Classifying Internet Objects

F. Luís Neves * and José N. Oliveira †

INESC Group 2361 and Dep. Informática, Universidade do Minho
4700 Braga, Portugal

June 12, 1995

Abstract

Navigation across the Internet may be an arduous task because of the absence of a classification scheme for the enormous amount of information available through billions of UNIFORM RESOURCE LOCATORS (URLS). The *bookmark* and *history* mechanisms, which are available in Internet browsers such as NETSCAPE and MOSAIC, try to help the users throughout the Internet world.

This paper presents a new approach for this problem based on the reuse methodology developed in the SOUR project. Internet links are seen as reusable objects, stored and maintained in a generalize/specialize structure based on a comparison-metrics algorithm. On the implementation side, SOUR is extended by making use of the NETSCAPE's OLE AUTOMATION and DDE interprocess communication mechanisms which allows a third party application to remotely control the NETSCAPE Navigator client.

1 Introduction

Navigation across the Internet consists of jumping across a set of links interactively chosen by the user during a session with an Internet browser. This may be arduous because of the absence of an effective classification scheme for the enormous amount of information available through billions of inter-linked UNIFORM RESOURCE LOCATORS (URLS). Altogether, this huge world-wide "information system" has the structure of an untyped *semantic network* [5]. The basic idea put forward in this paper is to use the SOUR software system [1] as an Internet Navigation Assistant. SOUR is a system for comparing, classifying and retrieving information about large software systems. Figure 1 depicts the overall structure of the system [19, 17, 18, 14, 15, 16].

The unit of information in SOUR is the so-called *Abstract Object* (AO), a notion which combines the *enumerative* and *faceted* classification schemes [12, 11, 10] as an extension of the popular attributive view of objects in the context of a hierarchical semantic network information model.

A crucial decision to make is how to map Internet nodes onto the SOUR information model. A URL refers to the format used by World Wide Web (WWW) [21] documents to locate files on other servers. A URL gives the type of resource being accessed (e.g., gopher, WAIS), the address of the server and the path of the file. The format is [2]

scheme://host.domain[:port]/path/filename

where *scheme* is one of

*EMAIL: `fln@di.uminho.pt`; URL: `http://s700.uminho.pt/~fln`

†EMAIL: `jno@di.uminho.pt`; URL: `http://www.di.uminho.pt/~jno`

[1] SOUR is the acronym of EUREKA 379 project "SOftware Use and Reuse".

[2] Extracted from [6].

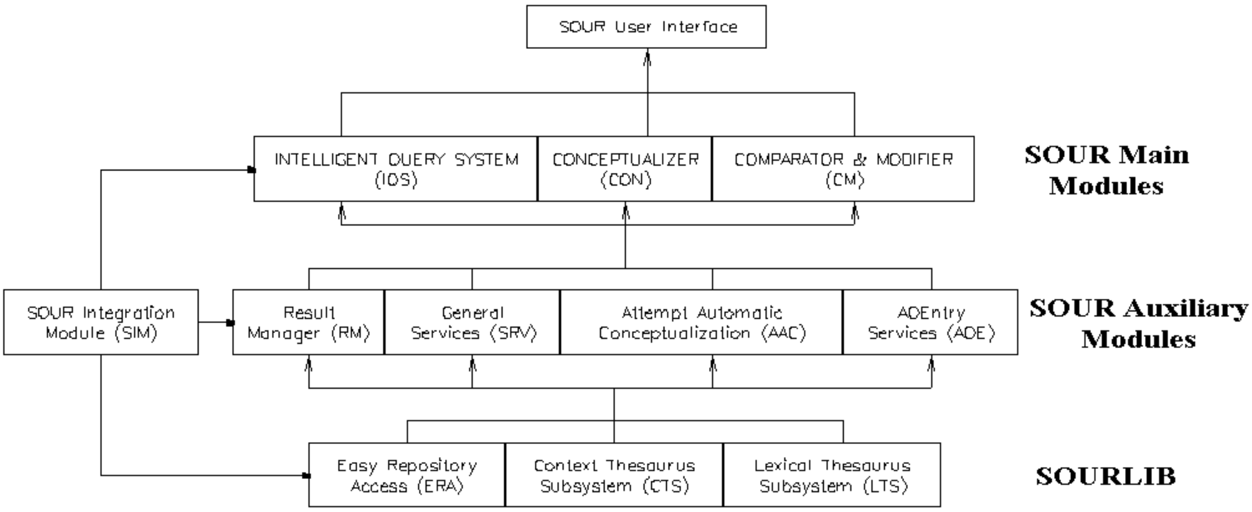

Figure 1: SOUR Overall Architecture

file	file on your local system, or a file on an anonymous ftp server
http	a resource on a World Wide Web server
gopher	a resource on a Gopher server
WAIS	a resource on a WAIS server
news	an Usenet newsgroup
telnet	a connection to a telnet-based service

The above information scheme can be turned into a SOUR class scheme in a way that will be described in this paper. But a summary of the overall SOUR information model will be presented beforehand.

2 Introducing AOs

Information in the SOUR software system [19] is generically recorded in the form of so-called *Abstract Objects* (AOs) which are independent of their physical support — e.g. text file, POSTSCRIPT file — or location — e.g. pathname.

Abstract objects (AOs) are catalogued in the system's abstract archive according to an adopted *standard* of classification called *conceptualization*, which is factored in two layers:

- *Coarse level* — hierarchical "is-a"-like *enumerative* classification, enabling conventional inherited attribute-based reasoning;
- *Fine level* — a *multifaceted* classification schema based on fuzzy concept-network reasoning which extends conventional keyword-oriented classification.

This conceptualization approach is thus a combination of the *enumerative* and *faceted* classification schemes [11, 12], whereby a physical object (e.g. a piece of C-code) "becomes" an AO by attaching to it a *profile* consisting of the following basic items:

- *Classname* — e.g. `code`: this item plugs the object into the system's standard "is-a" hierarchy;
- *Attributes* — e.g. `author`: these provide values for features or properties relevant about the objects, under a simple inheritance mechanism prescribed by the "is-a" hierarchy;

- *Links* — e.g. `implements`: these are AO-valued attributes which establish relevant relationships among AOs;
- *Facets* — e.g. `system=text-formatter`: these are fuzzy tuples of terms which play the rôle of "vague keywords" indexing AOs;
- *Actions* — e.g. `compile`: these are "method-like" AO-attributes recording pre-defined procedures upon which AOs are automatically submitted at standard instants of their life-cycle or triggered by UI-events (e.g. mouse double-clicking);
- *Members* — (if an AO container): collection of references to the sub-objects which the object contains, if any;
- *Relations* — (if an AO cluster): graph structure of links inter-relating to the sub-objects which the main object involves, if any.

Every AO has a unique identity represented by its AOID (*Abstract Object IDentifier*). AOIDs are managed by the system and are transparent to the end-user.

2.1 About the Conceptualization Standard

It should be noted that two different physical objects may happen to be attached to the same conceptual profile. This leads to a notion of "conceptualization equivalence" among objects which has to be managed by the system.

Classnames form a strict hierarchy representing the *enumerative* side of SOUR's classification scheme. From the user's perspective, the supremum of this hierarchy is AOG (*Abstract Object Generic attributes*), a class consisting mainly of system controlled attributes such as:

- **name** — AO name as defined by the user who created the object;
- **pathname** — plugging so-called PAOs ("Physical AOs") into the underlying *file system*;
- **insertedBy** — *username* of user who created the object;
- **lastModifyDate** — date of last updating;
- **lastLookDate** — date of last browsing;
- **nModify** — number of updatings so far;
- *etc.*

Facets are 6-tuples of terms, each term instantiating one of the following, predefined *facet types* [3]:

$$\left.\begin{array}{ll}1. & \textit{function}\text{ or }\textit{action}\\ 2. & \textit{object}\\ 3. & \textit{medium}\text{ or }\textit{data structure}\end{array}\right\}\textit{Functionality}$$
$$\left.\begin{array}{ll}4. & \textit{system type}\\ 5. & \textit{functional area}\\ 6. & \textit{settings}\end{array}\right\}\textit{Environment}$$

Every term instantiating a facet must be present in a subsystem of SOUR called LTS (Lexicon/Thesaurus Subsystem). Terms are related to each other by concepts in a fuzzy way supported by CTS (Context Thesaurus Subsystem), another component of the SOUR architecture [15, 16]. Facets may be regarded as "fuzzy" attributes. Reference [10] provides a formal discussion about the power of fuzzy classification in practice.

[3] Following [11, 12], where they are shown to be quite effective for software classification.

3 Mapping URLs to AOs

The URL addressing format allows a user to specify any object in the Internet, along with sufficient information to retrieve it. The WWW server is responsible for mapping a supplied URL into an object or responding with an error message [4]. This procedure leads to the notion that every Internet transaction is divided in two distinct phases:

- *Identification* — the server validates the URL specification;
- *Retrieval* — the server delivers the corresponding data to the client.

If these two phases are successfully executed it will be possible to access both to the (now valid) URL and the data. The access to the data will have a particular importance in the present study each time the URL identifies an HTML text file. In such case, from the analysis of its contents results that some parts of the text may be used for the conceptualization of the URL.

The following sections will explain in detail how these two entities — the URL and the data which identifies — provide the information that will be attached to an AO.

3.1 AO Identification

Assuming the previously (brief) description of an Abstract Object, it is possible to map the URL information scheme described earlier onto a SOUR object by making:

AO Name	scheme://host.domain[:port]/path/filename
AO Address	/path/filename
AO Type	filename extension (if any)
AO Class	scheme + filename extension (if any)

As expected, an URL provides, just by itself, the minimal information needed for a successful conceptualization [17]. However, if the URL is a HTML text file then some extra information may be added according to its contents (in a way that will be explained in the sections below), otherwise no more information will be attached.

3.2 AO Class

A specific class hierarchy must be created in order to accommodate the *host.domain* information which can be used to classify the AO at *coarse* level. The top of this hierarchy is a class named URL which must have (at least) the following attributes:

Domain_0	(e.g.www,gopher,ftp,s700)
Domain_1	(e.g.ncsa,telepac,inescn,di)
Domain_2	(e.g.uiuc,inesc,uminho)
...	(...)
Domain_n	(e.g.com,pt,org,edu)

At the same time, URL's subclasses reflect the possible scheme values and even filename extensions. However, if for a given URL the value *scheme + filename extension* is not the name of an existent (predefined) class then only the *scheme* value is used. The following class hierarchy is a tentative illustration of this idea and specifies some possible subclasses for the HTTP class:

URL			
	FILE		
	HTTP		
		HTTPHTML	(Html documents)
		HTTPTXT	(Text documents)
		HTTPPS	(Postscript documents)
		HTTPDOC	(Word documents)
		HTTPTEX	(TeX documents)
		HTTPGIF	(Gif images)
		HTTPZIP	(Zipped files)
	GOPHER	...	
		...	
	WAIS	...	
		...	
	NEWS	...	
		...	
	TELNET	...	
		...	

Of course, other attributes may be added to the classes reflecting the specific information of their objects.

3.3 AO Facets

Faceted classification as proposed in this paper will combine some notions of the software-oriented technology designed for text scheme management such as *full text* indexing and retrieval, *free-text scan*, *document clustering*, *unique word* and *vector-space* [3].

The *full text*'s approach generates, in the first place, a list of strings associated with a document [4]. Then, at retrieval time [5], a string match will be tried between each string in the index and a string in the available thesaurus. This strategy is combined with the *unique-word* and the *vector-space* approaches in order to give more retrieval power to the strings that occur more often in the text.

Document Clustering attempts to mimic the human thought process by grouping together documents with related ideas, concepts and terminology [8]. This notion is managed by SOUR's COMPARATOR & MODIFIER subsystem [14] as described later in this paper (see section 5).

All these notions can be put together to provide a default facet classification which will be tried by the so-called SOUR's *Attempt Automatic Conceptualization* (AAC) mechanism [17]. The AAC is applied to the HTML source text of the URL currently being accessed if, of course, the URL identifies a HTML file. The quality of the available CTS/LTS pair is of crucial importance to obtain good results in faceted classification.

For the relevant information to be extracted from the HTML source text, we choose the words that are included in the following HTML structures [6]:

- *Title* — words between elements <TITLE> and </TITLE>;
- *Headings* — words between elements <Hy > and </Hy > where y is a number between 1 and 6 specifying the level of the heading;

Since we are interested on the classification of documents by their contents, these must be reflected in the lexical terms available in the LTS. If, for example, we have a special interest in documents talking about the WWW, then the LTS shall have terms like *Internet*, *Information*, *Web*, *Hypertext*, *Virtual*, *Browser*, *CERN*,

[4] These strings are usually associated with the most relevant information available in the document. This relevant information becomes particularly easy to identify when we are dealing with languages for expressing context-free document descriptions (also known as *Markup Languages*) such as SGML[20], HTML[6], and TEX[7].

[5] Which is based on the *string-match* principle.

HTML, and so on. This specialization of lexical terms, which can improve both the conceptualization and the query mechanism, is supported by the SOUR's capability of working with several CTS/LTS repositories [6].

CTS provides the capability to cope with features of human reasoning such as *classifying by analogy* and *terminological vagueness* [15, 16, 10]. In particular, lexical terms can be connected by conceptual distances interrelating terms (words) according to its contextual meaning. These distances may be regarded as degrees of membership of arcs in a fuzzy graph.

Figure 2 shows a possible set of conceptual relations among the terms described above.

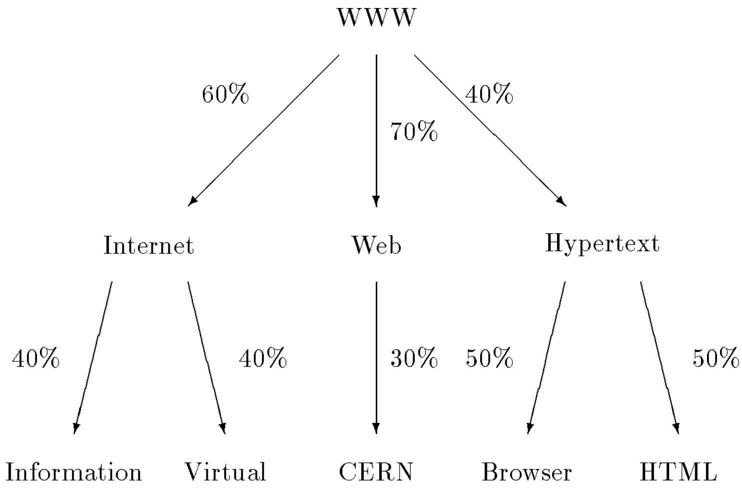

Figure 2: Example of Conceptual Relations

The *fuzzy logic* technique associated with this information structure provides a method to reduce the so-called *precision/recall* trade-off. This is one of the methods that has had some success in decreasing the changes of missing important information [3].

The 6-tuples of predefined terms presented earlier (see section 2.1) were designed by Prieto-Diáz for the specific task of software classification. It is an open problem how these can be extended or adapted to so generic information as accessed through any Internet navigation.

The pre-inserted values for each one of these facets will serve as guidelines for document classification. Possible matches among those values and the words extracted from the HTML text reflect part of the so-called AAC mechanism. The others will be described in the sections below.

3.4 AO Links

We now show how the *semantic network* overall structure of Internet matches with the internal AO-structure of SOUR.

The HTML source text of the URL currently being accessed can also be used to extract AO link information. Each hyperlink to an *external* file [7] will be identified as an *inlink* [8] of the AO that abstracts the current URL. Possible references include:

Hyperlink references — e.g.<A HREF = "URL">...</A>. References of this kind will create links identified by the "Part Of" label;

[6] However, the current version of the prototype allows to work only with one at a time, as indicated at each session's startup.

[7] A reference to a file that is stored in a different server.

[8] An *inlink* of an AO X is a triple of the form $Source \stackrel{LinkName}{\longrightarrow} X$ or just $Source \stackrel{LinkName}{\longrightarrow}$ for short.

Image references — e.g.<IMG SRC = "URL">...</IMG> References of this kind will create links identified by the "Image Of" label;

Embedded references — e.g.<EMBED SRC = "URL"> References of this kind will create links identified by the "Embedded In" label.

Hyperlink, *Image* and *Embedded* references become *inlinks* after the following procedures:

1. the references are mapped onto SOUR AOs following the way described earlier in section 3;
2. the resulting AOs are conceptualized into the SOUR system;
3. the references are identified as links under the current conceptualization.

As an example, consider the access to the following address:

http://www.di.uminho.pt/cnw3.html

Directly from the URL it is possible to extract the following AO information:

AO Name	http://www.di.uminho.pt/cnw3.html
AO Address	/cnw3.html
AO Type	HTML
AO Class	HTTPHTML
Domain_0	www
Domain_1	di
Domain_2	uminho
Domain_3	pt

Now consider that the HTML file identified by the previous URL is the following:

```
<HTML>

<HEAD> <TITLE> WWW National Conference </TITLE> </HEAD>

<BODY>

<H1>
<CENTER>
WWW National Conference <P>
<IMG ALIGN=MIDDLE SRC="/IMI/imi2-ing-interlace.gif"> <P>
Internet Multimedia Information
</CENTER>
</H1>

<H2>
<CENTER>
July 6-8, 1995 <P>
<A HREF="http://www.di.uminho.pt/english-um.html">Minho University</A> <P>
<A HREF="http://s700.uminho.pt/braga.html>Braga</A>,
<A HREF="http://s700.uminho.pt/homepage-pt.html>Portugal</A>
</CENTER>
</H2>
```

```
</BODY>

</HTML>
```

From the analysis of the HTML source text we obtain the following references:

1. <IMG ALIGN=MIDDLE SRC="/IMI/imi2-ing-interlace.gif">
2. <A HREF="http://www.di.uminho.pt/english-um.html">
3. <A HREF="http://s700.uminho.pt/braga.html>
4. <A HREF="http://s700.uminho.pt/homepage-pt.html>

References 1 and 2 will be analyzed in detail in the next section. The other two (references 3 and 4) will originate AOs

AO Name	http://s700.uminho.pt/braga.html
AO Address	/braga.html
AO Type	HTML
AO Class	HTTPHTML
Domain_0	s700
Domain_1	uminho
Domain_2	pt

and

AO Name	http://s700.uminho.pt/homepage-pt.html
AO Address	/homepage-pt.html
AO Type	HTML
AO Class	HTTPHTML
Domain_0	s700
Domain_1	uminho
Domain_2	pt

Finally, these two AOs will produce the *inlinks*:

- http://s700.uminho.pt/braga.html $\xrightarrow{``PartOf''}$
- http://s700.uminho.pt/homepage-pt.html $\xrightarrow{``PartOf''}$

which will become part of the conceptualization of the current URL.

Figure 3 shows the result of the conceptualization of the URL "http://www.di.uminho.pt/cnw3.html". This figure displays both the links and the comparison relations among AOs.

3.5 AO Members

In a similar way, the source text of the URL currently being accessed can also be used to extract AO member information. Each hyperlink to a *local* file [9] will be identified as a member of the AO that maps the current URL. Possible references include:

Hyperlink references — e.g.<A HREF = "FILE">...</A>;

Image references — e.g.<IMG SRC = "FILE">...</IMG>;

[9] A reference to a file that is stored in the current server.

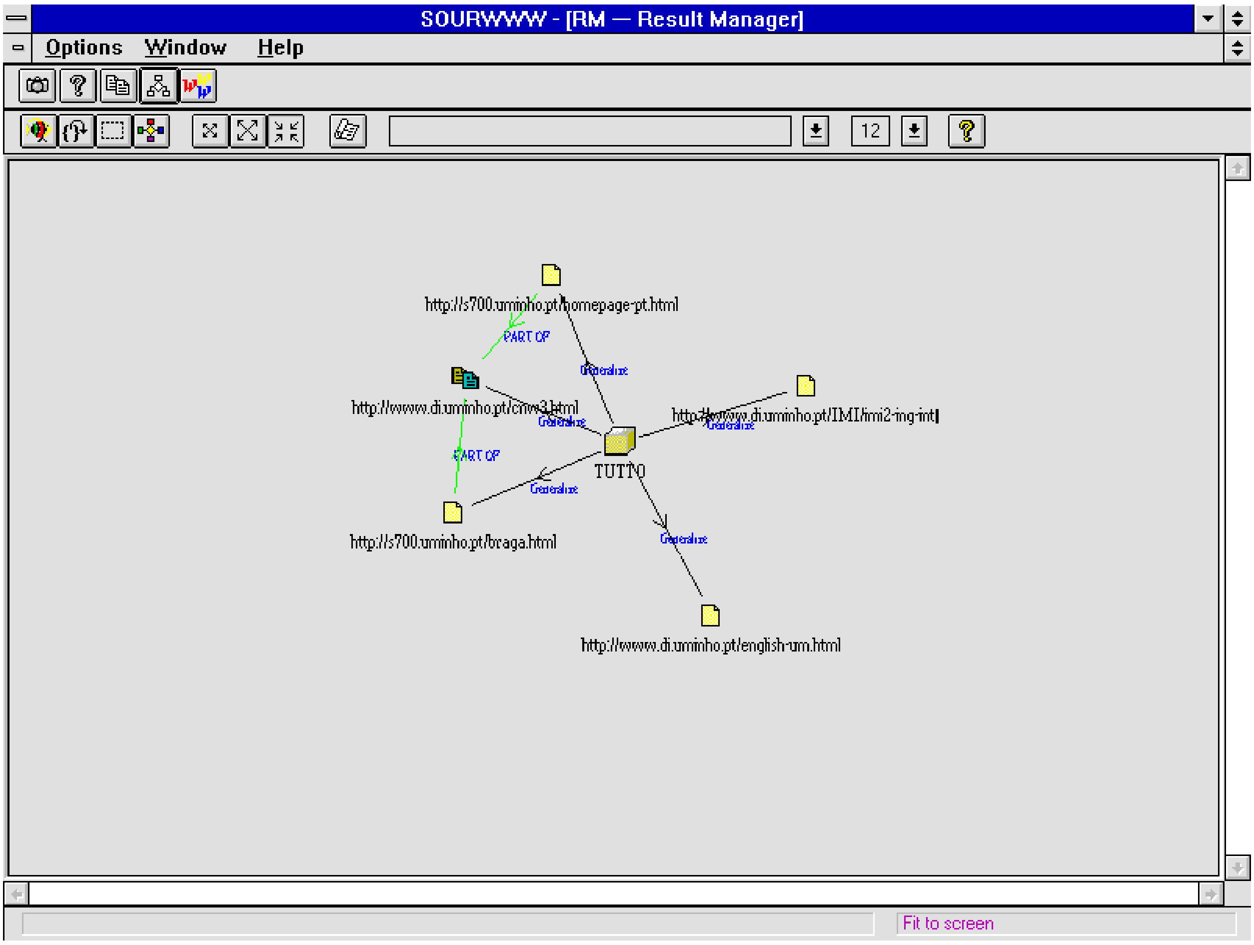

Figure 3: AO Links

Embedded references — e.g.<EMBED SRC = “FILE”>.

All these references will create *member links* identified by the label “Member Of”. In the running example above, references 1 and 2 from the previous section,

- <IMG ALIGN=MIDDLE SRC=”/IMI/imi2-ing-interlace.gif”>
- <A HREF=”http://www.di.uminho.pt/english-um.html”>

will produce AOs

AO Name	http://www.di.uminho.pt/IMI/imi2-ing-interlace.gif
AO Address	/IMI/imi2-ing-interlace.gif
AO Type	GIF
AO Class	HTTPGIF
Domain_0	www
Domain_1	di
Domain_2	uminho
Domain_3	pt

and

AO Name	http://www.di.uminho.pt/english-um.html
AO Address	/english-um.html
AO Type	HTML
AO Class	HTTPHTML
Domain_0	www
Domain_1	di
Domain_2	uminho
Domain_3	pt

Finally, these two AOs will produce the *member links*:

- http://www.di.uminho.pt/IMI/imi2-ing-interlace.gif $\stackrel{``MemberOf''}{\longrightarrow}$
- http://www.di.uminho.pt/english-um.html $\stackrel{``MemberOf''}{\longrightarrow}$

which will become part of the conceptualization profile of the current URL. Figure 4 shows the RM's **Zoom In** graphical functionality [19] operating on URL "http://www.di.uminho.pt/cnw3.html".

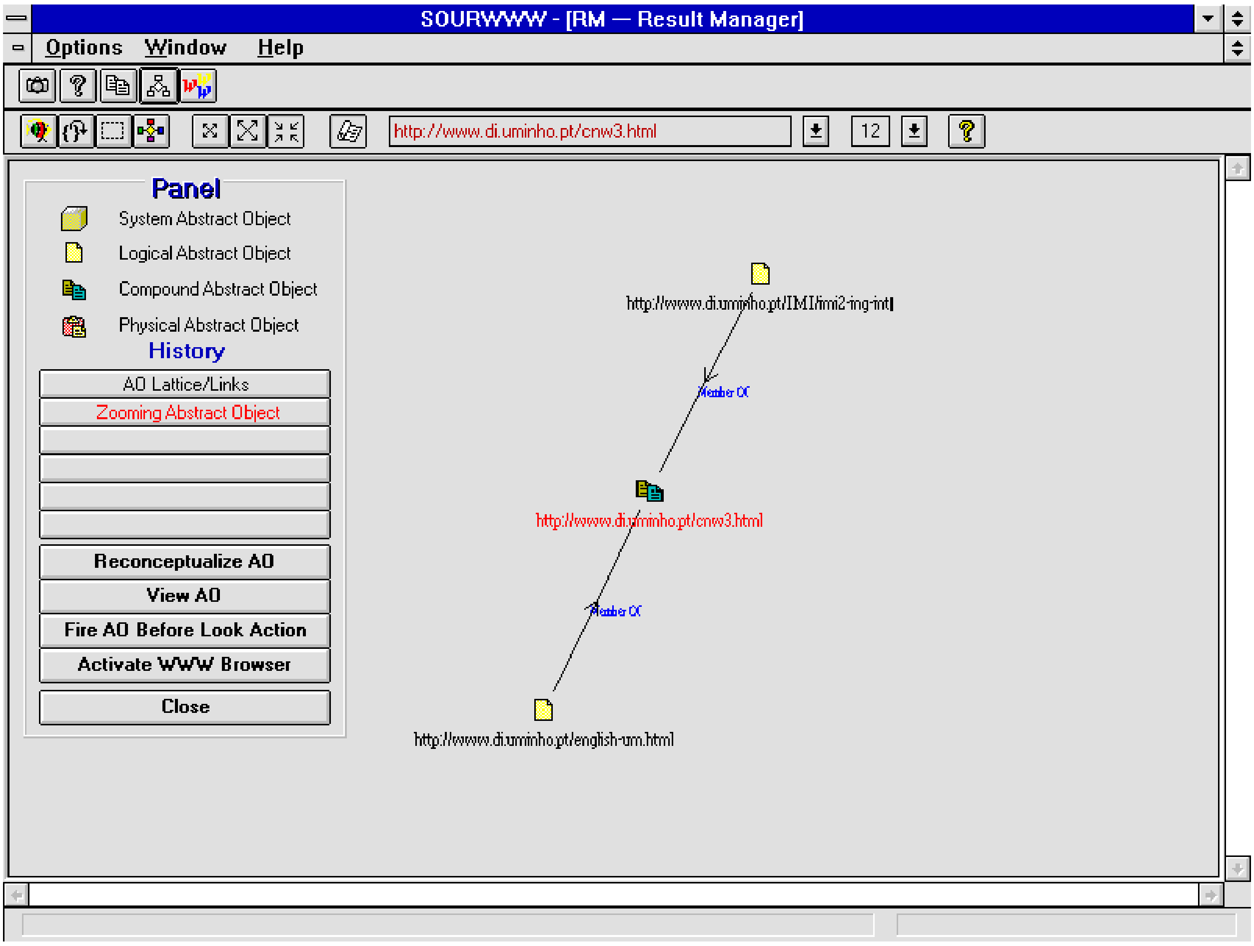

Figure 4: AO Members

4 The Query Mechanism

The INTELLIGENT QUERY SYSTEM is the SOUR's subsystem intended for consulting SOUR's information [18]. It supports an assisted query mechanism for retrieving information based on standard attribute and "fuzzy" query templates. While the former inspects directly the Internet Navigation hypertext structure, the later looks deeper allowing searches based on the contents of the target objects. In the running example of the previous sections, the following query

> *Get all the* URLs *stored in Portuguese servers that use the HTTP protocol and which make references to the file "http://s700.uminho.pt/homepage-pt.html" (the Portuguese home page).*

reflects the linked structure of the resulting URL (see figure 3), while the following one is based on the contents of the referenced data:

> *Get all the the documents talking about "WWW" with fuzzyness level greater or equal to 60%.*

This last query uses the *facet* values of each AO currently stored in the repository and the conceptual relations illustrated in figure 2, in order to find the objects within the specified fuzzyness level.

The SOUR query mechanism combines both the expressive power of the SQL and *fuzzy-logic* searching techniques. The outcome is something we might call a "fuzzy SQL processor" with a highly assisted, interactively user interface [19].

5 AO Comparison

The notion of "proximity", which leads us to the idea of arrangement or grouping, is crucial for the classification problem, in general, and for document organization and retrieval in particular. As documents become more and more the center of computer activity, it will be of a dramatic importance its identification, store, track, retrieve and presentation [13].

The COMPARATOR subsystem of SOUR[14] is encharged of maintaining an ordered structure of AOs which are grouped hierarchically according to a "proximity" order defined on the system's standard of conceptualization. It performs the crucial task of comparing Abstract Objects (AOs), while providing a meaningful decision procedure for AO-equivalence. The relevance of COMPARATOR cannot be underestimated — it amounts to the definition of AO-semantics itself, based upon the belief that document semantics can be effectively captured by the adopted *attributive* model.

The standard attribute-based comparison, at *coarse* level, is present as a preliminary, less discriminant decision procedure. But with such a procedure the expressive power of the system does not go beyond the conventional, object-oriented information model.

The desired increase in expressive power is achieved at *fine* level, where object comparison is "fuzzy" and is decided according to a metrics or algebra of proximity which computes intersections of the proximity closures of facet values within their hierarchical conceptual graphs. For the technical aspects of this sophisticated tool, see [14] and [10].

6 Implementation Details

The NETSCAPE Client APIs (NCAPIs) are provided as part of version 1.1N release of NETSCAPE. They are designed to allow third party applications to remotely control the NETSCAPE Navigator client [9]. This mechanism includes both the so-called OLE AUTOMATION and DDE Protocol mechanisms.

Wherever interacting with these APIs, SOUR will regard its object repository as stored in a dynamically "extended file system" across the Internet. For that, SOUR and NETSCAPE will cooperate based on the client/server technology supported by the DDE implementation of NETSCAPE version 1.1N [2]. In this way, SOUR will manipulate NETSCAPE to execute and/or extract the information of a given URL.

The first step is to gain access to the Netscape's Ole Automation object (the **Netscape.Network.1 Automation Object** to be more specific [1]). Using this object, Sour will be able to access network data through the same mechanisms Netscape uses. However, Netscape's Ole Automation does not provide the functionality necessary to manipulate the Netscape Navigator user interface. This is possible only by using the Dde protocol which will make Sour to act simultaneously like a Netscape client and server, as illustrated in figure 5.

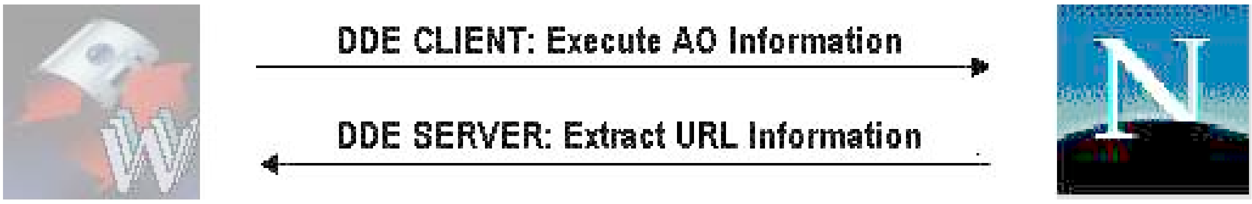

Figure 5: DDE Protocol

While acting as a Netscape's client, Sour uses Netscape as a displayer for the Urls which have been conceptualized. When it is working as a Netscape's server, Sour is notified every time the loading of a Url occurs [10]. After the notification, the Url is mapped to a Sour AO (see section 3) while the Html source text is saved for further analysis (see the sections 3.3,3.4 and 3.5 earlier in this paper). After all these steps, the AO information is finally described in the *Conceptualization Batch Language* format [17] and saved into a text file. Finally, whenever Sour becomes the active application, it will verify and load all the files created by this process.

Once satisfactorily conceptualized, each Url will be classified in the system's repository as a conventional Sour object. After that, it will be possible to use the Sour software system both for have access to its standard functionality (available in the release β of the system [19]) or to launch a "batch" Netscape navigation session, which is available through a specially developed capability of Sour's Rm (Result Manager) subsystem. Rm is a generic Sour service-tool for graphically displaying, consulting and executing AO's related information which, in the present context, becomes also a graphical environment for browsing the Internet linked structure (see figures 3 and 4).

Figure 6 shows the result of activating the Netscape Navigator for displaying a selected AO. This operation loads the correspondent Url (using the Ole Automation mechanism) into the current Netscape window.

7 Conclusions and Future Work

This paper presents some modifications and extensions to the current Sour prototype (running on Windows [19]) in order to make it a useful tool for the classification, storing and retrieval of Internet information.

The key aspects of the Sour information model reflects the way in which documents are regarded today, no longer just mere files, but rather as books of pointers to objects of several kinds [13]. On the one hand, it can provide some important mechanisms in order to organize and, consequently, make the Internet navigation simpler. On the other hand, as the access to the information becomes easier and powerful, it can act like a personal tool for getting the information directly from the Web and, at the same time, to store and arrange it in the personal workstation into a more human-based organization model.

Among the topics discussed in the paper, probably the most complex is designing a general classification framework for arbitrary documents. However, the approach adopted by Sour concerning software reuse in particular [12, 11], as well as recent studies on fuzzy object-comparison [10] offer good perspectives for the future.

[10] Look for the topics **WWW_RegisterURLEcho**, **WWW_URLEcho** and **WWW_UnRegisterURLEcho** in [2] for more details.

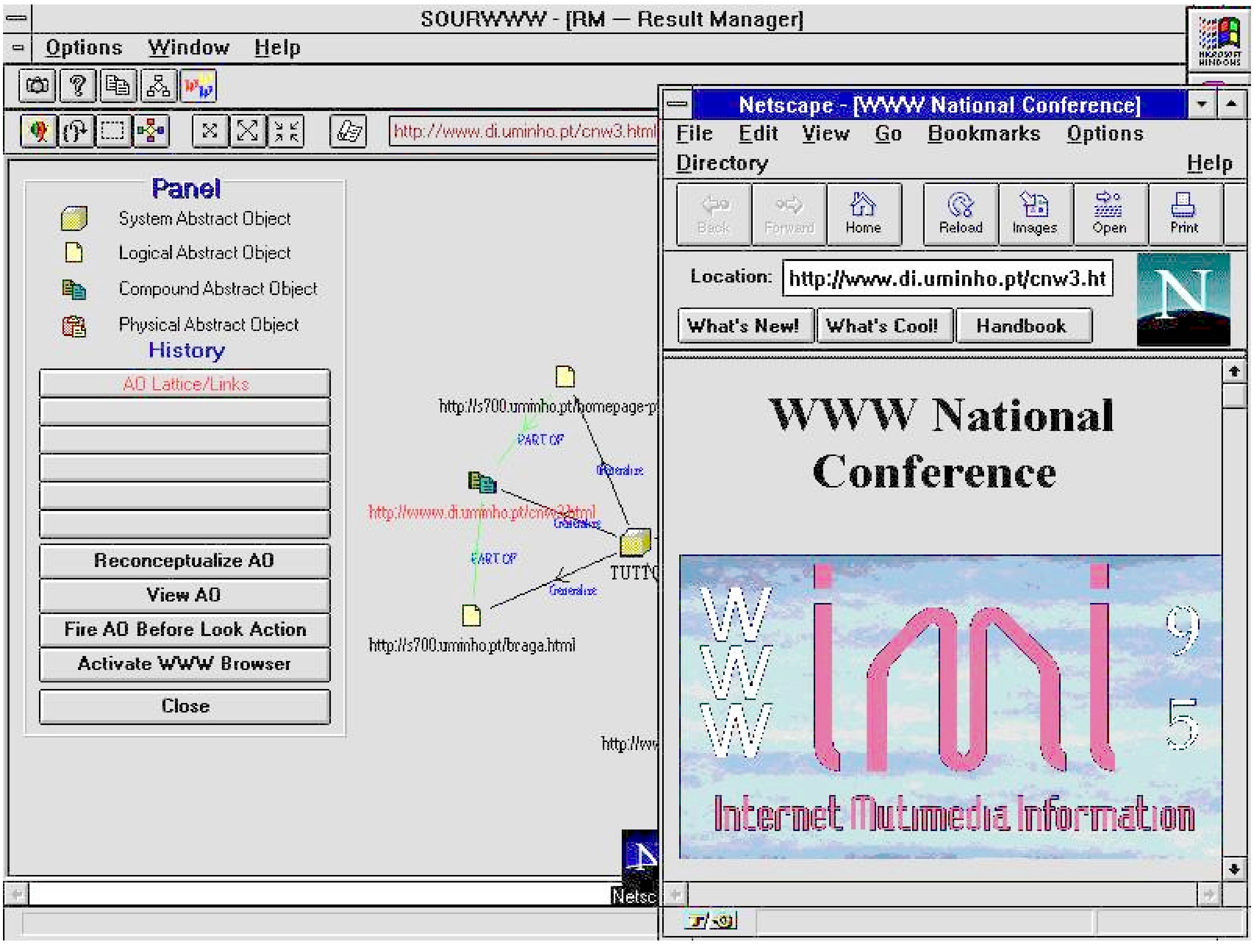

Figure 6: Activating NETSCAPE Browser

Future work includes the prospect of “globalizing” the adopted classification strategy. This means to scale up the approach from personal to world-wide classifiers. Some similarity between AOs and UNIVERSAL RESOURCE CITATIONS (URCs) [22] suggests that the SOUR AO profile-based paradigm can be scaled-up to a world-wide, Internet resource-based “yellow-page”-like service of bibliographic metadata about WWW documents. Naturally, URCs would have to be extended with fuzzy attributes. But performance feasibility will have to be studied beforehand.

8 Acknowledgements

The authors wish to thank all the colleagues in the SOUR consortium (INESC, SYSTENA, SSS and OIS RICERCA) who contributed to the many discussions along the project's lifetime. On the implementation side, comments by Garret Arch Blythe and Steve Caine are gratefully acknowledged.

References

[1] Garret Arch Blythe. OLE Automation in NETSCAPE. Technical report, Netscape Communications Corporation, March 1995. URL: <http://home.netscape.com/newsref/std/oleapi.html>.

[2] Garret Arch Blythe. NETSCAPE's DDE Implementation. Technical report, Netscape Communications Corporation, March 1995. URL: <http://home.netscape.com/newsref/std/ddeapi.html>.

[3] Earlene Busch. Managing Infoglut: Search and Retrieval. In Dennis Allen, editor, *BYTE*. McGraw-Hill, 1992.

[4] Bob Friesenhahn. Build Your Own WWW Server. In Raphael Needleman, editor, *BYTE*. McGraw-Hill, 1995.

[5] Mark Handley and Jon Crowcroft. The World Wide Web - Beneath the Surf. Technical report, UCL Press, 1994. URL: <http://www.cs.ucl.ac.uk/staff/jon/book/book.html>.

[6] A Beginner's Guide to HTML. Technical report, Nacional Center for Supercomputing Applications. URL: <http://www.ncsa.uiuc.edu/demoweb/html-primer.html>.

[7] Donald E. Knuth. *The TEXbook*. Addison-Wesley, Reading, Massachusetts, 1984.

[8] Thomas M. Koulopoulos. Managing Infoglut: Search and Retrieval. In Dennis Allen, editor, *BYTE*. McGraw-Hill, 1992. See the text box "Document Clustering".

[9] NETSCAPE Client APIs (NCAPIs) 2.0. Technical report, Netscape Communications Corporation, March 1995. News & References, Standards Documentation. URL: <http://home.netscape.com/newsref/std>.

[10] José Nuno Oliveira. Fuzzy Object Comparison and Its Application to a Self-Adaptable Query Mechanism. 1995. Invited paper, Sixth International Fuzzy Systems Association World Congress.

[11] R. Prieto-Diáz. Implementing Faceted Classification for Software Reuse. *CACM*, 34(5):89–97, May 1991.

[12] R. Prieto-Diáz and P. Freeman. Classifying Software for Reusability. *IEEE Software*, 4(1):6–16, January 1987.

[13] Andy Reinhardt. Managing the New Document. In Dennis Allen, editor, *BYTE*. McGraw-Hill, 1993.

[14] Syntax Sistemi Software. Comparator and Modifer — Functional Specification and Architecture. Technical report, SOUR Project, 1993. Ver.1.4, © SSS, Via Fanelli 206-16, Bari, Italy.

[15] Systena. Context Thesaurus Subsystem — Requirement Specification. Technical report, SOUR Project, September 1993. Ver.1.1, © Systena, Via Zanardelli 34, Rome, Italy.

[16] Systena. Lexicon/Thesaurus Subsystem — Requirement Specification. Technical report, SOUR Project, July 1993. Ver.2.0, © Systena , Via Zanardelli 34, Rome, Italy.

[17] Systena. Conceptualizer — Functional Specification and Architecture. Technical report, SOUR Project, 1994. Ver.2.1, © Systena, Via Zanardelli 34, Rome, Italy.

[18] Systena. Intelligent Query System — Functional Specification and Architecture. Technical report, SOUR Project, 1994. Ver.2.1, © Systena, Via Zanardelli 34, Rome, Italy.

[19] Systena and Syntax Sistemi Software. Integrated SOUR Software System — Demo Session Manual. Technical report, SOUR Project, 1994. Ver.1.2, © Systena & SSS, Via Zanardelli 34, Rome & Via Fanelli 206-16, Bari, Italy.

[20] Haviland Wright. Managing Infoglut: SGML Frees Information. In Dennis Allen, editor, *BYTE*. McGraw-Hill, 1992.

[21] The World Wide Web. Technical report, World Wide Web Consortium. URL: <http://www.w3.org/>.

[22] WWW Names and Addresses, URIs, URLs, URNs. Technical report, World Wide Web Consortium. URL: <http://www.w3.org/hypertext/WWW/Addressing/Addressing.html>.