



CLASSIFYING INTERNET OBJECTS

F. Luís Neves, José N. Oliveira

Abstract

Navigation across the Internet may be an arduous task. Although bookmark and history mechanisms, available in browsers like Netscape and Mosaic, are a help, there is an absence of a classification scheme for the enormous amount of information available through billions of URLs. 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 Netscape's OLE and Automation DDE interprocess communication mechanisms; these allow third party applications to remotely control the Netscape navigator client.

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 is arduous because of the absence of an effective classification scheme for the enormous amount of information available through billions of interlinked 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 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 [16, 15, 14, 18, 17, 19].

The unit of information in SOUR is the so-called *Abstract Object* (AO), a notion which combines the *enumerative* and *faceted* classification schemes [10, 11, 12] 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:

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

where *scheme* is one of:

file: a 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: a 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.

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*. Multifaceted classification schema based on fuzzy concept-network reasoning, which extends conventional keyword-oriented classification

The conceptualization approach is thus a combination of the enumerative and faceted classification schemes [12, 11], whereby a physical object like a piece of C-code “becomes” an AO. This is accomplished by attaching a *profile* to that object which consists of the following basic items:

- *Classname (code)*. This item plugs the object into the system’s standard “is-a” hierarchy.
- *Attributes (author)*. These provide values for features or properties relevant about the objects, under a simple inheritance mechanism prescribed by the “is-a” hierarchy.
- *Links (implements)*. These are AO-valued attributes which establish relevant relationships among AOs.
- *Facets (system=text-formatter)*. These are fuzzy tuples of terms acting as “vague keywords” that index AOs.
- *Actions (compile)*. These are “method-like” AO-attributes recording predefined procedures. AOs either are automatically submitted at standard instants of their life cycle or triggered by UI-events such as mouse double-clicking.
- *Members* (if an AO container). Collection of references to the subobjects contained in the object.
- *Relations* (if an AO cluster). Graph structure of links interrelating to the subobjects involved with the main object.

Every AO has a unique identity represented by its *Abstract Object Identifier* (AOID). AOIDs are

managed by the system and are transparent to the end user.

The Conceptualization Standard

It should be noted that two different physical objects may happen to be attached to the same conceptual profile. If this is the case, it 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 top of this hierarchy is *Abstract Object Generic attributes* (AOG), a class consisting of system controlled attributes such as the following:

name

AO name as defined by the user who created the object

pathname

Plugs Physical AOs (PAOs) into the underlying file system

inserted By

Username of creator

lastModifyDate

Date of last update

lastLookDate

Date of last browse

nModify

Number of updates so far

Facets are sextuples of terms, each term instantiating one of the predefined facet types shown in Figure 1.

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

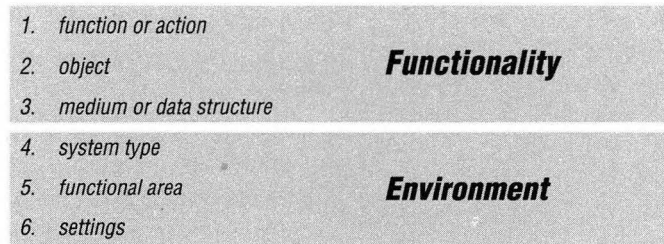


Figure 1: SOUR Overall Architecture

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]. As a result, every Internet transaction is divided in two distinct phases: an *identification* phase, where the server validates the URL specification, and a *retrieval* phase, where the server delivers the corresponding data to the client.

If these two phases are successfully executed it is possible to access both the (now valid) URL and the data. Data access has a particular importance in the present study each time the URL identifies an HTML text file. In such cases, an analysis reveals that some parts of the text may be used for conceptualizing the URL.

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

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 as shown below:

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, a URL alone provides the minimal information needed for a successful conceptualization [17]. However, if the URL is an HTML text file, then some extra information may be added according to its contents, otherwise no more information will be attached (see the following sections for more information).

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 (*www, gopher, ftp, s700*)

Domain_1 (*ncsa, telepac, inescn, di*)

Domain_2 (*uiuc, inesc, uminbo*)

... (...)

Domain_n (*com, pt, org, edu*)

While the URL's subclasses reflect possible scheme values and filename extensions, 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 illustrates this idea and specifies some possible subclasses for the HTTP class.

URL

FILE

HTTP	HTTPHTML	<i>Html documents</i>
	HTTPTXT	<i>Text documents</i>
	HTTPPS	<i>Postscript documents</i>
	HTTPDOC	<i>Word documents</i>
	HTTPTEX	<i>TeX documents</i>
	HTTPGIF	<i>Gif images</i>
	HTTPZIP	<i>Zipped files</i>
GOPHER	...	
	...	
WAIS	...	
	...	
NEWS	...	
	...	
TELNET	...	
	...	

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

AO Facets

Faceted classification as proposed in this paper combines several text-scheme management tools such as full text indexing and retrieval, free-text scan, document clustering, unique word, and vector-space [3]. These approaches are discussed below.

The full text approach first generates a list of strings associated with a document. Then, at retrieval time, 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 the section entitled "AO Comparison").

Together, all these notions provide a default facet classification that will be tried by 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 an 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 the elements <TITLE> and </TITLE>
- *Headings*. Words between the elements <H_y> and </H_y> where *y* is a number between 1 and 6 specifying the level of the heading

Since we are interested in 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*, *HTML*, and so on. This specialization of lexical terms, which can improve both the conceptualization and the query mechanism, is supported by SOUR's capability of working with several CTS/LTS repositories.

CTS provides the capability to cope with features of human reasoning such as *classifying by analogy* and *terminological vagueness* [10][16][15]. In particular, lexical terms can be connected by conceptual distances interrelating terms (words) according to their contextual meanings. 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.

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 chances of missing important information [3].

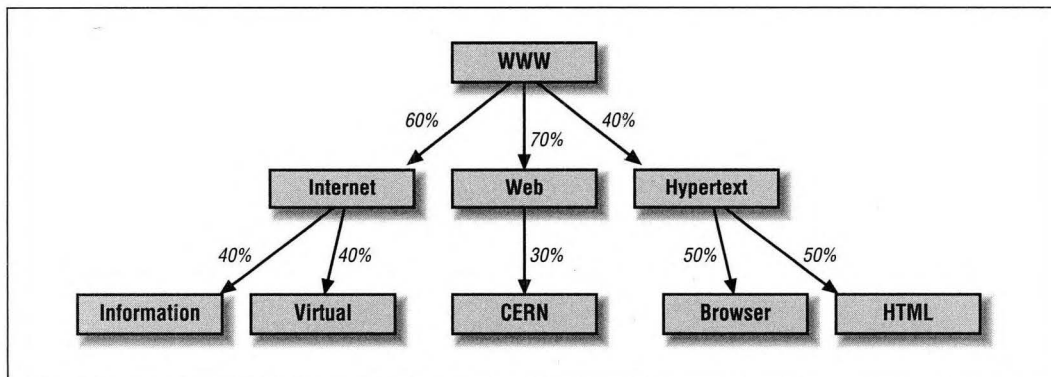


Figure 2: Example of conceptual relations

The 6-tuples of predefined terms presented earlier (see the previous section entitled “About the Conceptualization Standard”) were designed by Prieto-Díaz for the specific task of software classification. How to extend or adapt them to generic information as accessed through the Internet is an open problem.

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.

AO Links

In this section, we show how the overall *semantic network* structure of the 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 will be identified as an *inlink* of the AO that abstracts the current URL. Possible references include:

- *Hyperlink references*
 - e.g., ...

References of this kind will create links identified by the “Part Of” label.

- *Image references*
 - e.g., ...

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 this article.
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

You can extract the following AO information directly from the URL:

AO Name: *http://www.di.uminho.pt/cnw3.html*

AO Address: */cnw3.html*

AO Type: *HTML*

AO Class: *HTTPHTML*

Domain0: *www*

Domain1: *di*

Domain2: *uminho*

Domain3: *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.
2.

3.

4.

References 1 and 2 will be analyzed in detail in the next section. References 3 and 4 will originate AOs as follows:

AO Namez: *http://s700.uminho.pt/braga.html*

AO Address: */braga.html*

AO Type: HTML

AO Class: HTTPHTML

Domain0: *s700*

Domain1: *uminho*

Domain2: *pt*

and:

AO Name:

http://s700.uminho.pt/homepage-pt.html

AO Address: */homepage-pt.html*

AO Type: HTML

AO Class: HTTPHTML

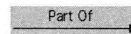
Domain0: *s700*

Domain1: *uminho*

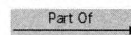
Domain2: *pt*

Finally, these two AOs will produce the inlinks:

- *http://s700.uminho.pt/braga.html*



- *http://s700.uminho.pt/homepage-pt.html*



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.

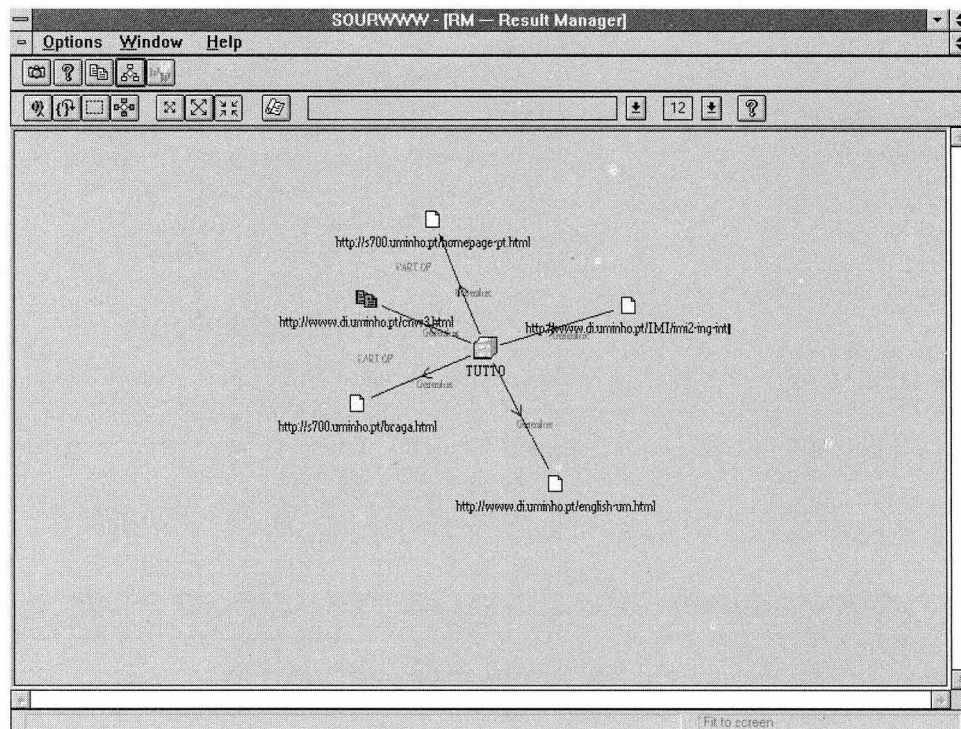


Figure 3: AO links

AO Members

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

- *Hyperlink references*
 - `...`
- *Image references*
 - `...`
- *Embedded references*
 - `<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,

- ``
- ``

will produce the following 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*

Domain2: *uminho*

Domain3: *pt*

and:

AO Name:
<http://www.di.uminho.pt/english-um.html>

AO Address:
</english-um.html>

AO Type: *HTML*

AO Class: *HTTPHTML*

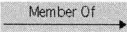
Domain0: *www*

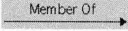
Domain1: *di*

Domain2: *uminho*

Domain3: *pt*

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

- *<http://www.di.uminho.pt/IMI/imi2-ing-interlace.gif>* 

- *<http://www.di.uminho.pt/english-um.html>* 

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>*.

The Query Mechanism

The Intelligent Query System is the SOUR subsystem intended for consulting SOUR's information [18]. It supports an assisted query mechanism for retrieving information based on standard attributes and "fuzzy" query templates. While the

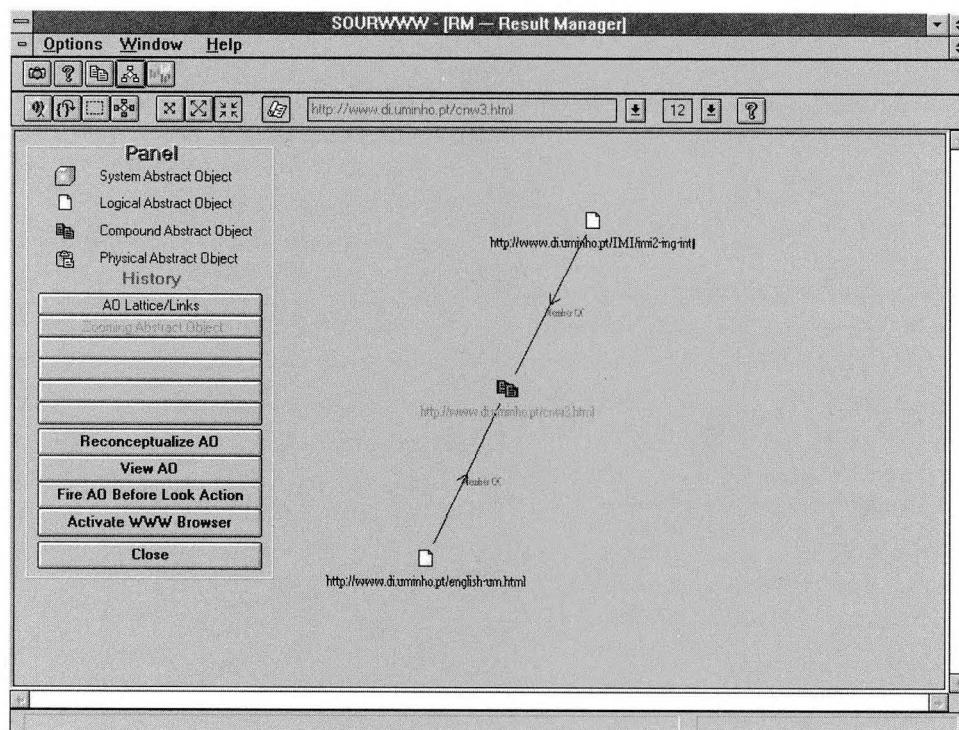


Figure 4: AO members

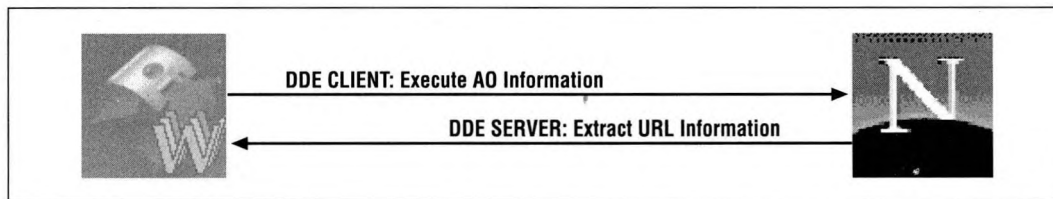


Figure 5: DDE protocol

former directly inspects the Internet Navigation hypertext structure, the latter 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 that 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 query is based on the contents of the referenced data:

Get all the the documents that talk about "WWW" with fuzziness 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 fuzziness 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, interactive user interface [19].

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, their identification, storage, tracking, retrieval, and presentation [13] will be of dramatic importance.

The COMPARATOR subsystem of SOUR[14] maintains an ordered structure of AOs that 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 that 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].

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 Ole Automation and the DDE Protocol mechanisms.

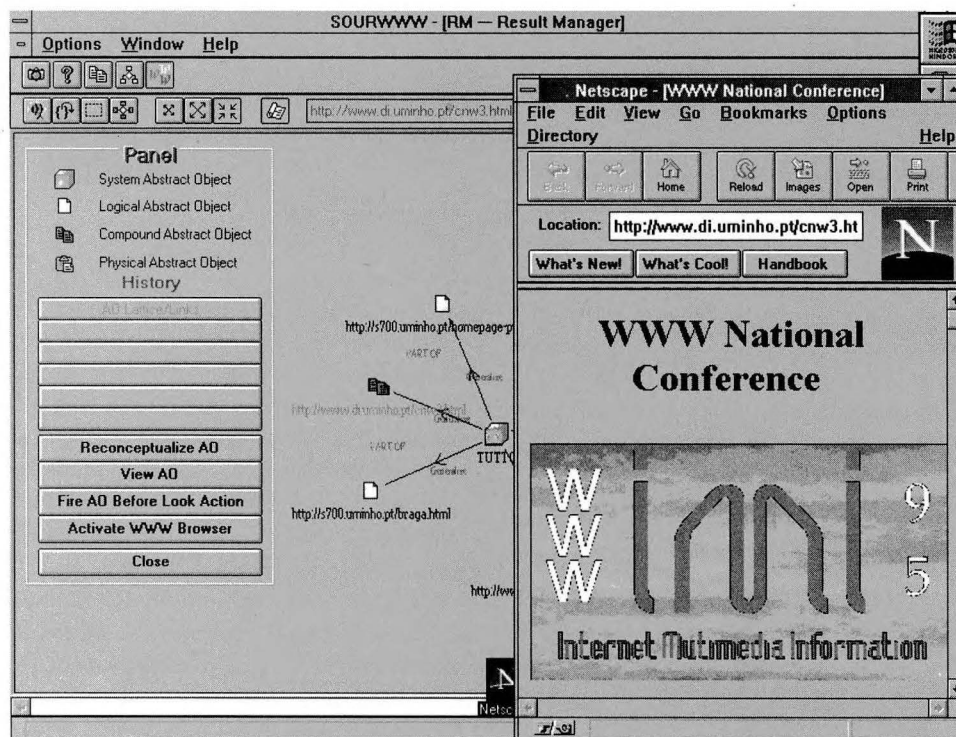


Figure 6: Activating Netscape browser

Whenever interacting with these APIs, SOUR will regard its object repository as if it is 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 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 act simultaneously like a NETSCAPE client and

server, as illustrated in Figure 5. While acting as a NETSCAPE client, SOUR uses NETSCAPE to display the URLs that have been conceptualized. When it is working as a NETSCAPE server, SOUR is notified every time the loading of a URL occurs. After the notification, the URL is mapped to a SOUR AO (see the section entitled “Mapping URLs to AOs” while the HTML source text is saved for further analysis (see the descriptions 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. 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 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, also 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 to display a selected AO. This operation loads the correspondent URL (using the OLE AUTOMATION mechanism) into the current NETSCAPE window.

Conclusion 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, storage, and retrieval of Internet information.

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

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 [11][12], as well as recent studies on fuzzy object-comparison [10] offer good perspectives for the future.

Future work includes the prospect of "globalizing" the adopted classification strategy. This means scaling 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. Before that can happen, however, performance feasibility will have to be studied. ■

Acknowledgments

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.
<http://home.netscape.com/newsref/std/oleapi.html>
2. Garret Arch Blythe's Implementation. Technical report, Netscape Communications Corporation, March 1995.
<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.
<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.
<http://www.ncsa.uiuc.edu/demoweb/html-primer.html>

7. Donald E. Knuth. *The TeX Book*. Addison-Wesley, Reading, Massachusetts, 1984.
 8. Thomas M. Kouloupoulos. 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.
<http://home.netscape.com/newsrefs/>
 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-Díaz. Implementing Faceted Classification for Software Reuse. *34(5):89-97, May 1991*.
 12. R. Prieto-Díaz 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 Modifier—Functional Specification and Architecture. Technical report, SOUR Project, 1993. Ver.1.4, © SSS, Via Fanelli 206-16, Bari, Italy.
 15. Systema. Context Thesaurus Subsystem—Requirement Specification. Technical report, SOUR Project, September 1993. Ver.1.1, © Systema, Via Zanardelli 34, Rome, Italy.
 16. Systema. Lexicon/Thesaurus Subsystem—Requirement Specification. Technical report, SOUR Project, July 1993. Ver.2.0, © Systema, Via Zanardelli 34, Rome, Italy.
 17. Systema. Conceptualizer—Functional Specification and Architecture. Technical report, SOUR Project, 1994. Ver.2.1, © Systema, Via Zanardelli 34, Rome, Italy.
 18. Systema. Intelligent Query System—Functional Specification and Architecture. Technical report, SOUR Project, 1994. Ver.2.1, © Systema, Via Zanardelli 34, Rome, Italy.
 19. Systema and Syntax Sistemi Software. Integrated SOUR Software System—Demo Session Manual. Technical report, SOUR Project, 1994. Ver.1.2, © Systema & 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.
<http://www.w3.org/>
 22. WWW Names and Addresses, URIs, URLs, URNs. Technical report, World Wide Web Consortium.
<http://www.w3.org/hypertext/WWW/Addressing/Addressing.html>
- F. Luís Neves and José N. Oliveira, “Classifying Internet Objects” in WWW National Conference '95, Minho University, Braga, Portugal

About the Authors

F. Luís Neves and José N. Oliveira
INESC Group 2361 and Dep. Informática
Universidade do Minho
4700 Braga, Portugal