often in a binary format because size is an issue, and metadata is usually included in the same file. A number of workarounds can fortunately be used to solve this problem. AML, an XML language, was designed to handle many different kinds of data used in astronomy, and is separated into a number of objects : astronomical objects, tables, images, articles, people and projects. This demonstrates the possibility to define heterogeneous (including scientific) data and metadata in the same language, to ease IR and KDD tasks. It is also possible to customise AML by adding institute-specific tags. Typically, a data provider can divide its data into public and private information, define a DTD for private information, use AML for public information (improving the AML DTD if necessary), and use AML to store or export data. This is what has been done for the ADIL (Astronomy Digital Image Library), for example. A browser for AML documents was designed and implemented in Java, to display any AML document (with a specific code for each AML object), and use the links between them as easily as in a web browser. Since it was created before many developments in XML, it is using a custom XML parser and a custom interface for a tree in memory: it could still be rewritten to make use of the new standards, SAX and DOM.

A KDD application of AML was designed and implemented : the AML maps. This is a clustering of AML documents, using their links and their keywords, with a semantic map user interface. Thanks to the XML formatting of the documents, it can also make use of any information inside them, for instance to calculate similarities between the documents. Two programs were developed for the AML maps, both with a graphical user interface. The first is for the clustering, accepting the different parameters and a way to select an initial document or a collection of documents. With the use of threads to download the documents simultaneously, it can create simple AML maps in about 30 seconds (this time should decrease until real interaction is possible, with the progress of the internet speed). It also saves a map in an XML file. The second program reads these files to display the results with a semantic map user interface and simple search capabilities. When requested, it displays selected AML documents with the AML browser. The AML maps were compared to other techniques such as the Kohonen maps, and an evaluation began for the clustering algorithm, a modified noising partitioning algorithm. It seems that the AML maps provide a better clustering and user interface system with some data, such as collections of heterogeneous, linked documents (which is not surprising, since it was designed for these particular data). In the future, more AML maps making use of the scientific information in the AML documents could be created, and this could be a powerful tool for some projects, even when they don't use data coming from different sources.

Three distributed search frameworks have been studied and extended to make use of AML documents : Harvest, Ingrid and Emerge. Each framework could then be used to index automatically and query AML documents efficiently. To summarise the differences, Harvest is indexing resources automatically, while Ingrid is creating links between individual resources in such a way that clusters of term combinations are formed, and Emerge is based on Z39.50 to be able to handle complex queries. Being complementary, Harvest and Emerge could be unified in a new framework in the future. Another evolution will consist in handling queries, taking advantage of XML particularities, with the future XQL language.

Finally, efforts were made to promote XML to the astronomical community, and apply the new developments in existing data centres. The adoption of new standards is one of the most important ways to improve information retrieval, but it is a slow process. New standards are being defined today, as part of the ISAIA project or just related to XML in astronomy, and this thesis was an important experiment to demonstrate possible choices for standards and technical solutions. It also demonstrates that KDD applications are already possible without a consensus on standards in the astronomical community.

The solutions provided by this thesis make a new basis for IR, with object-oriented XML documents linked to object-oriented software and distributed search frameworks. The applications for this framework are not limited to the astronomy field, but astronomy was a good test case to develop it, because of the various kinds of data handled in astronomy and the complexity of IR tasks.

REFERENCES

Note : all the web documents' URLs could be retrieved in 1999. The date of creation is usually unknown, so the year given here is usually 1999.

Abreu, F.B. and Melo, W. (1996). Evaluating the Impact of Object-Oriented Design on Software Quality, *Proc. 3rd International Software Metrics symposium*, IEEE Computer Society Press, Berlin, Germany.

ADS (1994), *Restructuring of the ADS*, 30 August 1994, http://adsdoc.harvard.edu/abs-doc/steve_announce.html

ADS (1999), *ADS : The NASA Astrophysics Data System*, http://adswww.harvard.edu/

Van Alstyne, Marshall, Brynjolfsson, Erik and Madnick, Stuart (1995). Why not one big database ? Principles for data ownership, *Decision Support Systems*, 15 n 4, 267-284, Dec. 1995.

Ansari, S.G., Giommi, P., Stokke, H. and Martinez, A. Preite (1993). Homogeneous Access to Astronomical Data - The ESIS Experience, *Handling and Archiving Data from Ground-based Telescopes*, April 21-23 1993, Trieste, Italy, eds. M. Albrecht and F. Pasiam.

Ansari, S.G., Donzelli, P., Simonsen, E., Ochsenbein, F. (1996). Transfer of ESIS to Scientific Institutes, *ESA Bulletin Nr 86*, May 1996, http://esapub.esrin.esa.it/bulletin/bullet86/ansa86.htm

Ames, Troy, Koons, Lisa and Sall, Ken (1999a). *XML for Astronomy*, Goddard Space Flight Center and Century Computing, http://pioneer.gsfc.nasa.gov/public/xml/

Ames, Troy, Koons, Lisa and Sall, Ken (1999b). *Astronomical Instrument Markup Language*, Goddard Space Flight Center and Century Computing, http://pioneer.gsfc.nasa.gov/public/aiml/

Birtwistle, G.M., Dahl, O.-J., Myhrhang, B. and Nygaard, K. (1973). *Simula Begin*, Van Nostrand Reinhold, New York.

Borne, Kirk D., Shaya and Edward J. (1999). *ADC XML Resources*, Goddard Space Flight Center, http://adc.gsfc.nasa.gov/xml/

Bowman, C.M., Dharap, C., Baruah, M., Camargo, B., and Potti, S. (1994). A file system for information management, *Proceedings of the Conference on Intelligent Information Management Systems*, June 1994.

Bowman, C.M., Danzig, P.B., Hardy, D.R., Manber, U., Schwartz, M.F. , Wessels, D.P. (1995). *Harvest : A Scalable, Customizable Discovery and Access System*, Technical Report CU-CS-732-94, Department of Computer Science, University of Colorado - Boulder, March 1995.

Brighton, Allan, Albrecht, Miguel (1999). *JSky Catalog Representation in XML*, ESO, http://archive.eso.org/JSky/catalog-xml.html

Brunner, R.J., Ramaiyer, K., Szalay, A., Connolly, A.J., Lupton, R.H. (1995). An Object-Oriented Approach to Astronomical Databases, *Astronomical Data Analysis Software and Systems IV*, ASP Conference Series, 77, eds. R.A. Shaw, H.E. Payne, and J.J.E. Hayes, 169.

Caltech (1999), *Extensible Scientific Interchange Language*, http://www.cacr.caltech.edu/SDA/xsil/

Carroll, J.D. and Pruzansky, S. (1975). *Fitting of hierarchical tree structure (HTS) models, mixtures of HTS models, and hybrid models, via mathematical programming and alternating least squares*, U.S.-Japan Seminar on Theory, Methods and Applications of Multidimensional Scaling and Related Techniques, San Diego.

Carroll, J.D. and Pruzansky, S. (1980). *Discrete and hybrid scaling models*, in Similarity and Choice, eds. E.D. Lanterman and H. Feger, Bern : Hans Huber, 108-139.

CDS (Centre de Données astronomiques de Strasbourg) (1999a). *VizieR*, http://vizier.u-strasbg.fr/cgi-bin/VizieR

CDS (1999b). *Development Site of the GLU*, http://simbad.u-strasbg.fr/glu/glu.html

CDS (1999c). *SIMBAD : Set of Identifications, Measurements, and Bibliography for Astronomical Data*, http://simbad.u-strasbg.fr/Simbad

CDS (1999d). ASU : Astronomical Server URL, http://vizier.u-strasbg.fr/doc/asu.html

Centre de Recherche Astronomique de Lyon (CRAL) (1999). *LEDA*, http://www-obs.univ-lyon1.fr/leda/home_leda.html

CGI, Common Gateway Interface, *de facto* standard, with a description maintained by the NCSA Software Development Group, http://hoohoo.ncsa.uiuc.edu/cgi/overview.html

Clark, J. (1999). *Jade*, http://www.jclark.com/jade/

Corter, J.E. and Tversky, A. (1986). Extended similarity trees, *Psychometrika*, 51, 429-451.

Cox, Brad J. (1986). *Object Oriented Programming - An Evolutionnary Approach*, Addison-Wesley, Reading, MA.

Demleitner, M., Eichhorn, G., Grant, C.S., Accomazzi, A., Murray, S.S. and Kurtz, M.J. (1999). ADS in a Nutshell, *American Astronomical Society Meeting,* 194, #103.05.

Dublin Core Directorate (1999). *Dublin Core Metadata Initiative*, http://purl.oclc.org/dc/

Egret, D. (1994). In the Jungle of Astronomical On-line Data Services, *Astronomical Data Analysis Software and Systems III*, ASP Conference Series, 61, eds. Dennis R. Crabtree, R.J. Hanish and J. Barnes, 14.

Egret, D. and Albrecht, A. (1995), Preface, *Information & On-line data in Astronomy*, Kluwer Academic Publishers, ed. Daniel Egret and Miguel A. Albrecht.

Eichhorn, G. (1994). An Overview of the Astrophysics Data System, *Experimental Astronomy*, v.5, 205-220.

Fernique, P., Ochsenbein, F., and Wenger, M. (1998). CDS GLU, a Tool for Managing Heterogeneous Distributed Web Services, ASP Conference Series, 145, *Astronomical Data Analysis Software and Systems VII*, eds. R. Albrecht, R.N. Hook and H.A. Bushouse (San

Francisco : ASP), 466.

FITS, *Definition of the Flexible Image Transport System (FITS)*, Standard, NOST100-1.0, NASA/Science Office of Standards and Technology, code 633.2, NASA Goddard Space Flight Center, Greenbelt, Maryland.
Flexible Image Transport System, http://www.cv.nrao.edu/fits/

Francis, P., Kambayashi, T., Sato, S. and Shimizu, S. (1995). Ingrid : A Self-Configuring Information Navigation Infrastructure, *Fourth International World Wide Web Conference*, Boston.

Garey, M.R. and Johnson, D.S. (1979). Computers and Intractability, *a Guide to the Theory of NP-Completeness*, Freeman, San Francisco.

GILS Maintenance Agency (1999),*GILS*, http://www.gils.net/

Giommi, Paolo and Ansari, Salim G. (1995). The European Space Information System (ESIS), *Information & On-line data in Astronomy*, Kluwer Academic Publishers, eds. Daniel Egret and Miguel A. Albrecht.

Goldman, R., McHugh, J., Widom, J. (1999). From Semistructured Data to XML : Migrating the Lore Data Model and Query Language, *Proceedings of the 2nd International Workshop on the Web and Databases (WebDB'99)*, Philadelphia, Pennsylvania, June 1999.

Grossman, J.W. and Ion, P.D.F. (1995). On a portion of the well-known collaboration graph, *Congressus Numerantium*, 108, 129-131,
http://www.acs.oakland.edu/~grossman/erdoshp.html

GSFC (Goddard Space Flight Center) (1999). *ADC : Astronomical Data Center*, http://adc.gsfc.nasa.gov/adc.html

Hanish, R. (1999), Distributed Data Systems and Services in Astronomy and the Space Sciences, ASP Conference Series, *Astronomical Data Analysis Software and Systems IX*, eds. Dennis Crabtree, Nadine Manset and Christian Veillet, to be pulished.

Hardy, D.R. and Schwartz, M.F. (1995). *Harvest User's Manual*, Technical report, February 1995, Version 1.1.

Harvest Web Indexing (1999). http://www.tardis.ed.ac.uk/harvest/

Heikkila, C., McGlynn, T.A. and White, N.E. (1999). Astrobrowse : a Web Agent for Querying Astronomical Databases, ASP Conference Series, 172, *Astronomical Data Analysis Software and Systems VIII*, eds. D.M. Mehringer, R.L. Plante and D.A. Roberts (San Francisco : ASP), 221.
AstroBrowse at HEASARC, http://heasarc.gsfc.nasa.gov/ab/

ICEsoft S/A (1999). http://www.icesoft.no/

IPAC (1999). *NED : NASA/IPAC Extragalactic Database*, http://nedwww.ipac.caltech.edu/

IRIT (1999). *Tétralogie*, http://atlas.irit.fr/

ISO/IEC (1997). *HyTime*, ISO/IEC JTC1/SC18/WG8 N1920 Second edition : 1997-08-01, http://www.ornl.gov/sgml/wg8/docs/n1920/html/n1920.html

Jackson, R.E., Adorf, H.-M., Egret, D., Heck, A., Koekemoer, A., Murtagh, F. and Wells, D.C.

(1995). AstroWeb -- Internet Resources for Astronomers, *Astronomical Data Analysis Software and Systems IV*, ASP Conference Series, 77, eds. R.A. Shaw, H.E. Payne, and J.J.E. Hayes, 52. *The Astroweb Consortium*, http://cdsweb.u-strasbg.fr/astroweb/consortium.html

Jayson, Raymond (1999). *Java and Distributed Object Systems*, Corbis Corp., http://www.seajug.org/html/DisObSys/TOC.htm

Johnson, D.S., Aragon, C.R., McGeoch, L.A. and Sheron, C. (1989). Simulated annealing for optimization : An experimental evaluation. Part I, graph partitioning, *Oper. Res.*, 37 (6), 865-892.

JXML, Inc (1999). *Coins*, http://www.jxml.com/coins/

Kaski, S., Honkela, T., Lagus, K., and Kohonen, T. (1998). WEBSOM - self-organizing maps of document collections, *Neurocomputing*, 21, 101-117.

Keahey, Kate (1999). *A Brief Tutorial on CORBA*, http://www.cs.indiana.edu/hyplan/kksiazek/tuto.html

King, Joseph H. (1998). *White paper on NASA science data retention*, NSSDC, http://nssdc.gsfc.nasa.gov/nssdc/data_retention.html

Kurtz, Michael J. (1995). Giant Shoulders : Data and Discovery in Astronomy, *Publications of the Astronomical Society of the Pacific*, 107, 776 (PASP Homepage).

Lee, J., Kurtz, M.J. and Dubin, D. (1999a). Lexical and co-occurrence evidence for subject vocabulary reconciliation in ADS databases, ASP Conference Series, 172, *Astronomical Data Analysis Software and Systems VIII*, eds. D.M. Mehringer, R.L. Plante and D.A. Roberts, ISBN : 1-886733-94-5, 287.

Lee, Jonghoon and Dubin, David (1999b). Context-sensitive vocabulary mapping with a spreading activation network", *Proceedings on the 22nd annual international ACM SIGIR conference on Research and developments in information retrieval*, August 15-19, Berkeley, CA USA, 198-205.

Lesteven, S., Bonnarel, F., Dubois, P., Egret, D., Fernique, P., Genova, F., Murtagh, F., Ochsenbein, F., Wenger, M. (1998). Information Extraction : New Developments in Astronomical Information Retrieval for Electronic Publications, *Library and Information Services in Astronomy III*, ASP Conference Series, 153, eds. U. Grothkopf, H. Andernach, S. Stevens-Rayburn and M. Gomez, Electronic Editor : H. E. Payne.

Louys, M., Starck, J.L., Bonnarel, F. and Murtagh, F. (1999). Astronomical Image Compression, *Astronomy & Astrophysics supplement series*, 136, 579-590.

Manber, U. and Wu, S., "Glimpse : a tool to search through entire file systems". *Proceedings of the USENIX Winter Conference*, 23-32, January 1994.

Megginson, D., *SAX*, 1999, http://www.megginson.com/SAX/

Mothe, J., Egret, D., Dkaki, T., Dousset, B. (1998). Analysis of Evolutionary Trends in Astronomical Literature using a Knowledge-Discovery System : Tétralogie, *Library and Information Services in Astronomy III*, Proceedings of a conference held in Puerto de la Cruz, Tenerife, Spain, April 21-24, 1998, eds. U. Grothkopf, H. Andernach, S. Stevens-Rayburn and M. Gomez, ISBN 1-886733-73-2, ASP Conference Series, 153, 69.

Murray-Rust, P. (1999). *Jumbo*, http://www.xml-cml.org/jumbo/

NCSA (1999a). *Emerge*, http://emerge.ncsa.uiuc.edu/

NCSA (1999b). *The Horizon Image Data Browser Package*, http://imagelib.ncsa.uiuc.edu/imagelib/andJava.html#horizon

NTT (Nippon Telegraph and Telephone Corporation) (1999). *Ingrid*, http://www.ingrid.org/

Object Management Group (1999). http://www.omg.org/

Ochsenbein, F., Genova, F., Egret, D., Bourekeb, I., Sadat, R., Ansari, S.G., Simonsen, E. (1995). A new research tool to retrieve data from astronomical catalogues and tables, *American Astronomical Society Meeting*, 187, #91.03.

Ochsenbein, F. (1999). Describing Astronomical Catalogues and Query Results with XML, initially defined at the workshop *From Information to Knowledge Using Astronomical Databases* in Strasbourg, in June 1999, http://vizier.u-strasbg.fr/doc/astroxml.htx

Orfali, R., Harkey, D. and Edwards, J. (1996). *The Essential Distributed Objects Survival Guide*, John Wiley & Sons, Inc.

Pasian, F. and Smareylia, R. (1998). Querying by Example Astronomical Archives, ASP Conference Series, 145, *Astronomical Data Analysis Software and Systems VII*, eds. R. Albrecht, R.N. Hook and H.A. Bushouse (San Francisco : ASP).

Plante, Raymond L., Crutcher, Richard M., McGrath, Robert E. (1998). The NCSA Astronomy Digital Image Library : From Data Archiving to Data Publishing, submitted for publication in *Future Generation Computer Systems* (Elsevier), September 1998.

Poinçot, P., Lesteven, S., and Murtagh, F. (1998). A spatial user interface to the astronomical literature, *Astronomy and Astrophysics Supplement*, 130, 183-191.

Poinçot, P., Lesteven, S., and Murtagh, F. (1999). Maps of information spaces : assessments from astronomy", *Journal of the American Society for Information Science*, submitted.

Rafanelli, M. and Shoshani, A. (1990). STORM : A Statistical Object Representation Model, Michalewicz, Z. (Ed.) : *Proceedings of the 5th International Conference on Statistical and Scientific Database Management* (5SSDBM, Charlotte, N.C., April 3-5), 14-29.

Rein, Lisa (1998). *Handling Binary Data in XML Documents*, XML.com, http://www.xml.com/xml/pub/98/07/binary/binary.html

Rising, Linda S. and Calliss, Frank W. (1994). Information-hiding metric, Journal of Systems & Software, 26 n 3, 211-220, September 1994.

Schmitz, M., Helou, G., Madore, B.F., Corwin, H.G., Benett, J., Lague, C., Sun, H. and Wu, X. (1995a). NED : A Behind-the-Scenes Look, *American Astronomical Society Meeting*, 186, #02.04.

Schmitz, M., Helou, G., Dubois, P., LaGue, C., Madore, B., Corwin Jr., H.G. and Lesteven, S. (1995b). NED and SIMBAD Conventions for Bibliographic Reference Coding, *Information & On-line Data in Astronomy*, eds. D. Egret & M.A. Albrecht (Dordrecht : Kluwer Acad. Publi.), 271.

Sharon, I. and Hudry, O. (1993). The noising method : a new method for combinatorial optimization, *Oper. Res. Lett.*, 14, 133-137.

Shaya, E., Gass, J., Blackwell, J., Thomas, B., Holmes, B., Cheung, C. (1999). XML at the ADC : Steps to a Next Generation Data Repository, ASP Conference Series, *Astronomical Data Analysis Software and Systems IX*, 1999, eds. Dennis Crabtree, Nadine Manset and Christian Veillet, to be pulished.

Shield, J. (1987). Partitioning concurrent VLSI simulation programs onto a multiprocessor by simulated annealing, *IEE Proc.*, 134 (1), 24-30.

Songline Studios (affiliate of O'Reilly & Associates, Inc.) (1999). *Perl, Practical Extraction and Report Language*, http://www.perl.com/

Stanford University Database Group (1999). *Lore*, http://www-db.stanford.edu/lore

Sudhakar, V. and Siva Ram Murthy, C. (1997). A modified noising algorithm for the graph partitioning problem, Integration, *the VLSI journal*, 22, 101-113.

Teschke, Michael, Ruf, Thomas and Lehner, Wolfgang (1995). Data Management in Scientific Computing : A study in Market Research, *Proceedings of ADB 95*, Santa Clara, California, University of Erlangen-Nuremberg,
http://www6.informatik.uni-erlangen.de/papers/view.pl.cgi?key=LeRT95a&typ=.ps

W3C (1999a). *XML 1.0 Recommendation*, 10 February 1998, http://www.w3.org/TR/REC-xml
Extensible Markup Language (XML), http://www.w3.org/XML/

W3C (1999b). *World Wide Web Consortium (W3C)*, http://www.w3.org/

W3C (1999c). *XLink Working Draft*, http://www.w3.org/TR/WD-xlink

W3C (1999d). *RDF Recommendation*, http://www.w3.org/TR/REC-rdf-syntax/

W3C (1999e). *DOM*, http://www.w3.org/DOM/

W3C (1999f). *XML Schema*, http://www.w3.org/TR/NOTE-xml-schema-req

W3C (1999g). *XML-QL*, http://www.w3.org/TR/NOTE-xml-ql/

W3C (1999h). *XPath*, 13 August 1999, http://www.w3.org/TR/xpath

W3C (1999i). *XSLT*, 13 August 1999, http://www.w3.org/TR/WD-xslt

W3C (1999j). *Extensible Stylesheet Language (XSL) Specification*, W3C Working Draft, 21 Apr 1999, http://www.w3.org/TR/1999/WD-xsl-19990421/

Wenger, M., Fernique, P., Genova, F., Bartlett, J., Bienayme, O., Bonnarel, F., Dubois, P., Egret, D., Jasniewicz, G., Lesteven, S., Monier, R. and Ochsenbein, F. (1996). SIMBAD on the Web and Links to other Services, *American Astronomical Society Meeting*, 189, #06.02.

Z39.50 Maintenance Organization (1999). *Welcome to the Library of Congress Maintenance Agency page for International Standard Z39.50*, http://www.loc.gov/z3950/agency/

PUBLICATIONS

Guillaume, Damien and Murtagh, Fionn (1999). Clustering of XML documents, submitted for publication in *Computer Physics Communications*, for the proceedings of the workshop *From Information to Knowledge Using Astronomical Databases*, June 1999.

McGrath, Robert E., Futrelle, Joe, Plante, Ray and Guillaume, Damien (1999). Digital Library Technology for Locating and Accessing Scientific Data, *Proceedings of ACM Digital Libraries 99*, August 1999, 188-194.

Guillaume, Damien and Murtagh, Fionn (1999). An Application of XML and XLink Using a Graph-Partitioning Method and a Density Map for Information Retrieval and Knowledge Discovery, ASP Conference Series, 172, *Astronomical Data Analysis Software and Systems VIII*, eds. D. M. Mehringer, R. L. Plante, & D. A. Roberts (San Francisco : ASP), 278.

Murtagh, Fionn and Guillaume, Damien (1998). Distributed Information Search and Retrieval for Astronomical Resource Discovery and Data Mining, *Library and Information Services in Astronomy III*, ASP Conference Series, 153, eds. U. Grothkopf, H. Andernach, S. Stevens-Rayburn and M. Gomez, Electronic Editor : H. E. Payne.

ACKNOWLEDGMENTS

◇ PhD grant from the French Ministère de l'Education Nationale, de la Recherche et de la Technologie, for the Observatoire Astronomique de Strasbourg (Université Louis-Pasteur, France)

◇ Hosted by the Computer Science Department of the Universty of Ulster in Derry (Northern Ireland), from September 1997 to January 1999

◇ Hosted by the Computer Science Department of Queen's University of Belfast (Northern Ireland), from February 1999 to June 1999

◇ Hosted by the Astronomy Department of the University of Illinois at Urbana-Champaign (USA), after July 1999

APPENDICES

Appendix I: Class xml.XMLTree

```
java.lang.Object
   |
   +----xml.XMLTree
```

public class **XMLTree**
extends Object

A simple XML tree, containing for each node:

- a tag
- a list of attributes
- a value
- a link to the first child
- a link to the next brother
- a link to the father

A child is always added at the end of the brothers-list (with procedure addChild), but a new brother is inserted as the next brother to the one specified (with procedure addBrother)

FIELDS

- attr
- father
- first_child
- next_brother
- tag
- val

CONSTRUCTOR

- XMLTree(XMLTree, String, Vector, String)
  XMLTree constructor, with the father, the tag, the attributes (null is a valid value), and the value.

METHODS

- addBrother(String, Vector, String)
  creates a new XMLTree with the parameters, and calls addBrother(new_brother) with it
- addBrother(XMLTree)
  links the specified XMLTree as the next brother for this node
- addChild(String, Vector, String)
  creates a new XMLTree with its tag,attributes and value, and calls addChild(new_child) to link it as a child
- addChild(XMLTree)
  adds the specified child at the end of the brother-to-brother list
- getAttVal(String)
  returns the attribute value corresponding to the given name
- getChildrenAt(int)

returns the children at index n in the tree, n beginning at 0

- getChildrenCount()
  returns the number of 1-depth children for this tree
- getLeavesCount()
  returns the number of leaves for this tree
- getNode(String)
  returns the first first-level tag in the tree having specified name
- writeTree(OutputStream, String)
  writes an XML output corresponding to the tree

Appendix II : Class xml.Parser_XML

```
java.lang.Object
   |
   +----xml.Parser
           |
           +----xml.Parser_XML
```

public class **Parser_XML**
extends Parser

A sample XML parser using the Parser class, and creating an XML tree in memory (with class XMLTree). As Parser, it does not support entities.

Except when an error occurs, it does not output any text. (so, the outstr variable is useless for this class)

The parsing is stopped when the first tag read is closed. So, only the first tag and its descendants are read and added to the tree.

See Also:
Parser, XMLTree

CONSTRUCTORS

- Parser_XML(InputStream, OutputStream, XMLTree)
- Parser_XML(URL, OutputStream, XMLTree)

METHODS

- attribute(String, String)
  Adds an attribute to the last attributes list
- close_tag(String)
  Store the data in the node, stops the parsing if the top-node is closed, and initialises the variables for the next tag.
- data_in_tag(String)
  records the last data in a tag, and substitutes some HTML entities.
- error(int, String)
  Simply prints an error to standard output...
- open_tag(String, int)
  Adds a node to the XML tree, with the last attributes parsed

Appendix III: AML DTD

```
<!-- AML (astronomical markup language) DTD, version 0.41 -->
<!-- last modified: 1999-08-30 -->

<!ELEMENT AML (METADATA, (AOBJECT | ARTICLE | TABLES | TABLE |
               IMAGE | PERSON | PROJECT)+) >

<!ATTLIST AML
    xmlns          CDATA                               #IMPLIED >

<!ELEMENT METADATA   (AMLVER | SOURCE | TITLE | CREATOR | DATE | IDENTIFIER)+ >
<!-- informations about the AML file (not about its content) -->

<!ELEMENT AMLVER     (#PCDATA) >
<!-- version of AML. Format: NN.NN -->

<!ELEMENT SOURCE     (#PCDATA) >
<!-- origin of the file (server, institute, ...). Format: name. Searchable -->

<!ELEMENT TITLE      (#PCDATA) >
<!-- title of the file. Format: ASCII. Searchable -->

<!ELEMENT CREATOR    (#PCDATA) >
<!-- creator of the file. Format: no restriction (should be changed to
     PERSONNAME one day). Searchable -->

<!ELEMENT DATE       (#PCDATA) >
<!-- date of the file. Format: YYYY-MM-DD (ISO 8601). Searchable -->

<!ELEMENT IDENTIFIER (#PCDATA) >
<!-- unique identifier for the file. Format: URL or URN.
     examples of urn:
       urn:aml.org:format=AML:amlobj=AOBJECT:ident=MRK334
       urn:aml.org:format=HTML:amlobj=AOBJECT:ident=MRK334:server=NED:where=USA
     Searchable.
-->


<!ELEMENT AOBJECT   (IDENTIFIER?, IDENTS, (COORD | OBJTYPE | MORPHO | RADVELO |
                     PMOTIONS | DIM | MAG | SPEC | ORIANGL | SELIDENT | BHEXT |
                     NOTES)*, LMEASUREMENTS?, REFS?, LINKS?,
                     LINKIDENTS?, SPECIFIC?) >
<!-- astronomical object -->

<!-- identifier already defined. Here it identifies the AML object -->
<!ELEMENT IDENTS     (IDENT+) >
<!ELEMENT IDENT      (#PCDATA) >
<!-- astronomical identifier. Format: see
"http://vizier.u-strasbg.fr/cgi-bin/Dic". Searchable -->

<!ELEMENT COORD      (LON, LAT) >
<!-- Searchable -->
<!ATTLIST COORD
    csys           (galactic | ecliptic | equatorial)   #REQUIRED
    equinox        CDATA                               #REQUIRED
    epoch          CDATA                               #IMPLIED
    uncmaj         CDATA                               #IMPLIED
    uncmin         CDATA                               #IMPLIED
    uncang         CDATA                               #IMPLIED
```

```
    ref            CDATA                                 #IMPLIED>
<!-- equinox: year (integer) -->
<!-- format: see"http://cdsweb.u-strasbg.fr/iau-spec.html#sec3.2.1":
"Coordinates shall be specified as LLL.ll+BB.bb or LLL.ll-BB.bb for
 galactic coordinates, and as HHMMSS.ss+DDMMSS.s or HHMMSS.ss-DDMMSS.s
 for equatorial coordinates (without spaces); more or fewer
 fractional digits may be used as appropriate." -->
<!-- uncmaj/uncmin/uncang: 95% confidence ellipse (2.5 sigma) -->
<!-- ref: associated bibcode -->
<!ELEMENT LON        (#PCDATA) >
<!ATTLIST LON
    format         CDATA                                 #REQUIRED >
<!ELEMENT LAT        (#PCDATA) >
<!ATTLIST LAT
    format         CDATA                                 #REQUIRED >

<!ELEMENT OBJTYPE   (#PCDATA) >
<!-- see "http://newb6.u-strasbg.fr/~guillaum/types_sim". Searchable -->

<!ELEMENT MORPHO     (#PCDATA) >
<!-- can anybody help me finding a standard classification online ? -->

<!ELEMENT RADVELO   (#PCDATA) >
<!ATTLIST RADVELO
    unit          (kmps | mps | z)    #REQUIRED
    uncert        CDATA               #IMPLIED
    ref           CDATA               #IMPLIED>
<!-- Searchable -->

<!ELEMENT PMOTIONS   (#PCDATA) >
<!ATTLIST PMOTIONS
    unit          CDATA               #REQUIRED
    uncert        CDATA               #IMPLIED
    ref           CDATA               #IMPLIED>

<!ELEMENT DIM        (#PCDATA) >
<!-- Object dimensions. Format: major_axis x minor_axis -->
<!ATTLIST DIM
    unit          (deg | arcsec | arcmin)    #REQUIRED
    uncert        CDATA                      #IMPLIED >

<!ELEMENT MAG        (#PCDATA) >
<!ATTLIST MAG
    filter        (CDATA | U | B | V | R | I | unknown)    #REQUIRED >

<!ELEMENT SPEC       (#PCDATA) >
<!-- spectral class, for stars -->

<!ELEMENT ORIANGL   (#PCDATA) >
<!ATTLIST ORIANGL
    unit          (deg)                      #REQUIRED >
<!-- orientation angle -->

<!ELEMENT SELIDENT  (#PCDATA) >
<!-- selected identification (see element ident above). Searchable -->

<!ELEMENT BHEXT     (#PCDATA) >
<!-- Burstein-Heiles extinction in B mag -->

<!ELEMENT NOTES (#PCDATA) >
<!-- free text information about the object or the data. -->
```

```
<!ELEMENT LMEASUREMENTS   (MEASUREMENTS+) >
<!ELEMENT MEASUREMENTS    (MEASUREMENT+) >
<!ATTLIST MEASUREMENTS
    what          CDATA                      #REQUIRED
    title         CDATA                      #IMPLIED >
<!ELEMENT MEASUREMENT     (#PCDATA) >
<!ATTLIST MEASUREMENT
    unit          CDATA                      #REQUIRED
    uncert        CDATA                      #REQUIRED
    ref           CDATA                      #REQUIRED
    other         CDATA                      #IMPLIED >
<!-- list of measurements. this can be used instead of the other elements
     for the measurements, or for other data.
     for example:
     <LMEASUREMENTS>
     <MEASUREMENTS what="radvelo" title="radial velocity">
     <MEASUREMENT unit="kmps" uncert="9"
ref="1991RC3.9.C...0000d">6600</MEASUREMENT>
     <MEASUREMENT unit="kmps" uncert="40"
ref="1991RC3.9.C...0000d">6608</MEASUREMENT>
     <MEASUREMENT unit="z" uncert="unknown"
ref="1985ApJS...57..643D">0.022</MEASUREMENT>
     </MEASUREMENTS>
     <MEASUREMENTS what="photometry">
     <MEASUREMENT unit="mag" other="U (U_T)" uncert="0.18"
ref="1991RC3.9.C...0000d">14.47</MEASUREMENT>
     <MEASUREMENT unit="Jy km s^-1^" other="HI (21 cm line)" uncert="0.21"
ref="1989H&amp;RHI.C...0000H">1.07</MEASUREMENT>
     </MEASUREMENTS>
     </LMEASUREMENTS>
-->

<!ELEMENT REFS      (REF+) >
<!ELEMENT REF       (#PCDATA) >
<!-- a reference should always be a bibcode -->
<!-- deprecated: has been replaced by LINKIDENT -->
<!ATTLIST REF
    title         CDATA                      #IMPLIED >

<!ELEMENT LINKS     (LINK+) >
<!ELEMENT LINK      EMPTY >
<!ATTLIST LINK
    xml:link      CDATA                      #FIXED    "simple"
    href          CDATA                      #REQUIRED
    content-role  NMTOKEN                    #REQUIRED
    title         CDATA                      #IMPLIED >
<!-- links to other AML objects (XLink).
     see W3C working draft: http://www.w3.org/TR/WD-xlink -->
<!-- href can be a URL or a URN -->
<!-- content-roles for AOBJECT:
     "has_for_ref", "being_studied_by", "is_included_in", "other_data",
     "has_for_image" -->

<!ELEMENT LINKIDENTS (LINKIDENT+) >
<!ELEMENT LINKIDENT  EMPTY >
<!ATTLIST LINKIDENT
    amlobj        NMTOKEN                    #REQUIRED
    name          CDATA                      #REQUIRED
    role          NMTOKEN                    #REQUIRED
    title         CDATA                      #IMPLIED >
```

```
<!-- linkidents are also links to other AML objects, but with
     identifiers instead of URLs. Has replaced the tags:
     REF, LINKOBJ, AUTHORS/PERSONNAME, LINKTAB, LINKIMG
     e.g. <REF title="the end of HTML">1234</REF> became:
          <LINKIDENT amlobj="ARTICLE" name="1234" role="has_for_ref"
                     title="the end of HTML"/>

     LINKIDENTS elements might be replaced by LINKS elements, with
     an AML URN in the href attribute.
     This would provide better XLink compatibility and more flexibility.
-->

<!ELEMENT SPECIFIC (#PCDATA) >
<!-- here (#PCDATA) can be replaced by ANY and an institute-specific
     DTD can be added at the end of the AML DTD, to allow validation -->
<!-- institute-specific tags. must be used with namespaces. -->
<!ATTLIST SPECIFIC
    institute     CDATA                      #REQUIRED >



<!ELEMENT ARTICLE  (IDENTIFIER?, BIBCODE, (AUTHORS | ARTTITLE | PUBLTOWN |
                    PUBLICATION | PUBLDATE | NOFIRSTPAGE | NBPAGES | KEYWORDS |
                    SUMMARY | COPYRIGHT | LINKS | LINKIDENTS | SPECIFIC)* ) >
<!-- IDENTIFIER already defined -->
<!ELEMENT BIBCODE        (#PCDATA) >
<!-- Searchable -->
<!ELEMENT AUTHORS        (PERSONNAME+) >
<!ELEMENT PERSONNAME     (LASTNAME, FIRSTNAMEA?, FIRSTNAMEB?, FIRSTNAMEC?) >
<!ELEMENT LASTNAME       (#PCDATA) >
<!-- Searchable -->
<!ELEMENT FIRSTNAMEA      (#PCDATA) >
<!-- Searchable -->
<!-- firstnames can be abbreviated by the first character and a dot -->
<!ELEMENT FIRSTNAMEB     (#PCDATA) >
<!ELEMENT FIRSTNAMEC     (#PCDATA) >
<!ELEMENT ARTTITLE       (#PCDATA) >
<!-- Searchable -->
<!ELEMENT PUBLTOWN       (#PCDATA) >
<!ELEMENT PUBLICATION    (#PCDATA) >
<!-- Searchable -->
<!ELEMENT PUBLDATE       (#PCDATA) >
<!-- Searchable -->
<!ELEMENT NOFIRSTPAGE    (#PCDATA) >
<!ELEMENT NBPAGES        (#PCDATA) >
<!ELEMENT KEYWORDS       (KEYWORD+) >
<!-- Searchable -->
<!ELEMENT KEYWORD        (#PCDATA) >
<!ELEMENT SUMMARY        (#PCDATA) >

<!ELEMENT COPYRIGHT     (COPYGEN, COPYYEAR?, COPYHOLDER?) >

<!ELEMENT COPYGEN     (#PCDATA) >
<!-- general statement for the copyright -->

<!ELEMENT COPYYEAR     (#PCDATA) >
<!-- the copyright year -->

<!ELEMENT COPYHOLDER     (#PCDATA) >
<!-- the copyright holder -->
```

```
<!-- LINKS already defined -->
<!-- content-roles for ARTICLE: "is_ref_of", "has_for_author", "includes" -->
<!-- LINKIDENTS already defined -->
<!-- SPECIFIC already defined -->



<!ELEMENT TABLES       (IDENTIFIER?, TABSDESCR?, REF?, AUTHORS?, KEYWORDS?,
                        TABLE+, LINKS?, LINKIDENTS?, SPECIFIC?) >
<!-- IDENTIFIER already defined -->
<!ELEMENT TABSDESCR      (#PCDATA) >
<!-- Searchable -->
<!-- REF already defined (will be replaced by a LINKIDENT) -->
<!-- AUTHORS already defined -->
<!-- KEYWORDS already defined (Very Much Searchable!)-->
<!-- LINKS already defined -->
<!-- LINKIDENTS already defined -->
<!-- SPECIFIC already defined -->
<!-- content-roles for TABLES: "has_for_author", "includes", "has_for_ref" -->


<!-- Astronomical table -->
<!ELEMENT TABLE (IDENTIFIER?, NAME?, TITLE?, DESCRIPTION?, COOSYS?, FIELDS,
DATA?, LINKS?, SPECIFIC?) >
   <!ATTLIST TABLE
    ID            CDATA              #IMPLIED
    nbcols        CDATA              #IMPLIED
    nbrows        CDATA              #IMPLIED >
<!-- nbrows recommended when xlink is used for the data -->

<!-- IDENTIFIER already defined -->

<!-- 1) In TABLE tag: Catalog or table shortname
        ex: "GSC1.2
     2) In FIELD tag: field name
        ex: "RA(J2000)" -->
<!ELEMENT NAME (#PCDATA) >

<!-- 1) In TABLE tag: Catalog or table description (paragraph)
        ex: "This is an excerpt of the GSC1.2. This version was re-reduced
             with PPM catalogue; see more details about..."
     2) In FIELD tag: field description (paragraph) -->
<!ELEMENT DESCRIPTION (#PCDATA) >

<!-- 1) In TABLE tag: Catalog or table title in one line
        ex: "The HST Guide Star Catalog, Version 1.2 (Lasker+ 1996)"
     2) In FIELD tag: field description (paragraph)
        ex: "Right ascension in J2000, epoch of plate" -->

<!-- Coordinate system declaration -->
<!ELEMENT COOSYS EMPTY >
   <!-- ID       to refer it                   "myJ2000"
        system    eq_FK4|eq_FK5|ICRS|ecl_FK4|ecl_FK5|
                  galactic|supergalactic|xy|barycentric|geo_app
        equinox  the equinox
        epoch    the epoch
        x        Alternative for referring to RA
        y        Alternative for referring to DE
   -->
<!ATTLIST COOSYS
   ID          CDATA           #IMPLIED
```

```
   system
(eq_FK4|eq_FK5|ICRS|ecl_FK4|ecl_FK5|galactic|supergalactic|xy|barycentric|geo_ap
p)      "eq_FK5"
   equinox      CDATA           #IMPLIED
   epoch        CDATA           #IMPLIED
   x            CDATA           #IMPLIED
   y            CDATA           #IMPLIED >

<!-- TITLE already defined -->

<!ELEMENT FIELDS     (FIELD+) >

<!-- Definition of a field -->
<!ELEMENT FIELD (NAME?, TITLE?, DESCRIPTION?, VALUES?, LINK? ) >
   <!-- ID       To refer this element
        unit     Unit used for the field (see appendix)
        datatype Datatype of field value
                 F-float, D-double, I-integer, A-ascii
                 L-boolean (logical), E-exponential
        precision Precision of field value: number of significant digits
                 after the dot (ex: "3")
        width    number of characters          "8"
        format   indicate the format by a "%fmt" template (as "printf()"
                 in language C). Use for coordinate format, time and date
                 formats   (ex: "%RAh:%RAmd %DEd:%DEmd" - (see appendix)
        ref      Reference to the system for this field. For example it can be
                 a coordinate or a photometric system.
                 ex : #id("myJ2000")  -> The syntax comes from Xpointers
        name     Alternative to specify the name of the field if it uses
                 only ascii characters (see the NAME element for the
                 other possibility)
        ucd      Unified column descriptor    "POS_EQ_RA" (ESO/CDS work)
        type     To characterize the field
                 hidden   for fields used typically for server/client exchange
                 no_query for fields which specify some parameters
                          (e.g. the equinox of a coordinate system)
                 trigger  for fields which contain a parameter for an action
                          to execute (e.g. an image number to be passed to a
link)
        actualvalues Reference to the range or list of actual values
                 (it is an alternative to the <VALUES> entity in order to
                 be able to factorize them)
        legalvalues Reference to the range or list of legal values
                 (it's an alternative to the <VALUES> entity in order to
                 be able to factorize them)
   -->
   <!ATTLIST FIELD
   ID           CDATA          #IMPLIED
   unit         CDATA          #IMPLIED
   datatype     (F|I|D|E|A|L)  #IMPLIED
   precision    CDATA          #IMPLIED
   width        CDATA          #IMPLIED
   format       CDATA          #IMPLIED
   ref          CDATA          #IMPLIED
   name         CDATA          #IMPLIED
   ucd          CDATA          #IMPLIED
   type         (hidden|no_query|trigger) #IMPLIED
   actualvalues CDATA          #IMPLIED
   legalvalues  CDATA          #IMPLIED >

<!-- To specify ranges or lists of values
```

```
     In case of use in <FIELD> tag, the type attribute
     can be used to indicate the kind of values.
     The multiple attribute specifies that it is
     not an exclusive list of value, and in this case
     the separator value indicates the character to
     separate several values
        type      legal|actual
        multiple  yes|no
        separator a letter, by default "," -->
<!ELEMENT VALUES (MIN*, MAX*, OPTION*, NULL?) >
   <!ATTLIST VALUES
   ID           CDATA          #IMPLIED
   multiple    (yes |no)       "no"
   type        (legal|actual)  "legal" >

<!-- Min of a value range
    value        the min
    inclusive    yes|no  (default: yes) -->
<!ELEMENT MIN (#PCDATA) >
   <!ATTLIST MIN
   value        CDATA          #REQUIRED
   inclusive    (yes |no)      "yes" >

<!-- Max of a value range
    value        the max
    inclusive    yes|no  (default: yes) -->
<!ELEMENT MAX (#PCDATA) >
   <!ATTLIST MAX
   value        CDATA          #REQUIRED
   inclusive    (yes |no)      "yes" >

<!-- Item of list of values. The content of the
     OPTION tag can be not empty to indicate
     a corresponding sentence (ex: value="I/239"
     and content: "Hipparcos catalog" -->
<!ELEMENT OPTION (#PCDATA) >
   <!ATTLIST OPTION
   value        CDATA          #REQUIRED >

<!-- Specify the value for the null representation
    value        the null representation -->
<!ELEMENT NULL (#PCDATA) >
   <!ATTLIST NULL
   value        CDATA          #REQUIRED >

<!-- LINK already defined -->

<!-- Link to relevant information.
     The link may contain $ symbols indicating variable substitution,
     (see "Remarks about the ID attribute" above)
     The kind of information which can be returned is defined by the
     content-role attribute; proposed keys are:
        query   URL template to query the resource
        hints   URL template to get additional information proposed
                by the server for use by the application
        doc     URL template to get documentation about the resource
        ....    (other tbd's) -->

<!-- Data of the table. The type attribute determines the
     way that the data are embedded
        type      csv    Character separed values
```

```
                         in an escape section <![CDATA[....]]>
                  xml    fully tagged (Ed. Shaya prop)
                  xlink  link to the data
        recsep    for CSV, record separator (by default \n) (ex: "\r")
        colsep    for CSV, column separator (by default \t) (ex: "|")
        headlines for CSV, number of heading lines to ignore (comment lines)
-->
<!ELEMENT DATA (#PCDATA | mtable | LINK)* >
   <!ATTLIST DATA
   type          (csv|xml|xlink) "csv"
   recsep        CDATA          #IMPLIED
   colsep        CDATA          #IMPLIED
   headlines     CDATA          #IMPLIED >

<!-- from MathLM -->
<!ELEMENT mtable       (mtr)+ >
<!ELEMENT mtr          (mn)+  >
<!ELEMENT mn           (#PCDATA) >

<!-- example of an href link to a catalog:
     href="http://strule.cs.qub.ac.uk/cgi-
bin/catcgi?file=/home/damieng/catcgi/1044/spf2.dat_nl&amp;from=$from&amp;to=$to"
     this will return a file with the catalog from row $from to row $to.
  -->



<!ELEMENT IMAGE   (IDENTIFIER?, IMCODE?, TITLE?, AUTHORS?, (IMPROP |
                   FILEHIST | REFERENC | COMMENTS | OBSERVATIONS |
                   IMBINFILE | IMSPACE | IMSPEC)+,
                   LINKS?, LINKIDENTS?, SPECIFIC?) >

<!-- mostly FITS keywords -->
<!-- see FITS documentation: "http://www.cv.nrao.edu/fits/" -->

<!-- IDENTIFIER already defined -->

<!ELEMENT IMCODE          (#PCDATA) >
<!-- a universal code name for astro. images, e.g. ADIL|98.MM.01.01 -->

<!-- TITLE already defined -->

<!-- AUTHORS already defined -->

<!ELEMENT IMPROP   (BITPIX, NAXIS, (NAXISN+ | (NAXIS1, NAXIS2, NAXIS3?,
                    NAXIS4?)), BUNIT?, BSCALE?, BZERO?, BLANK?,
                    DATAMIN?, DATAMAX?) >
<!ELEMENT BITPIX   (#PCDATA) >
<!ELEMENT NAXIS          (#PCDATA) >
<!ELEMENT NAXISN   (#PCDATA) >
<!ATTLIST NAXISN
            n     CDATA       #REQUIRED >
<!-- NAXIS1 NAXIS2 NAXIS3 NAXIS4 deprecated -->
<!ELEMENT NAXIS1   (#PCDATA) >
<!ELEMENT NAXIS2   (#PCDATA) >
<!ELEMENT NAXIS3   (#PCDATA) >
<!ELEMENT NAXIS4   (#PCDATA) >
<!ELEMENT BUNIT          (#PCDATA) >
<!ELEMENT BSCALE   (#PCDATA) >
<!ELEMENT BZERO          (#PCDATA) >
<!ELEMENT BLANK          (#PCDATA) >
```

```
<!ELEMENT DATAMIN  (#PCDATA) >
<!ELEMENT DATAMAX  (#PCDATA) >

<!ELEMENT FILEHIST (DATE | ORIGIN | BLOCKED)+ >
<!-- DATE already defined. Searchable -->
<!ELEMENT ORIGIN   (#PCDATA) >
<!-- Searchable -->
<!ELEMENT BLOCKED  (#PCDATA) >

<!ELEMENT REFERENC       (#PCDATA) >
<!-- Searchable -->

<!ELEMENT COMMENTS (COMMENT | HISTORY)+ >
<!ELEMENT COMMENT  (#PCDATA) >
<!-- Searchable -->
<!ELEMENT HISTORY  (#PCDATA) >
<!-- Searchable -->

<!ELEMENT OBSERVATIONS (DATE-OBS | TELESCOP | INSTRUME | SITELAT | SITELONG |
                        OBSERVER | OBJECT |
                        EQUINOX | EPOCH |
                        TIME | CAMNUM | CCD-ID | EPERADU | FOCALLEN |
                        CCDANGLE | PIXELDIM | BANDPASS | FILTER | COORD |
                        AIRMASS | EXPTIME | RNOISE | TEMP)+ >
<!ELEMENT DATE-OBS       (#PCDATA) >
<!-- Searchable -->
<!ELEMENT TELESCOP       (#PCDATA) >
<!-- Searchable -->
<!ELEMENT INSTRUME       (#PCDATA) >
<!-- Searchable -->
<!ELEMENT SITELAT  (#PCDATA) >
<!ELEMENT SITELONG       (#PCDATA) >
<!ELEMENT OBSERVER       (#PCDATA) >
<!-- Searchable -->
<!-- this field should be transformed into a personname, but it's going
to be difficult to transform FITS files... -->
<!ELEMENT OBJECT   (#PCDATA) >
<!-- Searchable -->
<!ELEMENT EQUINOX  (#PCDATA) >
<!ELEMENT EPOCH          (#PCDATA) >
      <!-- not in FITS standard (maybe from ESO) -->
<!ELEMENT TIME           (#PCDATA) >
<!ELEMENT CAMNUM   (#PCDATA) >
<!ELEMENT CCD-ID   (#PCDATA) >
<!ELEMENT EPERADU  (#PCDATA) >
<!ELEMENT FOCALLEN       (#PCDATA) >
<!ELEMENT CCDANGLE       (#PCDATA) >
<!ELEMENT PIXELDIM       (#PCDATA) >
<!ELEMENT BANDPASS       (#PCDATA) >
<!ELEMENT FILTER   (#PCDATA) >
<!-- COORD defined above -->
<!ELEMENT AIRMASS  (#PCDATA) >
<!ELEMENT EXPTIME  (#PCDATA) >
<!ELEMENT RNOISE   (#PCDATA) >
<!ELEMENT TEMP           (#PCDATA) >

<!ELEMENT IMBINFILE     (#PCDATA) >
<!-- this should be replaced by an XML Link -->
<!ATTLIST IMBINFILE
            filename    CDATA       #REQUIRED >
```

```
<!ELEMENT IMSPACE (CTYPEN, CRPIXN, CROTAN, CRVALN, CDELTN) >

<!ELEMENT CTYPEN   (#PCDATA) >
<!ATTLIST CTYPEN
            n     CDATA       #REQUIRED >
<!ELEMENT CRPIXN   (#PCDATA) >
<!ATTLIST CRPIXN
            n     CDATA       #REQUIRED >
<!ELEMENT CROTAN   (#PCDATA) >
<!ATTLIST CROTAN
            n     CDATA       #REQUIRED >
<!ELEMENT CRVALN   (#PCDATA) >
<!ATTLIST CRVALN
            n     CDATA       #REQUIRED >
<!ELEMENT CDELTN   (#PCDATA) >
<!ATTLIST CDELTN
            n     CDATA       #REQUIRED >

<!ELEMENT IMSPEC (RESTFREQ, FREQMIN, FREQ, FREQMAX, SPECIES?, TRANSIT?) >

<!ELEMENT RESTFREQ   (#PCDATA) >
<!-- rest frequency of primary spectral line being imaged -->

<!ELEMENT FREQMIN    (#PCDATA) >
<!-- minimum frequency covered by image -->

<!ELEMENT FREQ       (#PCDATA) >
<!-- identifying frequency of image (usually image center) -->

<!ELEMENT FREQMAX    (#PCDATA) >
<!-- maximum frequency covered by image -->

<!ELEMENT SPECIES    (#PCDATA) >
<!-- chemical formule for spectral line species being imaged -->

<!ELEMENT TRANSIT    (#PCDATA) >
<!-- the transition of the spectral line -->


<!-- LINKS already defined -->
<!-- content-roles for IMAGE:
     "has_for_author", "is_included_in", "has_for_ref", "other_data",
     "is_an_image_of" -->
<!-- LINKIDENTS already defined -->
<!-- SPECIFIC already defined -->



<!-- person working in astronomy (usually an author) -->
<!ELEMENT PERSON   (PERSONNAME,
                    AFFILIATION?, JOB-TITLE?,
                    ADDRESS?, TEL?, FAX?, EMAIL?, HOMEPAGE?, KEYWORDS?,
                    LINKS?, LINKIDENTS?, SPECIFIC? ) >
<!-- personname already defined. Searchable -->
<!ELEMENT AFFILIATION    (#PCDATA) >
<!ELEMENT JOB-TITLE      (#PCDATA) >
<!ELEMENT ADDRESS        (#PCDATA) >
<!ELEMENT TEL            (#PCDATA) >
<!ELEMENT FAX            (#PCDATA) >
<!ELEMENT EMAIL          (#PCDATA) >
<!ELEMENT HOMEPAGE       (#PCDATA) >
```

```
<!-- homepage URL -->
<!-- KEYWORDS already defined. Research Interests -->
<!-- LINKS already defined -->
<!-- LINKIDENTS already defined -->
<!-- content-roles for PERSON:
     "is_author_of", "is_studying", "other_data" -->
<!-- SPECIFIC already defined -->



<!ELEMENT PROJECT  (IDENTIFIER, PROJCODE, AUTHORS, TITLE, DEPDATE, COPYRIGHT?,
                    COMMENT?, KEYWORDS?, LINKS?, SPECIFIC?) >

<!-- IDENTIFIER already defined -->

<!ELEMENT PROJCODE          (#PCDATA) >
<!-- a universal code name for astro. projects, e.g. ADIL|98.MM.01 -->

<!-- AUTHORS already defined -->

<!-- TITLE already defined -->

<!ELEMENT DEPDATE       (#PCDATA) >
<!-- date/time of deposit of project; format: ISO 8601 -->

<!-- COPYRIGHT already defined -->

<!-- COMMENT already defined -->

<!-- KEYWORDS already defined -->

<!-- LINKS already defined -->

<!-- in the links: references, contact, related projects, ... -->
<!-- examples:
 <LINK xml:link="simple"
href="urn:aml.org:format=AML:amlobj=ARTICLE:ident=1997Ap.....40..285M:server=ADS
bib" content-role="has_for_ref" title="Relationship of Markarian galaxies with
their surroundings."/>
 <LINK xml:link="simple"
href="urn:aml.org:format=AML:amlobj=PERSON:ident=Meixner%2C%20Margaret" content-
role="has_for_contact" title="Margaret Meixner"/>

-->
<!-- content-roles: has_for_ref, has_for_contact, related -->

<!-- SPECIFIC already defined -->
```

Appendix IV: Modified noising algorithm

Step 1. Draw an initial solution. Record it as the *current_solution*. **Goto** step 2.

Step 2. Set *rate*, the noising rate, to *max_rate*.
Calculate *delta_rate*, the rate at which *rate* is decremented, using *max_rate*, *min_rate*, and *max_iter*.
Set the stop counter, *counter* to 0.
Goto step 3.

Step 3. Apply *descent* to the *current_solution* using the original data set *D*.
Record it as the *best_solution* and the cost function value of this solution as the *best_cost*.
Goto step 4.

Step 4. **Do** steps 5 and 6 **While** (*counter* is less than *stop_count* **and** *rate* is greater than or equal to *min_rate*)
Exit the algorithm return the *best_solution*.

Step 5. Noise the original data set using noising rate, *rate* to get the noised data set, *D'*.
If *counter* **mod** 2 = 0 **Then**
Descent on *best_solution* using *D'*.
Else *Descent* on *current_solution* using *D'*.
Set the above solution to *current_solution*.

Step 6. *Descent* on the *current_solution* using *D*.
If the cost of the *current_solution* < *best_cost* **Then**
Record it as the *best_cost* and the *current_solution* as the *best_solution*.
Set *counter* to 0.
Else Increment *counter*.
Decrease *rate* by a factor of *delta_rate*.

Descent(*s*):
s is the initial solution.
Repeat until all transformations have been tested without improvement:
Create a neighbor *s'* of *s* by using an elementary transformation
(it can be only moves, or moves and exchanges alternatively)
If the cost of *s'* < cost of *s* **Then** set *s* to *s'*

Appendix V: IM2PRE.XSL

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/XSL/Transform/1.0">
<xsl:output method="xml"/>

<xsl:template name="print-authors">
<xsl:param name="authors" select="IMAGE/AUTHORS"/>
<xsl:for-each select="$authors/PERSONNAME">
<xsl:value-of select="LASTNAME"/>
<xsl:if test="FIRSTNAMEA!=''">, <xsl:value-of select="FIRSTNAMEA"/></xsl:if>
<xsl:if test="FIRSTNAMEB!=''"> <xsl:value-of select="FIRSTNAMEB"/></xsl:if>
<xsl:if test="FIRSTNAMEC!=''"> <xsl:value-of select="FIRSTNAMEC"/></xsl:if>
<xsl:if test="not(position()=last())">; </xsl:if>
</xsl:for-each>
</xsl:template>

<xsl:template name="print-ra">
<xsl:param name="radeg" select="0"/>
<xsl:variable name="ra"><xsl:value-of select="number($radeg) div
15"/></xsl:variable>
<xsl:variable name="h"><xsl:value-of select="floor($ra)"/></xsl:variable>
<xsl:variable name="m"><xsl:value-of select="($ra - $h) * 60"/></xsl:variable>
<xsl:variable name="fm"><xsl:value-of select="floor($m)"/></xsl:variable>
<xsl:variable name="s"><xsl:value-of select="($m - $fm) * 60"/></xsl:variable>
<xsl:value-of select="format-number($h,'00')"/>:<xsl:value-of select="format-
number($fm,'00')"/>:<xsl:value-of select="format-number($s,'00.00')"/>
</xsl:template>

<xsl:template name="print-dec">
<xsl:param name="mdec" select="0"/>
<!-- there is no abs function. sigh. -->
<xsl:variable name="sgn">
<xsl:if test="$mdec &lt; 0">-</xsl:if>
<xsl:if test="$mdec &gt;= 0">+</xsl:if>
</xsl:variable>
<xsl:variable name="dec">
<xsl:if test="$mdec &lt; 0"><xsl:value-of select="-$mdec"/></xsl:if>
<xsl:if test="$mdec &gt;= 0"><xsl:value-of select="$mdec"/></xsl:if>
</xsl:variable>
<xsl:variable name="fd"><xsl:value-of select="floor($dec)"/></xsl:variable>
<xsl:variable name="m"><xsl:value-of select="($dec - $fd) * 60"/></xsl:variable>
<xsl:variable name="fm"><xsl:value-of select="floor($m)"/></xsl:variable>
<xsl:variable name="s"><xsl:value-of select="($m - $fm) * 60"/></xsl:variable>
<xsl:value-of select="$sgn"/>
<xsl:value-of select="format-number($fd,'00')"/>:<xsl:value-of select="format-
number($fm,'00')"/>:<xsl:value-of select="format-number($s,'00.00')"/>
</xsl:template>

<xsl:template match="/AML">
<IMGPRETMPL>
<METADATA>
<CREATORS>
<xsl:for-each select="METADATA/CREATOR">
<CREATOR><xsl:value-of select="."/></CREATOR>
</xsl:for-each>
</CREATORS>
<IDENTIFIER><xsl:value-of select="METADATA/IDENTIFIER"/></IDENTIFIER>
<DATE><xsl:value-of select="METADATA/DATE"/></DATE>
</METADATA>
```

```
<xsl:variable name="imident"><xsl:value-of select="substring-
after(IMAGE/IMCODE,'|')"/></xsl:variable>
<xsl:variable name="projident"><xsl:value-of
select="substring($imident,1,8)"/></xsl:variable>

<xsl:variable name="docname"><xsl:value-of
select="concat('http://imagelib.ncsa.uiuc.edu/project/amldoc/',$projident)"/></x
sl:variable>

<PROJ>
<xsl:copy-of select="document($docname)/AML/PROJECT/TITLE"/>
<xsl:copy-of select="document($docname)/AML/PROJECT/SPECIFIC/ADILPROJ/UAUTHOR"/>
</PROJ>

<xsl:copy-of select="IMAGE/IMCODE"/>
<xsl:copy-of select="IMAGE/SPECIFIC/ADILIM/UNAME"/>
<xsl:copy-of select="IMAGE/SPECIFIC/ADILIM/OBJNAMES"/>
<xsl:copy-of select="IMAGE/SPECIFIC/ADILIM/OBJTYPES"/>
<xsl:copy-of select="IMAGE/IMSPEC/TRANSIT"/>
<xsl:copy-of select="IMAGE/SPECIFIC/ADILIM/PREVIEW/PRVANNOT"/>
<xsl:copy-of select="IMAGE/SPECIFIC/ADILIM/FILESIZE"/>
<xsl:copy-of select="IMAGE/REFERENC"/>
<OBSERVATIONS>
<xsl:copy-of select="IMAGE/OBSERVATIONS/OBJECT"/>
<xsl:copy-of select="IMAGE/OBSERVATIONS/TELESCOP"/>
<xsl:copy-of select="IMAGE/OBSERVATIONS/EQUINOX"/>
<xsl:copy-of select="IMAGE/OBSERVATIONS/EPOCH"/>
<xsl:copy-of select="IMAGE/IMSPEC/RESTFREQ"/>
</OBSERVATIONS>
<xsl:copy-of select="IMAGE/IMPROP/BUNIT"/>

<IMSPACETABLE>
<FIELDS>
<FIELD><TITLE>Type</TITLE></FIELD>
<FIELD><TITLE>Pixels</TITLE></FIELD>
<FIELD><TITLE>Coord Value</TITLE></FIELD>
<FIELD><TITLE>at  Pixel</TITLE></FIELD>
<FIELD><TITLE>Coord Incr</TITLE></FIELD>
<FIELD><TITLE>Units</TITLE></FIELD>
</FIELDS>
<ROWS>
<xsl:for-each select="IMAGE/IMSPACE/CTYPEN">
<tr>
<xsl:variable name="n"><xsl:value-of select="position()"/></xsl:variable>
<td><xsl:apply-templates/></td>
<td><xsl:value-of select="../../IMPROP/NAXISN[@n=$n]"/></td>
<td>
<xsl:choose>
<xsl:when test="starts-with(.,'RA')">
<xsl:call-template name="print-ra">
<xsl:with-param name="radeg" select="../CRVALN[@n=$n]"/>
</xsl:call-template>
</xsl:when>
<xsl:when test="starts-with(.,'DEC') or starts-with(.,'GLAT') or starts-
with(.,'GLON') or starts-with(.,'ELAT') or starts-with(.,'ELON')">
<xsl:call-template name="print-dec">
<xsl:with-param name="mdec" select="../CRVALN[@n=$n]"/>
</xsl:call-template>
</xsl:when>
<xsl:when test="starts-with(.,'VELO') or starts-with(.,'FELO')">
```

```
<xsl:value-of select="format-number(number(../CRVALN[@n=$n]) div
1000.0,'######.####')"/>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="../CRVALN[@n=$n]"/>
</xsl:otherwise>
</xsl:choose>
</td>
<td><xsl:value-of select="format-
number(number(../CRPIXN[@n=$n]),'#####.##')"/></td>
<xsl:choose>
<xsl:when test="starts-with(.,'RA') or starts-with(.,'DEC') or starts-
with(.,'GLAT') or starts-with(.,'GLON') or starts-with(.,'ELAT') or starts-
with(.,'ELON')">
<td><xsl:value-of select="format-
number(number(../CDELTN[@n=$n])*3600,'#####.####')"/></td><td>arcsec</td>
</xsl:when>
<xsl:when test="starts-with(.,'VELO') or starts-with(.,'FELO')">
<td><xsl:value-of select="format-number(number(../CDELTN[@n=$n]) div
1000.0,'######.####')"/></td><td>km/s</td>
</xsl:when>
<xsl:when test="starts-with(.,'FREQ')">
<td><xsl:value-of select="../CDELTN[@n=$n]"/></td><td>Hz</td>
</xsl:when>
<xsl:otherwise>
<td><xsl:value-of select="../CDELTN[@n=$n]"/></td><td>?</td>
</xsl:otherwise>
</xsl:choose>
</tr>
<xsl:text>
</xsl:text>
</xsl:for-each>
</ROWS>
</IMSPACETABLE>



<KEYWORDS>
<KEYWORD>
<NAME>AUTHOR</NAME>
<VALUE><xsl:choose>
<xsl:when test="IMAGE/AUTHORS!=''">
<xsl:call-template name="print-authors">
<xsl:with-param name="authors" select="IMAGE/AUTHORS"/>
</xsl:call-template>
</xsl:when>
<xsl:otherwise>
<xsl:call-template name="print-authors">
<xsl:with-param name="authors" select="document($docname)/AML/PROJECT/AUTHORS"/>
</xsl:call-template>
</xsl:otherwise>
</xsl:choose></VALUE>
</KEYWORD>
<KEYWORD>
<NAME>TITLE</NAME>
<VALUE><xsl:value-of select="document($docname)/AML/PROJECT/TITLE"/></VALUE>
</KEYWORD>
<KEYWORD>
<NAME>RIGHT_ASCENSION</NAME>
<VALUE>[  <xsl:call-template name="print-ra">
<xsl:with-param name="radeg" select="IMAGE/SPECIFIC/ADILIM/IMCOORD/LONMIN"/>
```

```
</xsl:call-template>,   <xsl:call-template name="print-ra">
<xsl:with-param name="radeg" select="IMAGE/SPECIFIC/ADILIM/IMCOORD/LONMAX"/>
</xsl:call-template> ] (J2000)</VALUE>
</KEYWORD>
<KEYWORD>
<NAME>DECLINATION</NAME>
<VALUE>[ <xsl:call-template name="print-dec">
<xsl:with-param name="mdec" select="IMAGE/SPECIFIC/ADILIM/IMCOORD/LATMIN"/>
</xsl:call-template>,  <xsl:call-template name="print-dec">
<xsl:with-param name="mdec" select="IMAGE/SPECIFIC/ADILIM/IMCOORD/LATMAX"/>
</xsl:call-template> ] (J2000)</VALUE>
</KEYWORD>
<KEYWORD>
<NAME>RESTFREQ</NAME>
<VALUE><xsl:value-of select="IMAGE/IMSPEC/RESTFREQ"/></VALUE>
</KEYWORD>
<KEYWORD>
<NAME>FREQUENCY</NAME>
<VALUE>[ <xsl:value-of select="IMAGE/IMSPEC/FREQMIN"/>, <xsl:value-of
select="IMAGE/IMSPEC/FREQMAX"/> ] (Hz)</VALUE>
</KEYWORD>
<KEYWORD>
<NAME>OBJECT_NAME</NAME>
<VALUE><xsl:for-each select="IMAGE/SPECIFIC/ADILIM/OBJNAMES/OBJNAME">
<xsl:apply-templates/>
<xsl:if test="not(position()=last())">, </xsl:if>
</xsl:for-each></VALUE>
</KEYWORD>
<KEYWORD>
<NAME>OBJECT_TYPE</NAME>
<VALUE><xsl:for-each select="IMAGE/SPECIFIC/ADILIM/OBJTYPES/OBJTYPE">
<xsl:apply-templates/>
<xsl:if test="not(position()=last())">, </xsl:if>
</xsl:for-each></VALUE>
</KEYWORD>
<KEYWORD>
<NAME>TELESCOPE</NAME>
<VALUE><xsl:value-of select="IMAGE/OBSERVATIONS/TELESCOP"/></VALUE>
</KEYWORD>
<KEYWORD>
<NAME>SPECIES</NAME>
<VALUE><xsl:value-of select="IMAGE/IMSPEC/SPECIES"/></VALUE>
</KEYWORD>
<KEYWORD>
<NAME>TRANSITION</NAME>
<VALUE><xsl:value-of select="IMAGE/IMSPEC/TRANSIT"/></VALUE>
</KEYWORD>
<KEYWORD>
<NAME>IS_SURVEY</NAME>
<VALUE><xsl:choose>
<xsl:when test="document($docname)/AML/PROJECT/SPECIFIC/ADILPROJ/ISSURVEY!=''">
<xsl:value-of
select="document($docname)/AML/PROJECT/SPECIFIC/ADILPROJ/ISSURVEY"/>
</xsl:when>
<xsl:otherwise>F</xsl:otherwise>
</xsl:choose></VALUE>
</KEYWORD>
<KEYWORD>
<NAME>REFERENCE</NAME>
<VALUE><xsl:value-of select="IMAGE/REFERENC"/></VALUE>
</KEYWORD>
```

```
<KEYWORD>
<NAME>CODENAME</NAME>
<VALUE><xsl:value-of select="$imident"/></VALUE>
</KEYWORD>
</KEYWORDS>

<xsl:if test="count(PROJECT/LINKS/LINK[not(starts-with(@href,'urn:'))])>0">
<LINKS>
<xsl:for-each select="PROJECT/LINKS/LINK[not(starts-with(@href,'urn:'))]">
<LINK>
<TITLE><xsl:value-of select="@title"/></TITLE>
<HREF><xsl:value-of select="@href"/></HREF>
</LINK>
</xsl:for-each>
</LINKS>
</xsl:if>

</IMGPRETMPL>
</xsl:template>
</xsl:stylesheet>
```

Appendix VI: IMGPRETMPL.DTD

```
<!ELEMENT IMGPRETMPL (METADATA, PROJ, IMCODE, UNAME, OBJNAMES?, OBJTYPES?,
TRANSIT, PRVANNOT, FILESIZE, REFERENC, OBSERVATIONS, BUNIT, IMSPACETABLE,
LINKS?>

<!ELEMENT METADATA (CREATORS?, IDENTIFIER?, DATE?)>

<!ELEMENT CREATORS  (CREATOR+)>

<!ELEMENT CREATOR          (#PCDATA) >

<!ELEMENT IDENTIFIER      (#PCDATA) >

<!ELEMENT DATE            (#PCDATA) >


<!ELEMENT PROJ (TITLE, UAUTHOR)>

<!ELEMENT TITLE (#PCDATA) >

<!ELEMENT UAUTHOR (#PCDATA) >


<!ELEMENT IMCODE            (#PCDATA) >

<!ELEMENT UNAME            (#PCDATA) >

<!ELEMENT OBJNAMES (OBJNAME+) >
<!ELEMENT OBJNAME (#PCDATA) >

<!ELEMENT OBJTYPES (OBJTYPE+) >
<!ELEMENT OBJTYPE (#PCDATA) >

<!ELEMENT TRANSIT            (#PCDATA) >

<!ELEMENT PRVANNOT (#PCDATA) >

<!ELEMENT FILESIZE (#PCDATA) >

<!ELEMENT REFERENC (#PCDATA) >


<!ELEMENT OBSERVATIONS (OBJECT, TELESCOP, EQUINOX, EPOCH, RESTFREQ) >

<!ELEMENT OBJECT (#PCDATA) >

<!ELEMENT TELESCOP (#PCDATA) >

<!ELEMENT EQUINOX (#PCDATA) >

<!ELEMENT EPOCH (#PCDATA) >

<!ELEMENT RESTFREQ (#PCDATA) >


<!ELEMENT BUNIT           (#PCDATA) >


<!ELEMENT IMSPACETABLE (FIELDS, ROWS)>
```

```
<!-- generated using NAXISN, CTYPEN, CRPIXN, CRVALN, CDELTN (not CROTAN) -->

<!ELEMENT FIELDS (FIELD+)>
<!ELEMENT FIELD (TITLE, DESCRIPTION?)>

<!ELEMENT TITLE (#PCDATA) >

<!ELEMENT ROWS (tr+)>

<!ELEMENT tr (td+)>

<!ELEMENT td (#PCDATA)>


<!ELEMENT KEYWORDS (KEYWORD+)>
<!-- list of searchable keywords -->
<!-- generated using: PROJECT/AUTHORS, PROJECT/TITLE, LONMIN, LONMAX, LATMIN,
LATMAX, RESTFREQ, FREQMIN, FREQMAX, OBJNAMES, OBJTYPES, TELESCOP, SPECIES,
TRANSIT, PROJECT/ISSURVEY, REFERENC, IMCODE -->
<!-- coordinates here are formatted like HH:MM:SS.dd or +DD:MM:SS.dd -->

<!ELEMENT KEYWORD (NAME, VALUE)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT VALUE (#PCDATA)>


<!ELEMENT LINKS (LINK+)>

<!ELEMENT LINK (TITLE, TYPE?, HREF, FILESIZE?) >

<!ELEMENT TITLE (#PCDATA) >

<!ELEMENT TYPE (#PCDATA) >

<!ELEMENT HREF (#PCDATA) >

<!-- FILESIZE is already defined -->
```

Appendix VII: IMTMPL.XSL

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/XSL/Transform/1.0">
<!--                xmlns="http://www.w3.org/TR/xhtml1"> -->
<xsl:output method="html"/>

<xsl:template match="/IMGPRETMPL">
<html>
<head>
<xsl:for-each select="METADATA/CREATORS/CREATOR">
<meta name="Author" content="{.}"/>
</xsl:for-each>
<meta name="DC.identifier" content="{METADATA/IDENTIFIER}"/>
<meta name="DC.date" content="{METADATA/DATE}"/>
<title><xsl:value-of select="IMCODE"/>Project Page </title>
<link rev="made" href="mailto:dguillau@astro.uiuc.edu"/>
<base href="http://imagelib.ncsa.uiuc.edu/project/document/{IMCODE}"/>
</head>

<body bgcolor="#ffffff" link="#006666">

<xsl:variable name="imident"><xsl:value-of select="substring-
after(IMCODE,'|')"/></xsl:variable>
<xsl:variable name="projident"><xsl:value-of
select="substring($imident,1,8)"/></xsl:variable>

<font face="arial,helvetica" size="-1">
<table width="100%" border="0" cellspacing="0" cellpadding="2">
<tr><td valign="bottom">
<a href="http://www.ncsa.uiuc.edu/"><img border="0"
   src="http://imagelib.ncsa.uiuc.edu/imagelib/gifs/ncsafooter.gif"
   width="82" height="24" alt="NCSA "/></a><a
   href="http://imagelib.ncsa.uiuc.edu/imagelib"><img border="0"
   src="http://imagelib.ncsa.uiuc.edu/imagelib/gifs/adilfooter.gif"
   width="85" height="24" alt="Astronomy Digital Image Library"/></a>
</td>
 <td width="60"> </td>
 <td align="left" width="90">
<a href="http://imagelib.ncsa.uiuc.edu/project/description/{$projident}"
   target="ADIL_info"><img
   src="http://imagelib.ncsa.uiuc.edu/imagelib/gifs/cgridptr.gif"
   width="19" height="19" border="0" align="top"/>Abstract</a>
</td>
 <td align="left" width="90">
<a href="http://imagelib.ncsa.uiuc.edu/document/{$projident}"><img
   src="http://imagelib.ncsa.uiuc.edu/imagelib/gifs/cgridptr.gif"
   width="19" height="19" border="0" align="top"/>Project Page</a>
</td>
 <td align="left" width="90">
<a href="http://imagelib.ncsa.uiuc.edu/project/download/{$imident}"><img
   src="http://imagelib.ncsa.uiuc.edu/imagelib/gifs/cgridptr.gif"
   width="19" height="19" border="0" align="top"/>Download</a>
</td>
 <td align="left" width="90">
<a href="http://imagelib.ncsa.uiuc.edu/imagelib/QueryPage.html"><img
   src="http://imagelib.ncsa.uiuc.edu/imagelib/gifs/cgridptr.gif"
   width="19" height="19" border="0" align="top"/>Data Search</a>
</td>
<!-- <td align="left" width="100">
<a href="http://imagelib.ncsa.uiuc.edu/imagelib/GetByCode.html"><img
```

```
   src="http://imagelib.ncsa.uiuc.edu/imagelib/gifs/cgridptr.gif"
   width="19" height="19" border="0" align="top"/>Get By Code</a></td> -->
 <td align="left" width="50">
<a href="http://imagelib.ncsa.uiuc.edu/imagelib/help/help_previewpgW.html"
   target="ADIL_help"><img
   src="http://imagelib.ncsa.uiuc.edu/imagelib/gifs/cgridptr.gif"
   border="0" height="19" width="18" align="top"/>Help</a>
</td>
</tr>
 <tr bgcolor="#006666"><td colspan="7"><font
     face="arial,helvetica" size="+1" color="#ffffff">
NCSA Astronomy Digital Image Library      Image Preview Page
</font>
</td>
</tr>
</table>
</font>

<h1> <xsl:value-of select="PROJ/TITLE"/> </h1>

<h2> <xsl:value-of select="PROJ/UAUTHOR"/> </h2>

<strong>Codename: </strong> <xsl:value-of select="IMCODE"/>   
(<xsl:value-of select="UNAME"/>)<br/>
<strong>Object Name: </strong>
<xsl:for-each select="OBJNAMES/OBJNAME">
<xsl:apply-templates/>
<xsl:if test="not(position()=last())">, </xsl:if>
</xsl:for-each>
<br/>
<strong>Object Type: </strong>
<xsl:for-each select="OBJTYPES/OBJTYPE">
<xsl:apply-templates/>
<xsl:if test="not(position()=last())">, </xsl:if>
</xsl:for-each>
<br/>
<xsl:if test="TRANSIT!=''">
<strong>Spectral Line: </strong> <xsl:value-of select="TRANSIT"/> <br/>
</xsl:if>
<br/>

<a href="http://imagelib.ncsa.uiuc.edu/project/prvimg/{$imident}">
<img src="http://imagelib.ncsa.uiuc.edu/project/prvimg/{$imident}"
     alt="[selected plane from image]" align="bottom"/></a> <br/>
<xsl:value-of select="PRVANNOT"/>
<br/>
<strong>File Size: </strong> <xsl:value-of select="FILESIZE"/> kB<br/>

Reference:
<xsl:variable name="ref"><xsl:value-of select="REFERENC"/></xsl:variable>
<a href="http://adsabs.harvard.edu/cgi-bin/bib_query?{$ref}">
<xsl:value-of select="$ref"/></a> <br/>
<br/>
<a name="datainfo"/>
Object: <xsl:value-of select="OBSERVATIONS/OBJECT"/><br/>
<xsl:if test="TRANSIT!=''">
Line: <xsl:value-of select="TRANSIT"/><br/>
</xsl:if>
Telescope: <xsl:value-of select="OBSERVATIONS/TELESCOP"/><br/>
Image intensity units: <xsl:value-of select="BUNIT"/>
```

```
<xsl:for-each select="IMSPACETABLE">
  <xsl:if test="count(FIELDS/FIELD/DESCRIPTION)>0">
  <dl>
  <xsl:for-each select="FIELDS/FIELD">
    <dt><xsl:value-of select="TITLE"/></dt>
    <dd><xsl:value-of select="DESCRIPTION"/></dd>
  </xsl:for-each>
  </dl>
  <br/>
  </xsl:if>
  <table width="100%" border="1" cellspacing="0" cellpadding="2">
  <tr><xsl:for-each select="FIELDS/FIELD">
    <th><xsl:value-of select="TITLE"/></th>
  </xsl:for-each></tr>
  <xsl:for-each select="ROWS/tr">
    <xsl:copy-of select="."/>
  </xsl:for-each>
  </table>
</xsl:for-each>


<xsl:if test="OBSERVATIONS/EQUINOX!=''">
Equinox:     <xsl:value-of select="OBSERVATIONS/EQUINOX"/><br/>
</xsl:if>
<xsl:if test="OBSERVATIONS/EPOCH!=''">
Epoch:       <xsl:value-of select="OBSERVATIONS/EPOCH"/><br/>
</xsl:if>
Rest frequency:  <xsl:value-of select="OBSERVATIONS/RESTFREQ"/> Hz<br/>
<br/>
<a href="http://imagelib.ncsa.uiuc.edu/document/{$projident}"><img
   src="http://imagelib.ncsa.uiuc.edu/imagelib/gifs/cgridptr.gif"
   width="19" height="19" border="0" align="top"/>
Project Overview</a> <br/>

<a href="http://imagelib.ncsa.uiuc.edu/project/description/{$projident}"
   target="ADIL_info">
<img src="http://imagelib.ncsa.uiuc.edu/imagelib/gifs/cgridptr.gif"
   width="19" height="19" border="0" align="top"/>
Abstract and Project Description</a> <br/>
<a href="http://imagelib.ncsa.uiuc.edu/project/document/{$imident}/img.hdr">
<img src="http://imagelib.ncsa.uiuc.edu/imagelib/gifs/cgridptr.gif"
   width="19" height="19" border="0" align="top"/>
FITS Header Keywords</a><br/>
<a href="http://imagelib.ncsa.uiuc.edu/project/document/{$imident}/img.his">
<img src="http://imagelib.ncsa.uiuc.edu/imagelib/gifs/cgridptr.gif"
   width="19" height="19" border="0" align="top"/>
FITS Header History</a><br/>
<!-- movie page-->
<a href="http://imagelib.ncsa.uiuc.edu/project/moviepg/{$imident}">
<img src="http://imagelib.ncsa.uiuc.edu/imagelib/gifs/cgridptr.gif"
   width="19" height="19" border="0" align="top"/>
Movie Page: Browse other planes from this image</a><br/>
<!-- end movie page -->
<a href="http://imagelib.ncsa.uiuc.edu/project/download/{$imident}">
<img src="http://imagelib.ncsa.uiuc.edu/imagelib/gifs/cgridptr.gif"
   width="19" height="19" border="0" align="top"/>
<strong>Download</strong> the FITS image</a><br/>

<h2> Other Related Data Files </h2>

<xsl:if test="count(LINKS)>0">
```

```
<ul>
<xsl:for-each select="LINKS/LINK">
<li><a href="{@href}"><xsl:value-of select="@title"/></a></li>
</xsl:for-each>
</ul>
</xsl:if>

<h2> Searchable Keyword Values </h2>

<table width="100%" border="1" cellspacing="0" cellpadding="2">
<xsl:for-each select="KEYWORDS/KEYWORD">
<tr><td><xsl:value-of select="NAME"/></td><td><xsl:value-of
select="VALUE"/></td></tr>
</xsl:for-each>
</table>

<br/><hr noshade="1" size="1"/>
<table width="100%" border="0" cellspacing="0" cellpadding="2">
<tr><td valign="top">
<a href="http://www.ncsa.uiuc.edu/"><img border="0"
   src="http://imagelib.ncsa.uiuc.edu/imagelib/gifs/ncsafooter.gif"
   width="82" height="24" alt="NCSA "/></a><a
   href="http://imagelib.ncsa.uiuc.edu/imagelib"><img border="0"
   src="http://imagelib.ncsa.uiuc.edu/imagelib/gifs/adilfooter.gif"
   width="85" height="24" alt="Astronomy Digital Image Library"/></a>
</td>
 <td width="60"> </td>
 <td align="left" width="100"><font face="arial,helvetica" size="-1">
<a href="http://imagelib.ncsa.uiuc.edu/imagelib/QueryPage.html"><img
   src="http://imagelib.ncsa.uiuc.edu/imagelib/gifs/cgridptr.gif"
   width="19" height="19" border="0" align="top"/>Data Search</a></font></td>
 <td align="left" width="100"><font face="arial,helvetica" size="-1">
<a href="http://imagelib.ncsa.uiuc.edu/imagelib/GetByCode.html"><img
   src="http://imagelib.ncsa.uiuc.edu/imagelib/gifs/cgridptr.gif"
   width="19" height="19" border="0" align="top"/>Get By Code</a></font></td>
 <td align="left" width="50"><font face="arial,helvetica" size="-1">
<a href="http://imagelib.ncsa.uiuc.edu/imagelib/help/help_projectpgW.html"
   target="ADIL_help"><img
   src="http://imagelib.ncsa.uiuc.edu/imagelib/gifs/cgridptr.gif"
   width="19" height="19" border="0" align="top"/>Help</a></font>
</td>
</tr>
</table>

<font face="arial,helvetica" size="-2">
The <a href="http://imagelib.ncsa.uiuc.edu/imagelib">Astronomy
Digital Image Library</a> is a project of
<a href="http://monet.astro.uiuc.edu">Radio Astronomy Imaging
Team</a><br/> at the <a href="http://www.ncsa.uiuc.edu/">National Center
for Supercomputing Applications</a> on the campus of the
<a href="http://www.uiuc.edu/">University of Illinois at
Urbana-Champaign</a><br/>
  <br/>
Contact the ADIL:
<a href="mailto:imagelib@ncsa.uiuc.edu">imagelib@ncsa.uiuc.edu</a><br/>

</font>
<br/>

</body>
</html>
```

```
</xsl:template>
</xsl:stylesheet>
```