# Digital document handling with WebPack

László Kovács, András Micsik

MTA SZTAKI
Computer and Automation Research Institute
of the Hungarian Academy of Sciences
Department of Distributed Systems
H-1111 Budapest XI. Lágymányosi u. 11. Hungary
{laszlo.kovacs, micsik}@sztaki.hu

**Abstract**

Current practice and shortcomings of handling digital hypermedia documents on the Internet are investigated. The WebPack format is defined to eliminate some of these shortcomings. The structure and operations of this new container format are discussed, and the use of WebPack in document management is explained.

**Keywords:** Internet, hypermedia, document-like objects (DLO), digital library, World Wide Web, metadata, Dublin Core

## 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 content 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. The meaning of document in case of the WWW is changing. WWW documents are sometimes more similar to a piece of software than to printed material. They may contain animations or may have an annotation facility, and what is most important they are linked together.

The Dublin Metadata Workshop [12] investigated this new kind of information source, and tried to set the floor for descriptive techniques for WWW documents which were termed Document-like Objects (DLOs) [11]. 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 parts are interconnected with links.
- it may contain active parts (e.g. scripts, applets) that respond to user interaction by executing programs embedded into the document

DLOs are called digital documents for simplicity in this paper. First the nature of digital documents is examined and their shortcomings are listed in Section 2. Next a new format called WebPack[1] is proposed (Section 3) for handling digital documents. A tool for management of the WebPack format is introduced in Section 4.

[1]In previous articles it is mentioned as Portable Hypermedia (PHM) format [22].

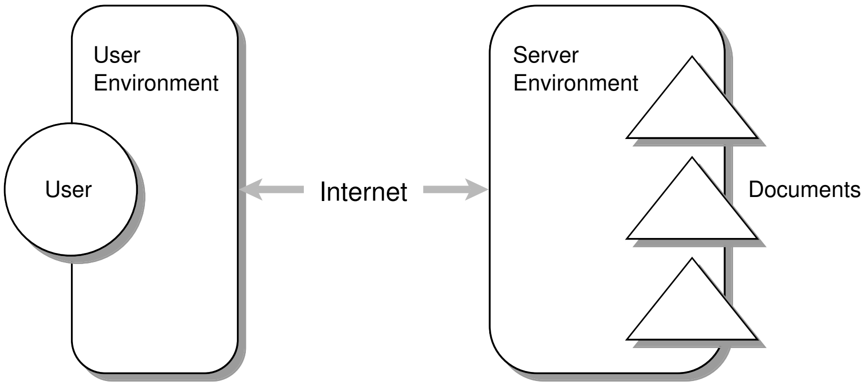


Figure 1: Accessing WWW documents

# 2 Digital documents: current usage and shortcomings

Some digital documents are just simple digital representations of printed material, offering the same content and layout as the printed version. On the other hand there exist digital documents hardly comparable to printed material, and some features of these documents, such as interactive or animated parts, make it impossible to reproduce them in print.

The most significant feature of digital documents is remote accessibility which gave momentum to various digital library efforts. The basic functionality of a digital library is to maintain a collection of digital documents and to make these documents accessible for its user community. Of course there are also more advanced services which a digital library can implement, but even this basic task introduced new problems, and some of those problems are still unsolved.

## 2.1 Serving and maintaining documents

The first area of problems is the consequence of the fact that digital documents may contain several files and may rely upon other external services, programs and files. The functionality of a document is therefore distributed over several files, directories and hosts, and this creates difficulties in serving, maintaining and preserving digital documents.

Figure 1 shows the basic components in a scenario when a user accesses a digital document. The *user environment* is specific to the actual user of the document, while the *server environment* is specific to the accessed document. The user environment provides browsing and viewing capability for the user. This usually includes an operating system, a WWW browser, additional viewers (e.g. Postscript, VRML), personal settings in the WWW browser and other non-standard features (e.g. JavaScript, plugins). The server environment offers the general document access services. It contains the operating system on the server side, a WWW server and several additional services inside or outside the Web server (e.g. access control, servlets, URL redirection or aliasing). A server environment usually serves several documents.

The *correct operation* of a digital document means that all of its functionality (hyperlinks, images, interactive parts, etc.) is 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 digital documents or external data as well. Those dependencies that can hold up the correct operation of a document may fall into three categories: hyperlinks, file dependencies and the so-called service dependencies.

### 2.1.1 Hyperlinks

Hyperlinks are the glue that hold parts of digital documents together, and also place documents into the globally linked network of the Internet. Hyperlinks are formulated as URIs [4], and most commonly as URLs, a subclass of URIs [5]. URL can refer to an object on the same host, or to an object on a different host, or to a part of an HTML page, or to 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 a local descriptor of the object, most usually a file path. 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. The URI schema defines a highly customizable reference methodology for the Internet which would allow location transparent naming facilities, but currently a widely used and location transparent naming facility for the Internet is missing.

A digital document has two inner structures: a link and a storage structure. The storage structure is a directory hierarchy where document parts are stored as files. This structure can be represented as an abstract tree without significant loss of generality.

The link structure can be represented as a directed graph where vertices are parts of the document and edges are hyperlinks between parts (potentially labelled with the anchor of the hyperlink). Hyperlinks in a document can be classified in two ways: first, as *external* or *internal* links, pointing outside or inside the document. Secondly, as *static* links stored statically in files, and *generated* links which are generated by an executable part during the access of the document.

The problem is that hyperlink targets are defined as paths in the storage structure. The storage structure of the document changes whenever the document is moved in the file system of the server. If the storage structure itself or the position of the storage structure changes in the file system, then the link structure has to be checked and corrected. This can be a fairly simple task in case of static links, and there are many freeware tools that detect and correct URLs in a set of HTML pages. In this case the only question that remains to answer is that if the URL is incorrect how the correct link target can be found. If the spoilt link is an external one, there is no universal method to find the target again (e.g. the server host name has changed). This is in connection with general document identification issues discussed in the next subsection. It has to be mentioned that more and more file formats start to use URLs (e.g. VRML, PDF), so the URL detecting algorithms has to be adapted to these new file formats as well.

URLs can also be generated by CGI scripts or Javascript. Detection of such generated links during a link integrity check is not always possible. CGI scripts can be written in various programming languages (C, Visual Basic, Perl, Python, etc.), and scripts may concatenate URLs from short text strings. One can imagine a very sophisticated software analysis tool that is able to identify how URLs are assembled in the code, but even that tool fails if the source code is not available.

### 2.1.2 File dependencies

The simplest example for file dependencies is inline images in HTML. As inline images are kept in separate files, they can easily be lost. A more complex example is a CGI script that requires different Perl modules. Perl modules are usually installed in a central location on the server, but this location may differ from server to server. So the CGI script may stop working if the required Perl module cannot be found.

Generally two major classes of file dependencies can be distinguished: dependencies on data files and dependencies on executables. In the first case, the location of the data file is needed only. In the second case not only the location of the executable is needed, but also proper access rights and version compatibility. Possible solutions for data file dependency problems are similar to solutions for hyperlink problems.

### 2.1.3 Service dependencies

Digital documents may utilize various services and features of both the user and server environments. Some examples: access restrictions (based on client host address or user authentication), enabling CGI scripts and setting the name of the so-called index files ("index.html", "default.htm", etc.) that are loaded when the URL references a directory instead of a file. These are supported by most of the WWW servers, but with different configuration syntax. If the WWW server changes in the server environment of the digital document, then these service dependencies may break the correct operation of the document.

A complex case of service dependencies is viewer configuration. In order to enjoy a document fully the user needs to have viewers for all formats used in that document. This needs some cooperation between the user and server environments, as the viewer on the user side is selected according to the MIME type of the document part, and the MIME type is determined by the WWW server (usually based on a translation table that maps filename extensions to MIME types).

### 2.1.4 Maintenance

If any of the above mentioned dependencies remains unfulfilled, the digital document is not fully operational. This can happen very easily as a consequence of changes in the operating system, changes in the WWW server, changes in auxiliary programs or when the document is moved to another server. Generally in the maintenance of a digital document collection the following tasks can be identified:

- Installation/removal of digital documents
- Reorganizing the collection (moving documents)
- Checking correct operation of digital documents
- Archiving documents
- Maintenance of catalogue

In case of simple document formats these are relatively easy tasks (e.g. single file Postscript format), and can be done by scripts as in the Dienst distributed digital library system [19]. Given the possible complexity of digital documents it can be seen that some of the above tasks can only be done by hand, and these tasks need a person with deep knowledge of the server environment [23].

### 2.1.5 Related efforts

Solutions may go into two directions; the first is to develop new ways and tools based on existing standards and usage to help performing these maintenance tasks. The second direction is to increase the intelligence of digital documents so that these maintenance tasks are performed by the document itself.

Distributed object management or agent technology could provide a solution in the second direction. Several agent frameworks has appeared in the last years, and some of those support mobile agents. A mobile agent could embed a digital document, and provide not only viewing

[24, 25] but also management services for its users. Another example in this direction is DigitalPaper [10]. It is a single file document format with enhanced portability options. Its goal is to ensure equivalent presentation of the document independent of hardware, operating system or installed fonts. DigitalPaper documents are created by printing the document from any application through a special print device driver. Documents can be viewed using a plugin with the two most popular WWW browsers, and there is also a stand-alone viewer application. DigitalPaper can incorporate hyperlinks, highlights, bookmarks and sticky notes. It has also built-in security features. The main drawback of this approach is that it applies a totally new document format which needs new software solutions for viewing and creation.

Taking the first direction we can see emerging commercial products (e.g. Microsoft Frontpage, Macromedia Dreamweaver) and freeware tools that ease some maintenance tasks in limited situations. With these tools one can move or copy HTML pages on a Web server, and the URLs in the pages will be automatically corrected. Other tools check all URLs on a Web server and report dangling links.

A 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. WebDAV is a very promising effort that will likely solve some of the problems mentioned in this section. It supports creating collections from HTML pages, and pages or collections may have properties (metadata) attached to them. Currently WebDAV is near the end of the standardization of its basic functionality, and it lacks software tools that implement its basic and advanced features.

#### 2.1.6 Benefits of WebPack

In our view the first important step would be to list and store all required functionalities for digital documents in a general way. This list of dependencies can serve as a checklist for librarians or administrators, but also it can be a base of building intelligent tools for document maintenance. As it can be concluded from our previous investigations in some cases it is not feasible to build intelligent tools for some tasks, for example to automatically install a new programming language on the server because a digital document needs it. Similarly the collection of information on document dependencies cannot be fully automated. Therefore the WebPack tool provides an easy-to-use visual environment to browse and edit the list of document dependencies, and in the meantime it is capable to automatically detect some of the dependencies. The natural place to store this dependency list is the metadata attached to the document. Unfortunately this kind of structural metadata has no recommended use up to now. More investigations on the use of metadata are given in the next subsection.

Finally, it is clear that the user environment is totally under the control of the user, and librarians cannot change or affect directly the user environment. Librarians or administrators, however, can inform the user about the required functionalities for viewing a document, and they can also give recommendations or detailed help for the user about the necessary modifications in his/her system. The list of document dependencies can be a base for providing this information for the user.

### 2.2 Searching and identification

The second area of problems is about the ways digital documents are found, identified or reused in the user community. If a user wants to find a certain document on the Internet, he/she has two possibilities: search using the specialized search services of digital libraries, or use general Internet search engines (e.g. Altavista, Infoseek). The first possibility means that he/she may have to visit the specialized search engines for each collection one by one, although these search engines may give a better quality result than general Internet search engines. For example the search engine of ETRDL (ERCIM Technical Reference Digital Library) [20] can search by keyword or by author. Digital library search engines also have a notion of document as a wanted entity, while general Internet search engines has only the notion of HTML pages.

If somebody uses a general Internet search engine to find a digital document, he covers the whole Internet in his search, but the result will be generally of low quality. These search engines cannot use proper bibliographic information for digital documents, i.e. they cannot tell the author or date of documents. Moreover they give a set of HTML pages as a query result, so it may occur that the result contains only one page somewhere inside the desired document in which case the user might overlook that single item as it may not be very significant for the whole document. Or the result may contain dozens of pages from the desired document which does not give a clear overview of the found documents.

There are further problems with the identification of a digital document. Currently there is no guarantee that if we find a document we will be able to find it again. The URL of the document can change, and the URL we bookmarked for that document will point to nowhere. It would also be useful for librarians and users to know the boundaries of digital documents, to know which pages belong to that document, and to know when they leave that document while browsing through hyperlinks. Identification could also mean that certain relations between documents are known (e.g. this is the Hungarian translation of that document, these documents are the same, etc.).

### 2.2.1 Related efforts

There is no general and widely used method for identification of digital documents except URLs [5]. As URLs identify merely locations on the Internet, and has nothing to do with content, they does not implement a location transparent naming method. Emerging solutions for location transparent naming are the Persistent URL Servers [9], and the CNRI Handle System [8]. However these efforts provide only a location transparent identifier for registered HTML pages, and they do not work with the notion of document either.

For a long time there was no standard way of compiling, formatting and attaching the bibliographic data to a digital document. The Dublin Metadata Workshop [12] defined the Dublin Core metadata set which is appropriate to hold bibliographic data. Later the Warwick Framework [14] defined packaging rules for metadata sets. Since then metadata issues has been constantly evolving [15]. There is a draft RFC on encoding Dublin Core metadata in HTML files. Several new metadata schemata have appeared to handle rating (PICS), distributed authoring (WebDAV), and digital signatures (DSig). The Resource Description Framework (RDF) [17] gathers these efforts into a general framework, but the emphasis is on assigning metadata to individual files instead of file collections. There is no general practice or recommendation on how to attach metadata to digital documents containing tens or hundreds of HTML pages.

New search engines are soon to appear on the Internet that will utilize Dublin Core Metadata attached to HTML pages (e.g. MetaWeb tools [16]). This will enhance the quality of search results only if large number of HTML pages will have attached metadata.

### 2.2.2 Benefits of WebPack

As the WebPack format aggregates HTML pages and other files into a digital document, it can be a natural target for persistent and location transparent naming. The WebPack format also supports semantical relations between documents. Boundaries of a WebPack container are clear, and it is always possible to find the title page of a document automatically if the

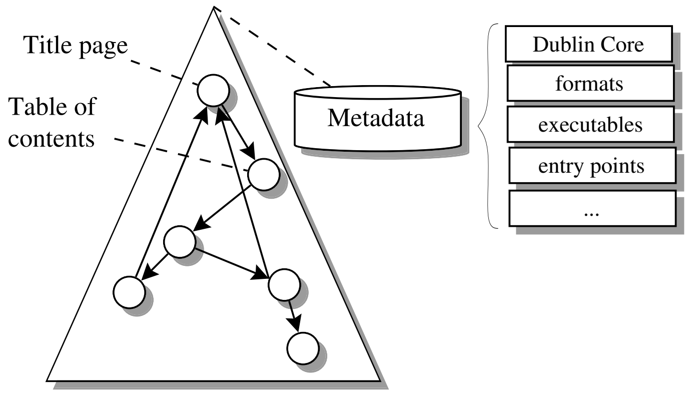


Figure 2: A WebPack container

URL of an internal page is given in the same document. Search engines could contract several result items belonging to the same WebPack into a single hit making the search result more compact and practical.

# 3 The WebPack format

The WebPack format is an effort to create a digital document format suitable for wide use on the present Internet. This is a container format for collecting the pieces of digital documents which are currently managed individually. Considering the current usage, and the large amount of widely used software on present-day Internet, the following requirements can be set for the WebPack format:

- URIs are used as existing syntax and semantics for hyperlinks,
- presentation formats widely used on the Internet can be integrated,
- supports portability and manageability in a wide range of servers and operating systems
- contains metadata for cataloging and management purposes,
- extensible

With these requirements present-day file formats and tools remain usable, while semantical relations within and between digital documents can be enhanced.

## 3.1 WebPack architecture

A WebPack is a container for the parts of the digital documents (Fig. 2). The container has an additional metadata repository to store various metainformation about the document. The following statements must be true for any WebPack:

- A WebPack is maintained in a way that present-day Web browsers can show its contents without the knowledge of the WebPack format.
- For any part its WebPack container can be determined unambiguously, even during a remote access.
- Files in the WebPack may be changed during WebPack management operations, but these changes cannot affect the correct operation of the WebPack.

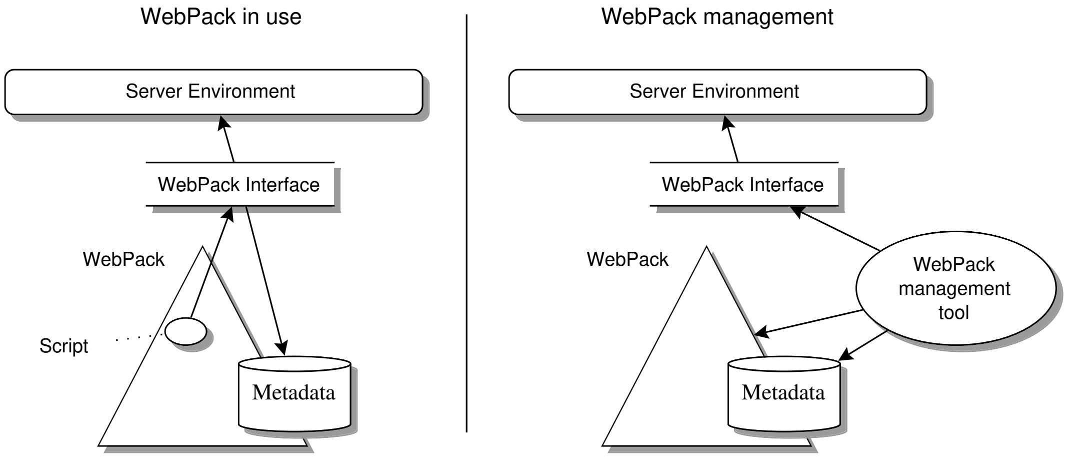


Figure 3: WebPack architecture

In the file system a WebPack container is represented as a directory subtree containing files belonging to the digital document. A WebPack can be viewed as a well-defined part of a WWW server supplemented with metainformation. The repository for metadata is placed into the root container directory inside a subdirectory with a predefined name.

The architecture contains a WebPack Interface which mediates management information between the server environment and the WebPack container and executes management commands (Fig. 3). The server environment includes the operating system, the WWW server and any other configurable options or software needed for the correct operation of digital documents. When the WebPack is in use, its parts can obtain appropriate information about the server environment via the WebPack Interface. An example is a script which needs access to a configuration file of the WWW server. Management is the other case when WebPack Interface is used. Management tools query all environment dependent information through the WebPack Interface.

The WebPack interface has two parts. A general part helps management tools to access the metadata stored in the WebPack container, and provides comfortable methods for file path translations. The other part provides methods to query different settings in the server environment. This part is based on a server and system dependent module which is selected and configured by the administrator of the site.

### 3.2 Metadata in WebPack

WebPack metadata is arranged into packages and stored in RDF format. Metadata is divided into two parts. One part is meant to help cataloging, identification and searching. Dublin Core acts as the basis of this collection of metadata. Copyright can be added here as well. The second part is the management/technical description, for which own metadata packages are used. This contains the description of:

**Information on formats used in the WebPack:** This includes the mapping of file extensions to MIME types, and additional format-dependent information (format subtypes, character sets, etc.).

**File dependencies:** This is the list of all file dependencies for the files in the document, containing external and internal hyperlinks, and other necessary files.

**Information on active parts:** These are the parts of the WebPack that are executed on the server side (e.g. CGI scripts, Java servlets) or on the user side (e.g. Java applets). Information stored here contains dependencies on other programs or program versions.

**Entry points:** Digital documents has distinguished pages which are essential for navigation or simply very often used. It is very likely that these are the pages where hyperlinks are pointing from other places in the Internet, therefore these are called document entry points. Some examples for typical entry points: the homepage or title page, table of contents, index, or search page.

**Other characteristics of files:** This may include for example access restrictions, language options, etc.

**Relations to other WebPacks:** Possible relationships are surveyed in the next subsection.

### 3.3 Relations between WebPacks

Relations between WebPacks can be helpful in many ways for the document user and the document maintainer as well. Because of the semantical meaning most of these relationships cannot be detected automatically, rather they are declared by the author or maintainer. Some examples for WebPack relations are detailed here.

**Alternative relationship** A WebPack may have alternatives that contain the same information presented differently in language, in formats or in links. For example an English language HTML document may have a German alternative or an alternative in PDF or an alternative which keeps sections in one HTML file, not in separate HTML files.

**Equivalence relationship** WebPacks are equivalent if they appear/work identically for each user (in a semantical way), though they may have different file names or different formats. For example if all GIF images in one document are replaced with equivalent PNG images in the other (and there are no other differences), then those documents are equivalent.

**Master/Replica relationship** A digital document is often replicated to different servers to enhance its availability. In this case there is a master (original) version of the document which is periodically copied to the replica sites.

## 4 The WebPack tool

As a part of the prototype for the WebPack architecture, the WebPack tool is being implemented at SZTAKI. It is written entirely in Java, and offers a graphical user interface for the management of WebPacks. After opening a WebPack container the tool shows the contents of the attached metadata packages (Fig. 4). Metadata can be easily changed, so this tool is both appropriate for the author of the document to enter metadata about the document, and for the maintainer of the document to browse attached metadata.

It is very easy to wrap an existing digital document into a WebPack with this tool. The user defines the directory where files of the document reside and the possible entry points. After this the tool automatically explores the document and creates initial metadata for the WebPack container which the user can later refine.

The menu offers some basic operations on the WebPack (e.g. move, copy, verify). In case of moving the WebPack to another location the tool not only moves the files of the document to their new locations but also tries to adjust hyperlinks and other settings according to the knowledge in the metadata repository.

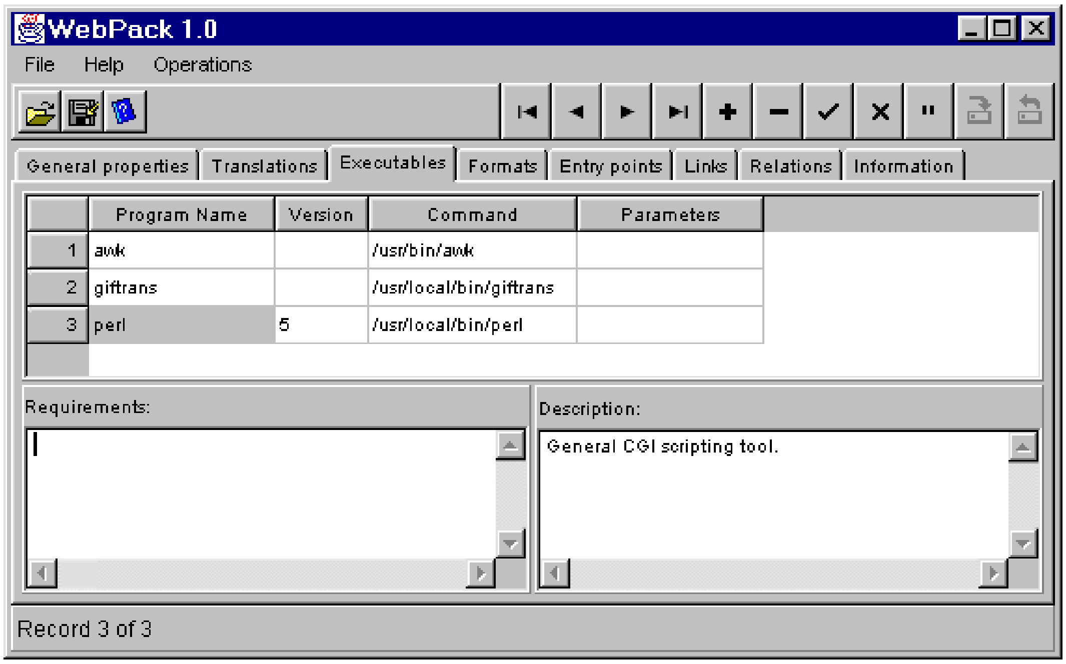


Figure 4: WebPack tool

Some immediate uses of WebPack include Web server maintenance and mirroring. A WWW server can be logically split into WebPack containers, creating a modular document space which is manageable by the WebPack tool. Documents are often mirrored (replicated) to several locations. This task can be automated based on the WebPack architecture and its master/replica relationship. Several scenarios for intelligent mirroring are given in [26]. A prototype of a simple mirroring tool based on WebPack can be found in [27].

In the future we would like to merge the WebPack architecture with WebDAV, and apply the powerful primitives of WebDAV to enhance the functionality of WebPack.

## 5 Summary

The WebPack format and architecture were proposed to reduce the problems with handling Document Like Objects on the Internet. WebPack is a container format, and does not obsolete current Internet usage and file formats, but enhances manageability and portability aspects. This approach can serve as a middle-term solution in the trend of making network information services more and more intelligent based on object and agent technology.

**Acknowledgment** We would like to thank Róbert László for 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] Hypertext Markup Language,
URL: http://www.w3.org/hypertext/WWW/MarkUp/MarkUp.html

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

[4] WWW Names and Addresses, URIs, URLs, URNs,
URL: http://www.w3.org/hypertext/WWW/Addressing/Addressing.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] The Virtual Reality Modeling Language Specification, Version 2.0,
URL: http://vrml.sgi.com/moving-worlds/index.html

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

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

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

[11] 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

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

[13] Dublin Core Metadata Initiative, URL: http://www.purl.org/DC

[14] 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

[15] Metadata activity at the World Wide Web Consortium,
URL: http://www.w3.org/Metadata/

[16] MetaWeb Project, URL: http://www.dstc.edu.au/RDU/MetaWeb/

[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] C. Lagoze, J. R. Davis: "Dienst: an Architecture for Distributed Document Libraries", Communications of the ACM, 38 (4) April 1995

[20] S. Biagioni, J. Borbinha, R. Ferber, P. Hansen, S. Kapidakis, L. Kovács, F. Roos, A. M. Vercoustre: "The ERCIM Technical Reference Digital Library", Second European Conference on Digital Libraries (ECDL'98), Heraklion, Greece, September 1998, Lecture Notes in Computer Science 1513. Springer 1998

[21] L. Kovács: "Discovery of Resources within a Distributed Library System", Communications of the ACM, Vol. 41. No. 4. April 1998

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

[23] A. Micsik: A study of portability in the deployment of WWW, *to appear in:* Acta Cybernetica Vol. 14. (1999) No. 2.

[24] L. Gulyás, L. Kovács, A. Micsik, L. Tersztenyák: Personalized Home Pages - A Working Environment on the World Wide Web, Telecooperation Proc. of the XV. IFIP World Computer Congress 1998, Vienna-Budapest

[25] L. Gulyás, L. Kovács, A. Micsik, L. Tersztenyák: Agent Based Internet (WWW) Services, SZTAKI Technical Report TR97-2, March 1997

[26] 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

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