Acta Cybernetica 14 (1999) 389-402.

A study of portability in the deployment of WWW

András Micsik *

Abstract

There are several problems with handling compound hypermedia documents on the Internet currently. One class of these problems, namely portability is studied in this paper in detail, and possible solutions are examined. The offered solutions are based on a new container architecture, the WebPack format, currently under development at SZTAKI.

Keywords: hypermedia documents, document-like objects (DLO), World Wide Web, portability, metadata, Internet

1 Introduction

As the World Wide Web [1] spread the world from the beginning of this decade, it incorporated more and more powerful tools and formats, and the information served via WWW became more and more complex. The content and layout of WWW pages became competitive with printed material, and in other aspects WWW pages have far more potential than printed documents. Searching, interactivity, animations and virtual reality are just keywords to make the additional possibilities felt.

The meaning of document in case of the Internet is changing. Digital documents are sometimes more similar to a piece of software than to printed material. Furthermore these documents on the Internet are interconnected with each other via hyperlinks. The Dublin Metadata Workshop [15] investigated this new kind of information source, and created a new term: Document-like Object (DLO) [14]. A DLO can be characterized like this:

- it may contain files in lots of different formats: text, graphics, animation, video, audio and 3D models
- its files and data are interconnected with hyperlinks.
- it may contain executable parts (applets, scripts, objects, etc.)

*Department of Distributed Systems, MTA SZTAKI, 1111 Budapest XI. Lágymányosi u. 11. Hungary, e-mail: micsik@sztaki.hu

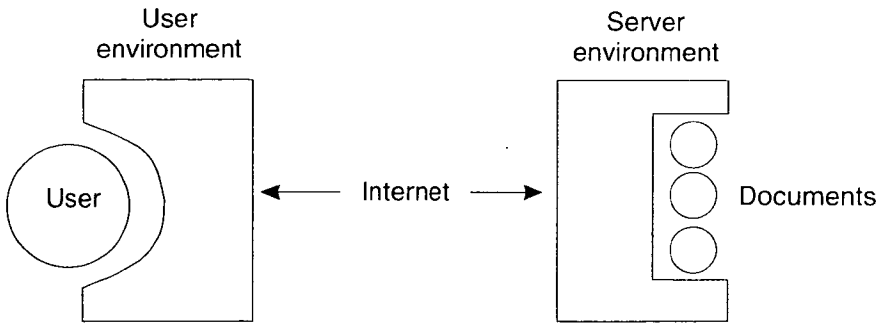

Figure 1: Using WWW documents

DLOs are called WWW documents or simply documents for simplicity in this paper. WWW documents can be surprisingly immovable compared to other widely used formats such as Postscript, Microsoft Word or ToolBook. This is because these documents may depend on the presence of several files and programs, and they may also depend on services of the WWW server. Currently there is no general approach to provide collection and management of these dependencies and therefore the management of WWW documents is solved individually by web administrators. Management in general is not in the scope of this paper, rather one of its subtopics, portability. When moving a document from one host to another, all dependencies in a WWW document must be explored and fixed in order to restore correct operation. The nature of portability dependencies is analyzed in Section 2. A formal descriptive framework for poratbility conditions is given in Section 3, and technical solutions for the identified problems are given in Section 4. An outlook to related work and the summary concludes the paper.

2 Analysis of portability issues

Users access WWW documents through the user and server environments. The *user environment* is specific to the actual user of the document, while the *server environment* is specific to the used document. The user environment provides browsing and viewing capability for the user. The server environment offers the services of the operating system and the WWW server for the document. A server environment usually serves several documents.

The *correct operation* of a WWW document means that all its features (hyperlinks, images, interactive parts, etc.) are available for the user in the same way as the author has implemented them. This relies on both the user and server environments, and may rely on other WWW documents or external data as well. As the user environment is totally under the control of the user, this approach concentrates on the correct operation of the server side. If the operation of the server side is based on open standards, it is possible to give recommendations for the user

environment as well.

In case of a WWW document *portability* means that if the document is placed into a new server environment, or its server environment has changed, there is a way to keep the same operation of the document as in its previous environment. In the next subsection the most generally used formats and technologies are listed together with the problems that jeopardize the portability of WWW documents.

2.1 Summary of formats, technologies and portability problems

URIs and URLs Embedded or linked objects are most commonly referred to as Uniform Resource Locators (URLs) [5] on the Internet. A URL can refer to

- an object on the same host
- an object on a different host
- a part of an HTML page
- dynamically created objects

The problem with URLs is that they are location specific. The referred object is identified by the combination of the host machine name and the descriptor of the object in the file system. This means that moving a part of a document to another location in the file system or to another host can make that part inaccessible.

URLs are a subclass of Uniform Resource Identifiers (URIs) [2]. The URI schema defines a highly customizable reference methodology for the Internet, which would allow location transparent naming facilities, but currently there isn't any widely used and location transparent naming facility for the Internet.

HTML The Hypertext Markup Language [3] serves as a basis for WWW documents. It provides embedding and linking facilities for other digital formats, and usually acts as the main user interface for the whole document. After moving an HTML page to another location URLs for inline images, or URLs to other pages may become incorrect making the page unusable and the referred pages inaccessible.

CGI The Common Gateway Interface [6] is a simple and general mechanism to call executable programs from static WWW pages. The program is identified as a URL, and there are two ways to pass parameters to the program: encoded in the URL or via HTML forms. When the program is launched the calling parameters are passed, and the program output is an object to display for the user. CGI scripts can work only with the help of a WWW server, but the scripts are executed in the context of the operating system. Therefore any kind of compiled or interpreted program can be used to write CGI scripts. For example Perl is a very popular language of CGI.

CGI scripts mean a great problem concerning portability, because the execution is controlled only by the operating system. The program may rely upon other programs or data files, and these may not be present on another host.

Applets Applets are written in Java language [7], and provide a full-featured graphical user interface in a part of the screen during WWW browsing. Java is an interpreted and platform-independent language, so an applet should execute in the same way in every WWW browser on every machine (in practice there are small, not very significant differences among different browsers and different operating systems).

Applets can load parts of their code and additional data files from the server host, and can communicate directly with other network services on the server host. Therefore hidden file dependencies can still occur similarly to CGI scripts.

JavaScript JavaScript [8] offers a way to associate custom code with HTML pages, and in this way to add new functionality to WWW browsers. For example with JavaScript the programmer can change the behaviour of a button in an HTML form, or can load other pages automatically into the browser. Unfortunately JavaScript is not one uniform scripting language. It has no specification, and its support differs remarkably in different browsers (namely Netscape and Internet Explorer) and even in different browser versions.

Server side includes There are several very different tools that are described here commonly as server side includes. The basic server side include facility is a way to include parts into a HTML page on the fly when the page is accessed by the user. Traditionally there were two possibilities: to include another file or to include the output of a program. Later these possibilities were enhanced to enable flexible scripting and database access. Products falling into this category are for example PHP and Microsoft's Active Server Pages. From the portability viewpoint server side scripts require the presence of their specific interpreter environment, and may introduce additional dependencies on data files and executables.

VRML With the use of the Virtual Reality Modeling Language [9] one can integrate 3D models into a WWW document. These 3D models may contain animations, interactive scripts, and hyperlinks to other objects. Models often incorporate external objects (images, sounds, etc.). Links and embedded objects are both represented as URLs, so in this aspect VRML holds the same problems as HTML, furthermore VRML can contain scripts as well, inheriting the portability problems with executables. VRML files are sometimes stored in a compressed format to speed up download. To find external references, these files must be uncompressed temporarily.

The server environment The server environment provides elementary document services called features later in this paper. These include authentication and

access control, handling of object types, aliasing or redirection of URLs, and server side includes or scripting. The configuration of these features differs from server to server, and often needs the editing of complicated configuration files. Additionally the server environment provides a set of installed auxiliary programs: scripting tools, image manipulating software, etc.

The user environment This environment is not only the WWW browser but also a set of external viewer applications for various formats (e.g. Postscript, PDF or VRML) that cannot be viewed within the browser. Also the browser preferences set by the user are very important, here viewers are associated with formats, Java or JavaScript is enabled or disabled, etc. Finally the type and version of the WWW browser determines the capability of JavaScript and Java execution.

3 Describing portability conditions

The first step towards the solution of the above mentioned problems is the formal description of portability conditions. The description is based on the dependence and reference relationships. With these relationships all aspects that are important for portability can be formalized. For these relationships a metadata-based representation is given for automatic processing, and a graphical representation for humans. Metadata literally means data about data [15], data characterizing an object. Typically metadata provides descriptive, administrative and structural information about an object.

The descriptive framework for portability conditions is built with the following terms: the server environment contains a set of WWW documents. A *WWW document* is defined as a set of objects. *Objects* are single-file entities in any format. Each entity (objects, documents, etc.) may have metadata associated with it. *Metadata* describes either relations among objects or properties of objects. Relations may have properties as well. *Properties* are simply name-value pairs.

3.1 Definition of dependence and reference relationships

Saying object X *depends on* object Y means that the correct access and operation of object X requires that object Y is accessible and operates correctly for object X. This definition of "depends" is the transitive closure of the dependence relationship.

Definition 1 *Dependence relationship* (X, Y) *holds for objects* X, Y *if* X *depends on* Y*, but there is no such object* Z *that* X *depends on* Z *and* Z *depends on* Y*.*

Each dependence relation (X, Y) creates an entity called dependency. Objects and dependencies may have properties containing additional information, and they are typed. Type is given as a property according to the following type hierarchy (Fig 2): on the highest level an object can be data, program or feature, a dependency can be read or execute type. On lower levels the type of object is given by a MIME type definition, except for features. Features are special services of the server or user

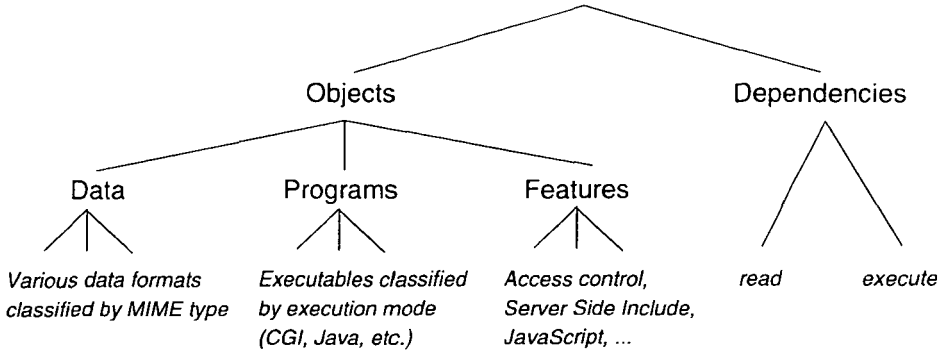

Figure 2: The type hierarchy for the dependence relationship

environment, for example server side include or authorization techniques. In case of dependencies the referencing structure may be given as a property containing the significant HTML or VRML tag, for example an image can be referenced as background (*BACKGROUND*) or inline image (*IMG*).

The reference relationship is defined similarly to the dependence relationship:

Definition 2 *Reference relationship* (X, Y) *holds for objects* X, Y *if* X *contains a hyperlink pointing at* Y.

The difference between dependencies and references is that dependencies show what is needed for an object itself to operate properly, while references show how a set of objects are interconnected to form hypermedia. These two relationships are called together as *portability relationships*.

3.2 Graphical notation and textual representation

The graphical notation for the above defined relationships may be appropriate for human understanding of portability conditions. The notation is explained through an example (Fig 3). This example shows the portability relationships of a simple searchable list (e.g. a hotel list) as a WWW document. Rectangles denote objects and ellipses denote features. References are drawn with dashed lines and dependencies are drawn with solid lines. In the rectangles the name and type of the objects are printed. The dashed ellipse shows the boundaries of the document.

The example document has a starting page containing a logo which is a shared object between several documents, and therefore it is not contained in this document. Searching is done by the search script, called from the home page. The search script requires version 4 of Perl, which is denoted here as a feature of the server environment for simplicity reasons. Another approach would be to denote the Perl program as an external object, and detail other Perl module dependencies as well. This depends on the definition of features. The search script reads the

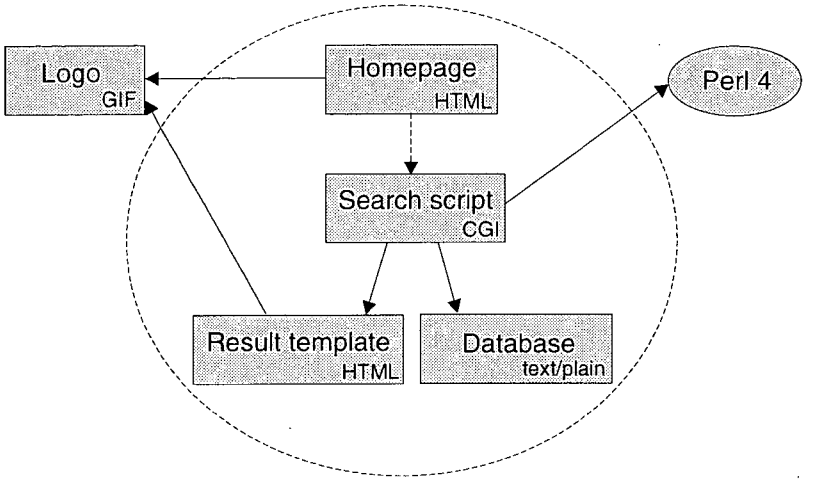

Figure 3: An example for the dependency/reference notation

```
<rdf:RDF>
<rdf:Description about="file:/www/cgi-bin/search_script">
  <PORT:type="CGI" />
  <PORT:dependsOn resource="file:/www/templates/search_result"
     PORT:mode="read" />
  <PORT:dependsOn resource="file:/www/support/index_file"
     PORT:mode="read" />
  <PORT:dependsOn resource="file:/bin/perl4"
     PORT:mode="execute" />
</rdf:Description>
...
</rdf:RDF>
```

Figure 4: RDF representation of dependencies for the search script in Fig 3

database file, and inserts the results of the search into the result template. The generated page contains the logo as well.

Some dependencies are not explicitly drawn on the figure. Each object of type X depends on the feature of handling type X by the server and user environment, but these dependencies are omitted to make the notation clearer. Type names in the boxes implicitly show these feature dependencies. Objects sharing the same dependency can be grouped together with a dashed ellipse to decrease the number of arrows in the graphical representation.

As Fig 3 shows, portability relationships can be interpreted as directed graphs as well. These graphs can be quite complex, and may contain thousands of nodes for a server environment. The graph need not be connected, and it may contain cycles. There are some rules for invalid connections according the type of nodes in the graph, for example an edge from an image to a program is invalid.

Portability relationships fall into the category of structural metadata, and can be described with RDF [17]. RDF (Resource Description Framework) is an effort led by the WWW Consortium to standardize the description of metadata, and coupling metadata with Internet objects. This gives us a machine readable representation of portability metadata applicable for automated document management. Part of the RDF description of the example in Fig 3 is shown in Fig 4.

3.3 Definition of portability requirements

The portability profile is the summary of all portability conditions for the document. It can be calculated by merging all portability relations from various metadata sources and automatic detection processes. For simplicity in further investigations dependency will mean either a dependency or a reference, as the correctness of references are also essential for the correct operation of the whole document.

Definition 3 *Formally a portability profile of document D is*

$$\mathcal{P}_D = \{\, (X,Y) \in \mathcal{P} \mid X \in D \vee \exists Z\colon Z \in D \wedge (Z,X) \in \bar{\mathcal{P}} \,\}$$

where $\mathcal{P}$ contains all described or detected dependencies in the server environment, and $\bar{\mathcal{P}}$ is the transitive closure of $\mathcal{P}$.

The portability profile can be further divided into three subsets:

Definition 4 *Internal dependencies are between objects inside the document:*

$$\mathcal{I}_D = \{\, (X,Y) \in \mathcal{P}_D \mid X \in D \wedge Y \in D \,\}$$

Definition 5 *External dependencies are between document objects and external objects:*

$$\mathcal{E}_D = \{\, (X,Y) \in \mathcal{P}_D \mid X \notin D \vee Y \notin D \,\}$$

Definition 6 *Viewer dependencies are requirements about the services of the user environment:*

$$\mathcal{V}_D = \{\, V \in \mathcal{V} \mid \exists X\colon (X,V) \in \mathcal{P}_D \,\}$$

where $\mathcal{V}$ contains all features that have specific user environment requirements.

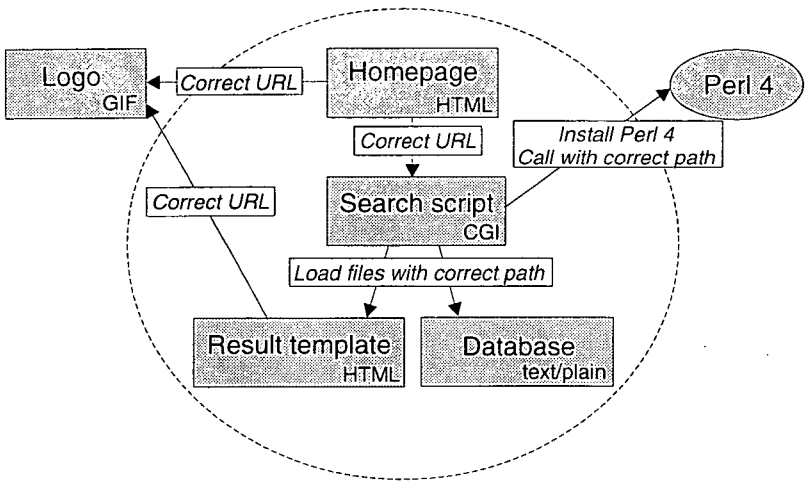

Figure 5: Restoring dependencies (example continued)

Moving/copying a document into a new server environment is formally a process to create correspondance between objects in the original server environment and objects in the new server environment. During this process corresponding objects are found in the new server environment, that are capable to play the same role for the document copy as their pairs in the original server environment.

Definition 7 *Suppose document D' in server environment S' is the copy of document D in server environment S. D' is a* complete mirror *of D if for every (X, Y) in $\mathcal{P}_D$ there is X' and Y' in S' that*

- *X corresponds to X',*
- *Y corresponds to Y',*
- *if X or Y is in D then also X' or Y' is in D'*

and then (X', Y') is in $\mathcal{P}_{D'}$.

However in practice making correspondance is not enough, each dependency has to be operational, has to be working for any user accessing the document. A dependency (X, Y) is *fulfilled* if object X is operational and accessible from object Y. To achieve this the dependency is *tested* and *restored.*

4 Solutions for creating portable documents

A brief list of necessary actions when moving our example document (Fig 3.) is given: the administrator should find the corresponding external objects for the logo

and Perl in the new environment. This could also include installation of Perl, or the decision to use version 5 of Perl to interpret the search script. In Fig 5. the required actions are shown to restore all dependencies. URLs referring to the logo and the search script has to be checked and corrected in the homepage, and in the result template as well. The search script needs two data files and a program file, each loaded by file paths which should be corrected, and execute permissions for the Perl program has to be set. It is obvious from this small example that moving a document means many hardly detectable and easily forgettable tasks for the administrator.

This process can be automated using portability relationships. The automated process contains the following steps:

1. The portability profile is calculated
2. The new place of the document is investigated
3. The document is moved/copied to the new place
4. Each dependency is tested and restored if necessary

Now each step is detailed. The calculation of portability profile needs to rely on portability conditions given by the author, because automatic detection of dependencies is not possible in all cases. It is very easy in an HTML object, but can be nearly impossible in a CGI program written in C where the source code is not available. Therefore the author's contribution is very important, and should be eased with software tools.

The new server environment where the document is to be placed has a set of offered features and documents. These are compared with the external dependency set of the document. The result is a list of missing features or external objects. The administrator of the server environment can install the required features (e.g. Perl 5.0), and match the required external objects to existing objects in the server (e.g. password files). When this is done copying of the document can start.

The document is copied to its new location, and each dependency is tried to be restored. There are three methods for restoration:

- *Changing the object source*: typically used for HTML files, where the URL referring to the depended object can be easily found and corrected in the source.
- *Changing configuration files*: typically used for features (e.g. CGI execution) to enable them in the server configuration.
- *Changing translation tables*: this is a new method and requires that the object accesses the depended object through a translation interface. The object looks for symbolic names in the translation table, which are translated to their path descriptors in the current server environment. Typically used for CGI scripts.

Finally viewer dependencies can provide ways to warn the user automatically about required/missing features in his environment. This can be a listing of required viewer features, or a help to install the missing features.

Generally there are three approaches that can help making WWW documents easily portable: guidelines for authors, document management tools, and application of middleware layers. None of these approaches is exclusive, rather they should be used in conjunction with the others.

4.1 Guidelines for authors

In the first place guidelines can be used to list non-portable technical solutions so that authors can avoid using them. There are also several simple solutions to create very easily portable documents. If relative URLs are used in HTML and VRML, there is no need to restore internal dependencies if the document is moved in one piece.

When using CGI scripts, external data files should be grouped together and referred to with a relative file path. If the script uses external objects, the script should read the location of these objects from a configuration file.

Applets should read their data using the Java built-in resource loading feature and the packaging mechanism.

4.2 Document management tools

The WebPack tool being implemented at SZTAKI is a prototype of a WWW document management tool. It has a notion of document, and maintains a central metadata repository for each document. The central metadata of the document may contain descriptive information about the whole document (e.g. authors, creation date, keywords), and may also contain administrative metadata including portability relations. Metadata can be edited through a graphical user interface, and dependencies in HTML files are automatically detected.

The administrator can move or copy documents inside the server with the WebPack tool. Further useful operations that can be provided by a WWW document management tool are: merging/splitting of documents, removing a document, installing a new document, filtering the contents of the document and verifying the operation of a document.

4.3 Application of middleware layers

A middleware layer could be used to provide standardized communication between the document and the server environment. In this way the immediate dependency of operating system and WWW server features can be lessened to indirect dependency. For example the document calls a software by its standard name, and the middleware layer directs the call to the software in the server environment. Middleware layers can be the solution for automatic configuration of server features as well.

5 Related work

There are several applications that partially support locally manageable WWW documents (e.g. Microsoft's FrontPage, Macromedia's Dreamweaver). These usually handle the web server as a whole and automatically adjust URLs in HTML files when it is needed. Netscape Composer's publishing functionality automatically updates the links when the page is published. However these solutions are restricted to HTML files, and lack a consistent approach to all possible portability problems on the Web.

There are two categories of public domain tools in connection with our topic: HTML integrity test tools, and mirroring tools. Mirroring tools replicate the files of a document on a remote server via HTTP or FTP protocol. However it is not always possible to retrieve all needed files in this way because of access and authentication problems, furthermore these programs do not explore and restore spoilt dependencies. For the latter HTML integrity test tools can be used. These traverse the hyperlinks in HTML files and report dangling URLs.

An organized standardization effort of IETF in this area is WebDAV (World Wide Web Distributed Authoring and Versioning) [18] which aims at "defining the HTTP extensions necessary to enable distributed web authoring tools to be broadly interoperable, while supporting user needs". In this respect WebDAV will support remote management and authoring of WWW pages, but currently it is only near the end of the standardization phase.

Location transparent naming is a central problem for portability. The CNRI handle system [10] and PURLs (Persistent URLs) [11] are two experiments for location transparent naming on the Internet. The problem with both systems is that they provide a flat naming scheme for objects, and thus will not be able to cope with the millions of existing WWW pages.

Object request brokers and distributed object management may give a long term solution for powerful and portable hypermedia documents. The Object Management Group's Object Management Architecture and Common Object Request Broker Architecture (CORBA [13]) will provide a very general framework for network objects including naming services. These network objects could encompass documents or parts of documents. However as a short and middle term solution the Internet community needs the tools outlined in Section 4.

6 Summary

The complex issues of portability for hypermedia documents on Internet were summarized, and a unified approach to handle portability problems was given. This approach contains a formal description technique for portability dependencies, and a method to match and restore these dependencies. Using this approach the exchange of complex WWW documents could become as simple as the exchange of documents in Microsoft's Word format. The WebPack framework serves as a testbed for the implementation of technical solutions based on the theoretical in-

vestigations.

Acknowledgment I would like to thank Róbert László his work in the implementation of the WebPack toolkit.

References

[1] About the World Wide Web, URL: http://www.w3.org/pub/WWW/WWW/

[2] WWW Names and Addresses, URIs, URLs, URNs, URL: http://www.w3.org/hypertext/ WWW/Addressing/Addressing.html

[3] Hypertext Markup Language, URL: http://www.w3.org/hypertext/ WWW/MarkUp/MarkUp.html

[4] Hypertext Transfer Protocol, URL: http://www.w3.org/hypertext/WWW/ Protocols/Overview.html

[5] Berners-Lee, T., Masinter, L., and M. McCahill, "Uniform Resource Locators (URL)", RFC 1738

[6] Rob McCool: The CGI Specification, URL: http://hoohoo.ncsa.uiuc.edu/ cgi/interface.html

[7] Java Technology Home page, URL: http://java.sun.com

[8] JavaScript resources, URL: http://developer.netscape.com/tech/javascript /resources.html

[9] VRML97 International Standard, URL: http://www.vrml.org/Specifications

[10] The CNRI Handle System, URL: http://www.handle.net/

[11] Persistent URLs, URL: http://purl.oclc.org/

[12] Common Ground Digital Paper, URL: http://www.hummingbird.com/cg/

[13] CORBA (Common Object Request Broker Architecture) URL: http://www. omg.org/corba/

[14] Reginald Ferber: Hypermedia and Metadata, 2nd DELOS Workshop, October 1996, Bad Honneff, Germany, URL: http://www.darmstadt.gmd.de/ ~ferber/delos/ws2/frame/frame.html

[15] Stuart Weibel, Jean Godby, Eric Miller and Ron Daniel: OCLC/NCSA Metadata Workshop Report, URL: http://www.oclc.org:5046/oclc/ research/conferences /metadata/dublin_core_report.html

[16] Carl Lagoze, Clifford A. Lynch, Ron Daniel Jr.: The Warwick Framework - A Container Architecture for Aggregating Sets of Metadata, Cornell Computer Science Technical Report TR95-1558

[17] Resource Description Framework, URL: http://www.w3.org/Metadata/rdf

[18] IETF WebDAV Working Group, URL: http://www.ics.uci.edu/~ejw/ authoring

[19] L. Kovács, A. Micsik: "Portable Hypermedia: a New Format for WWW Documents", SZTAKI Technical Report TR97-1

[20] L. Gulyás, L. Kovács, A. Micsik, L. Tersztenyák: Personalized Home Pages - A Working Environment on the World Wide Web, *to appear in:* IFIP'98 World Computer Congress, Vienna-Budapest, September 1998

[21] L. Kovács, A. Micsik: Replication within Distributed Digital Document Libraries. Proceedings of the 8th ERCIM Database Research Group Workshop on Database Issues and Infrastructure in Cooperative Information Systems, Trondheim, Norway, 1995

[22] L. Kovács, A. Micsik, G. Schermann: An Environment for Mirroring Hypermedia Documents, JENC 7, Budapest, May 13-16 1996.