Journal of Universal Computer Science, vol. 11, no. 6 (2005), 926-943
submitted: 1/6/05, accepted: 25/6/05, appeared: 28/6/05 © J.UCS

Fine-Grained Transclusions of Multimedia Documents in HTML

Josef Kolbitsch
(Graz University of Technology, Austria
josef.kolbitsch@tugraz.at)

Abstract: Transclusions are a technique for virtually including existing content into new documents by reference to the original documents rather than by copying. In principle, transclusions are used in HTML for the inclusion of entire text documents, images, movies and similar media. The HTML specification only takes transclusions of entire documents into account, though. Hence it is not possible, for instance, to include a part of an existing image into an HTML document.
In this paper, fine-grained transclusion of multimedia documents on the Web are proposed, which presents a logical realisation of the concept of transclusions in HTML. The proposal makes it possible, for instance, to include sections of existing images or small portions of entire movies into HTML documents.
Two different approaches to implementing the functionality presented are detailed. The first architecture is based on a transparent extension module to conventional HTTP servers, whereas the alternative design makes use of a CGI program. Both approaches are fully self-contained, reside on an HTTP server and do not require browser plug-ins or any other special software components to be installed on client computers. An amendment to the HTTP specification is not required either. A prototype implementation demonstrates the proposal for a number of document types.

Keywords: hypermedia, transclusions, Xanalogical Structure, authoring systems, publishing systems, web-based applications, multimedia

Categories: H.1, H.3, H.4

1 Introduction

In 1965 Ted Nelson presented *"a file structure for the complex, the changing and the indeterminate"* at the ACM Twentieth National Conference (see [Nelson 1965]). In this publication he introduced the term hypertext and a technique called *trans-clusions.* Transclusions are the fundamental concept that allow authors to virtually include portions of existing documents into new articles without the need to duplicate them. From a technical perspective, transclusions are references: the transclusion t_B in document A is a reference to a portion of the content of a document B that is virtuall included into document A.

1.1 Origin of Transclusions

Nelson regards transclusions as complete replacement for all *cut-and-paste* mechanisms employed. He claims that the widely used cut-and-paste operation is not what people actually intend to do. It is a restriction imposed upon authors by the

nature of paper. Writers actually do not want to duplicate an existing document, cut out the section they want to quote, and paste it in their document. They want to include, i.e., transclude, the *original* content and let readers know what the source and context of the quotation are (e.g., [Nelson 1981]).

Another rather pragmatic solution that becomes necessary because of the nature of paper are reference lists at the end of scientific publications. They are usually not what is desired by writers and wanted by readers. They are an attempt to preserve the context of the quotation that is lost by copying-and-pasting a portion of printed material.

The physical restrictions of paper were embraced by most computing systems in an attempt to resemble office environments and the common processes in work environments (cf., [Yocom 2004]). Therefore most current operating systems with graphical user interfaces do not make use of the strengths of hypertext. They utilise metaphors such as a desktop, folders and documents, where a document has to be put in exactly one folder. There is a clipboard, and content from a different document is included using copy-and-paste mechanisms (see [Nelson 1996]).

1.2 Ramifications of Transclusions

Transclusion technology is not only a replacement for copy-and-paste. With transclusions, the original context of a quotation is preserved and a visible link to the source of the transclusion can be provided. Ted Nelson's approach to implementing this functionality involves *transpointing windows* (e.g., [Nelson 1995]).

Furthermore, document authors can receive notifications whenever their articles are transcluded by other users. Thus they can, for instance, be informed about other researchers in the same area. Writers using transclusions, on the other hand, can be notified automatically about changes in source documents of the transcluded content (see [Krottmaier 2002]).

The nature of transclusions has an impact on the storage space required. Since content is not duplicated when it is quoted, disk space can be saved. This was a major issue in the 1960s when Nelson published his ideas and storage space was still precious. Although nowadays disk space is available at low cost, it can become a concern when handling large amounts of multimedia data.

Version control systems are another area where storage space can pose a problem (e.g., [Tichy 1985]). When n versions of a document exist, and every version retained in the system contains the entire content in a file, the space for storing the content can increase rapidly. Therefore it can be more effective to keep only the changes made from version $n-1$ to version n. When version n of a document is retrieved, content from all previous versions is transcluded into the current version.

Aside from obvious improvements in authoring and publishing systems, transclusions are able to provide a solution to copyright issues experienced on the world wide web. Authors include content into their documents by means of transclusions instead of copying and pasting. Whenever a reader views a transclusion information on the permissions and rights associated with the transcluded content is added. In addition to this, an automatic micropayment can be made to the corresponding content owner ([Nelson 1999]). Nelson coins this paradigm *transcopyright* (see [Nelson 1998]).

1.3 Implementation of Transclusions

Ted Nelson attempts to realise his notion of hypertext in a system named Xanadu (e.g., [Nelson 1981]). Transclusions are a fundamental part of this system. Their implementation, however, relies on a document model that is radically different from most paradigms used today. In Xanadu, documents do not contain any content but only references to the actual content. The content is retained in dedicated content repositories. It is indexed and referenced at the level of single characters—the finest granularity possible.

Documents in Xanadu are lists of references to content stored in the system, e.g., a document might consist of *"characters 120 to 843 and characters 11,196 to 13,101 from the repository"*. In such a system transclusions can be implemented easily: transcluding content from document B into document A corresponds to adding the references to the actual content in the repositories to the reference list of document A.

Thus, both creating and retrieving transclusions in Xanadu are trivial list operations. Functions that handle situations in which large portions of documents are modified or deleted can also be implemented (e.g., [Nelson 1999]). They require content to be persistent, i.e., documents in Xanadu cannot be deleted. When a document transcludes a portion of content that has been changed or removed in version *n* of the source document the transclusion can be retrieved from version *n-1* of the document. In this case, the user has to be notified that a more recent version of the source document is available. Alternatively, the system can attempt to determine which parts of the source document have been modified and which portions in version *n* correspond to the content in version *n-1*. In this case, the user has to decide whether the new content transcluded is still appropriate. It can assumed, though, that the context of an existing document remains relatively unchanged. Thus, both sense and connotation of an existing document remain relatively unchanged (e.g., [Nelson 1981]).

In contrast to Xanadu, most other system that allow the use of transclusions (e.g., HTML, see section 1.4) do not require content to be persistent. Therefore these environments are not able to handle situations in which content becomes unavailable accordingly. When users transclude an image into their documents and the original image is deleted or extensively modified they are usually not notified. This is a factor that deters many authors from transclude remote resources.

1.4 Transclusions in HTML

The Hypertext Markup Language (HTML, [HTML 1999]) used on the world wide web is a relatively simple language for describing hypertext pages. The focus of the early development was on simplicity, style and graphical presentation features rather than on functionality and modern technologies. Hence innovative ideas such as bidirectional hyperlinks were not taken into consideration (e.g., [Maurer 1996; Maurer and Lennon 1996; Kappe 1995]). Potential difficulties already known before the invention of HTML including the distinction between URLs and URIs were not handled either (e.g., [Pam 1995]).

In various aspects, HTML makes use of transclusions. Markups such as `<iframe>`, `<img>`, `<object>` or `<embed>` virtually include content such as entire HTML files, images, video clips and animations into HTML pages by means of

linking. Transclusive functionality in HTML is very limited, though. The transclusion mechanisms available can only be applied to entire documents; fine-grained transclusions such as a small spatial selection of an image are not implemented. This is most likely due to concessions made in favour of ease of implementation, in order to reduce the computing power required and to increase performance on the web.

Therefore fine-grained transclusions of multimedia documents as a logic extension of to the infrastructure used in HTML-based environments are proposed. This makes it possible, for example, to include only a part of an existing sound file into a web page. The proposed functionality is an addition to a previous work that allows users to make textual transclusions in HTML (see [Kolbitsch and Maurer 2005b]). By combining these two projects, it becomes possible to transclude text documents and a wide range of multimedia documents in conventional HTML pages at the finest level of granularity and without the need to install software on client computers.

2 Fine-Grained Transclusions of Multimedia Documents

The concept of fine-grained transclusions can be applied to numerous content types. Therefore, this section briefly explains a variety of multimedia document types, and their use in connection with transclusions is addressed.

The syntax used in all examples resembles the syntax for conventional HTTP GET requests and the common HTML syntax, for instance, of `<area>` markups.

2.1 Drawings, Vector Graphics

Drawings and vector graphics are used very often in the technical domain and when abstract concepts are graphically depicted. Examples are CAD drawings such as plans of buildings, cars and machinery in general; both two- and three-dimensional models of objects such as molecules in chemistry; flowchart diagrams and organisational charts. A common property of most vector graphic formats including SVG (scalable vector graphics, [SVG 2003]) is that every vector and every other object is stored in way that it can be addressed independently. This means that even after saving a document, a line can still be selected as a line, and its length or position can be modified.

The same is also true of some formats that are employed to describe three-dimensional models or "virtual reality" scenes. The recent X3D format, for instance, stores every object separately in the file, and it is possible to modify every object individually (see [X3D 2005]).

These characteristics can be made use of when creating transclusions. Fine-grained transclusions of vector graphics can be based on spatial selections or on object selections. Thus when a transclusion is created, the user can select either certain objects or a spatial region of the drawing. The component generating the actual transclusion (see below) has to select the given objects from a drawing and subsequently interpret, i.e., render, the resulting data. In case of a spatial selection, the vector graphic has to be interpreted first (window-viewport transformation), and only then a selection can be made.

Object-based selection is most likely only reasonable when a small number of objects have to be dealt with. Real-world models often contain millions of polygons, which makes selecting groups of objects impracticable. In such a scenario, spatial region selction seems to be more appropriate. The first example in Table 1 describes the transclusion of a rectangular shape 170 points wide and 20 points high, starting at 10 points from the left top corner.

Doc.Type	**Selection**	**URL**
Drawing	rectangular shape	`image.svg?shape=rect&coords=10pt,10pt,180pt,30pt`
Image	rectangular shape	`image.jpeg?shape=rect&coords=10,20,210,220`
	arbitrary shape	`image.jpeg?shape=poly&coords=10,10,30,30,50,10,50,90,30,50,10,90`
Video	temporal region	`movie.mpeg?start=3m42s&length=4m59s`
	spatio-temporal region	`movie.mpeg?shape=rect&coords=0,20,320,220&start=0m0s&end=47m12s`
Sound	start/end positions	`song.aiff?start=0m0s&end=1m13s`
	relative length, time	`song.aiff?start=0m0s&length=1m13s`
	relative lenght, frames	`song.aiff?start=0m0s&length=1825f`
	index points	`song.aiff?start=index1&end=index3`

Table 1: Example of selections in different multimedia document types.

2.2 Photos and Images

Photos and rasterised images are particularly wide spread on the world wide web. They are usually produced by conventional photo cameras, specialized cameras such as infrared cameras, and other imaging devices such as ultrasound and x-ray detectors or radar units. Diverse areas such as medical imaging, satellite imaging, microscopy, (print) publishing make use of rasterised images. Some technologies for the description of three-dimensional models and virtual reality scenes such as Apple's Quicktime VR utilise rasterised information as well (e.g., [QTVR 2005]). A series of images are stored and displayed in a particular way so that users have the impression that they are viewing a three-dimensional object. (The "object" is, however, merely a recombination of two-dimensional images from different perspectives to a new two-dimensional image.)

For rasterised images it is usually not possible to select separate objects, only pixels or regions of pixels can be addressed. A more general technique is the usage of normalised device co-ordinates instead of pixels (NDC; e.g., [Foley et al. 1997]). With NDC, the left-bottom corner of the image is described with the co-ordinate pair (0,0) and the top-right corner corresponds to (1,1). This method makes references independent, for example, from actual devices and implementations.

A straightforward approach for creating fine-grained transclusions of images and photos are selections based on regions. When users want to make a fine-grained transclusion of an image they can mark a certain (spatial) region of the image, where regions can be rectangles, other geometric objects and arbitrary shapes defined with polygons or Bézier curves.

The examples in Table 1 list two types of spatial selections: the fist one is a simple rectangular selection 200 pixels wide and 200 pixels high, starting at coordinates (10, 20). The second example is an arbitrary polygon consisting of six co-ordinate pairs.

2.3 Video Content

Most digital video content is represented as a series of frames, where each frame is a rasterised image. Therefore basically all methods that can be utilised with images and photos are also applicable to video content, e.g., spatial selections using areas of pixels or normalised device coordinates. In addition to this, two further region selection mechanisms can be identified:

- temporal selection and
- spatio-temporal selection.

When using a temporal selection, the user specifies a certain period of time on the time line of a video clip in order to denote that a temporal region of a movie is to be transcluded. The combination of a spatial and a temporal selection leads to a spatio-temporal selection. This means that a fixed area in one frame of a video clip is selected also in a number of consecutive frames. This method is useful, for instance, when only a certain area and a certain section filmed by a surveillance camera are relevant.

Table 1 lists two examples for temporal and spatio-temporal selections. The first video file is a temporal selection with a length of 4 minutes and 59 seconds, starting at 3 minutes and 42 seconds. In the second example, an area 320 pixels wide and 200 pixel high is cropped, and a temporal selection with a length of 47 minutes and 12 seconds, starting at the beginning of the video file, is made.

2.4 Sound and Music

A number of different approaches for describing sound and musical data exist. Most technologies available to date such as AIFF (see [AIFF 1989]) are frame-based. All current audio formats include some sort of timecode, and some of them support the use of index points.

Therefore temporal region selection is the apparent method for providing fine-grained transclusions of sound files. Users can define a start position and either the length of the content to be transcluded or the end position. As illustrated in Table 1, positions and length can be given as absolute time codes (e.g., `1m13s`), as index points (e.g., `index1`) or as number of frames where applicable (e.g., `1825f`).

2.5 Compound Multimedia Documents

Multimedia animations are much more complex than the media types introduced above. Although they can be seen as compound documents they can usually not be

treated as composites of basic media types. One reason is that the most common documents formats such as Macromedia Flash are more or less proprietary (e.g., [Macromedia 2005]). This means that even if a video clip within a multimedia animation is represented as an individual object in the document format, it might not be possible to access it separately or to extract it. Thus, even relatively simple operations such as cropping the area to be displayed might be very hard to implement.

Despite these difficulties and in order to underline the fundamental concept, it should be mentioned that, at least in theory, it is possible to make fine-grained transclusions of compound multimedia documents in HTML. Examples are:

- spatial, temporal or spatio-temporal selections of entire animations;
- selections of single objects or a group of objects within complex animations; or
- spatial, temporal or spatio-temporal selections of individual objects or a group of objects of complex animations.

An actual implementation, however, would most probably require active support from the companies maintaining the respective document format.

3 Approaches to an Implementation

Fine-grained transclusions are not only an abstract concept in hypermedia theory but there are several potential application areas that are particularly well suited for the proposed functionality. Therefore two approaches to an implementation are introduced, the design goals are detailed, and a prototype implementation that can be accessed over the Internet is presented.

3.1 Design Goals and Requirements

The idea of fine-grained transclusions attempts to foster the reuse of information that is already available on the world wide web. A wide range of differing media formats should be supported, and their characteristic features should be made use of. Therefore a number of design goals have to be considered:

- use of any document on the Web: it should be possible to make fine-grained transclusions of any media document accessible on the Web—not only of files from a closed repository;
- extensibility: a system for creating and retrieving fine-grained transclusions should be extensible, i.e., adding support for new document types should be taken into consideration;
- flexibility: the system should make use of the features offered by the various media types, e.g., the types of selections (spatial region, object selection, etc.) should be adapted to the document and encoding types;
- use of existing standards: the definition of fine-grained transclusions should not require the introduction of new HTML markups;
- reuse of existing infrastructure: it should be possible to reuse existing infrastructure including browser plug-ins. Moreover a seamless integration into existing frameworks must be provided, i.e., the use of fine-grained

transclusions of multimedia documents must not break existing plug-ins, etc.;

- no additional software: users should not have to install additional software on their client computers to be able to make use of the proposed functionality;
- transparency: an implementation should be transparent to both web browsers and users;
- ease of use: for authors, it should be easy to create fine-grained transclusions of media documents.

An implementation can be based on plain HTML, without any additional markups, and plain HTTP, without the need to alter the communication protocol used between client and server. The basic requirement is a conventional HTTP server that can be extended with either external extension modules or CGI programs. Most current web servers such as the popular Apache HTTP server incorporate both capabilities (e.g., [Apache 2005]).

3.2 System Architecture

Two different implementations are proposed: an implementation that involves an extension module for the HTTP server, and a CGI-based approach (see sections 3.3 and 3.4). Independent of the actual approach used, the system follows a particular architecture that is described in the following paragraphs.

A fine-grained transclusion is defined by a number of additional parameters that describe which portion of the content is to be extracted. They are appended, for instance, to the `src` attribute of the `<img>` tag in an HTML document (see Table 1 for a number of examples). The parameters defining the fine-grained transclusion can be provided "manually" by the author, can be generated using dedicated authoring tools or a simple web interface.

When a web page containing fine-grained transclusions is requested, a component on the server (extension module or CGI program) analyses the definition of the fine-grained transclusion and retrieves the original document. Subsequently, the specified portion of content is extracted from the original document using an appropriate plug-in module for the corresponding document type, and is sent to the client (see Figues 1 and 2 and section 3.5).

Thus, the implementation follows a client-server approach, where the complexity and logic lie within the server-side component.

The major benefit of this approach is that it is completely transparent to HTTP clients and the current infrastructure for playing back sound and videos already present in most web browsers can be reused. No additional software components need to be installed on client computers. A video clip, for instance, can still be included into a web page by means of the `<object>` markup. Only the URL given in the corresponding tag is a altered: it has to define the piece of media to be transcluded.

Another advantage is that the content does not have to be stored in any particular database or special repository. Content can be stored as files in directories of a conventional file system, but also in a remote file system, in databases, etc. The architecture presented also avoids unnecessary network usage by only transferring the data actually requested from the server to the client.

This generic structure of the system is the foundation of the implementations of the server-side components detailed in the following two sections.

3.3 Extension Module for HTTP Servers

The functionality of many HTTP servers including the widely-used Apache web server can be extended by means of modules (see [Apache 2005] and [Thau 2003]). Modules can perform various tasks from authentication to enabling access to databases and handling various error conditions. Frequently employed modules include `mod_perl`, a module that implements an interpreter for the Perl programming language, and `mod_php`, an extension for using the PHP scripting language.

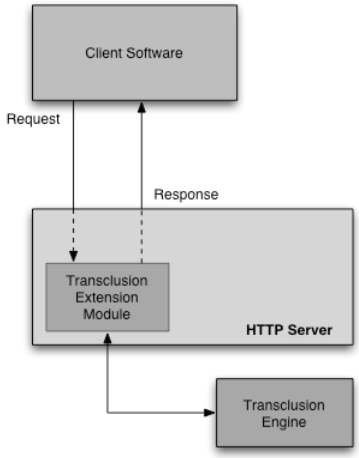

Figure 1: Architecture of an approach, where the functionality of the HTTP server is extended with an external module.

This implementation of fine-grained transclusions relies on an extension module that is invoked whenever data is requested from the server (see Figure 1). It analyses the request, and if multimedia content is requested, the module checks whether the entire file is requested or only a part thereof, i.e., if a fine-grained transclusion is to be made. If so, the extension module checks if a plug-in for the corresponding document type exists (see section 3.5) and generates the requested portion of content. Finally, the content selected from the entire multimedia document is returned to the HTTP client.

If the HTTP request does not refer to a multimedia document, the request is passed on to the HTTP server. If a plug-in for a given media type is not present or the transclusion cannot be made, an error message is returned to the client.

This approach is completely transparent to both HTTP clients and users. Only the parameters defining the portion of content to be transcluded can be "seen" by users; otherwise a fine-grained transclusion can, for instance, not be distinguished from a

conventional inline image. The disadvantage of this approach is that an extension module has to be installed on the HTTP server. In environments where the configuration of the HTTP server cannot or must not be changed, this implementation is not favourable.

3.4 CGI Program

Similar results can be achieved by using a CGI program. When the CGI-based approach is utilised, every fine-grained transclusion has to be loaded through a particular CGI program (see Figure 2). The CGI program carries out the same operations as the HTTP server extension module: it checks if the request is valid, finds an appropriate plug-in, extracts the requested portion of content and sends it to the client.

The difference becomes obvious when looking at the structure of an HTML tag referring to a fine-grained transclusion. In Listing 1 an example is illustrated: The first markup is a traditional `<img>` tag. The second markup defines a fine-grained transclusion that is loaded through an HTTP server extension module (cf., Table 1). It is transparent to users. The third markup is the same fine-grained transclusion loaded through a CGI program. Although it is transparent to web browsers—they do not "recognise" that they do not load the original image—it is not transparent to users because they can see that an intermediate component is used when the image is retrieved.

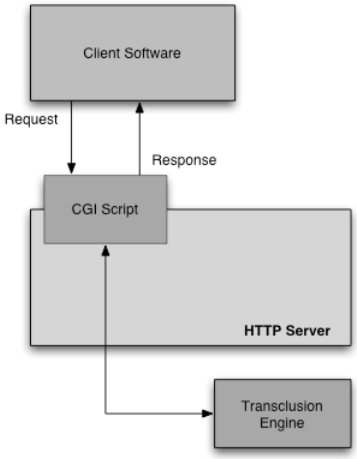

Figure 2: System architecture of a CGI-based approach. The client sends a request to a CGI script that, in turn, forwards the request to the transclusion engine.

Since CGI programs can almost always be added, even if extension modules cannot be installed, this approach can be advantageous when the setup of an HTTP server cannot be modified. Moreover the CGI program can be implemented in a way that not only local documents can be transcluded but also files from remote URLs

(see section 3.6). So even users that do not have the CGI program installed on their servers could make use of fine-grained transclusions.

3.5 Plug-In Architecture

Both the extension module to HTTP servers and the CGI programs are mere frameworks that offer the functionality needed to handle the requests of clients. The requested portion of content is generated by plug-in modules.

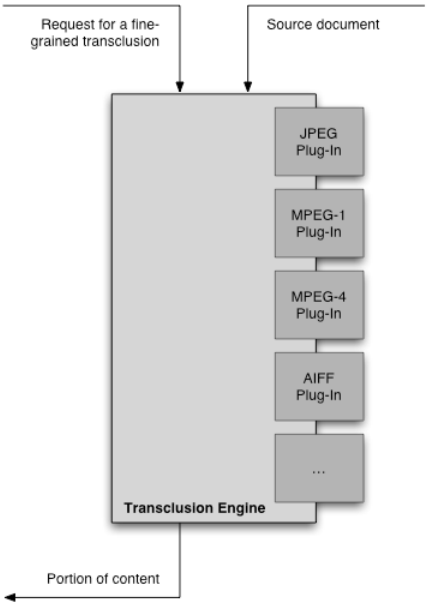

Figure 3: Basic structure of the plug-in architecture. The "transclusion engine" receives a request, fetches the given source document and processes it with an appropriate plug-in module. The extracted content is returned.

When a client retrieves a fine-grained transclusion of a certain document the HTTP server extension module or the CGI program analyse the request. If the requested document is a multimedia document and the request is valid, an appropriate plug-in that can handle the given document type (e.g., a GIF image) is searched for. If a plug-in can be found the original document is retrieved and passed on to the corresponding plug-in along with the parameters describing the content to be selected. The plug-in extracts the demanded portion of content and returns it to the HTTP server extension module or the CGI program. The basic structure of the plug-in architecture is depicted in Figure 3; the retrieval strategy with plug-ins is shown in Figure 4.

Although this approach may seem overly complex, it has various advantages. Every file type can be realised as separate plug-in. Therefore it is fairly easy to offer support fort new document types or encoding standards. The approach is flexible in

that the plug-ins can take the particularities of certain media types into account and deal with their peculiarities.

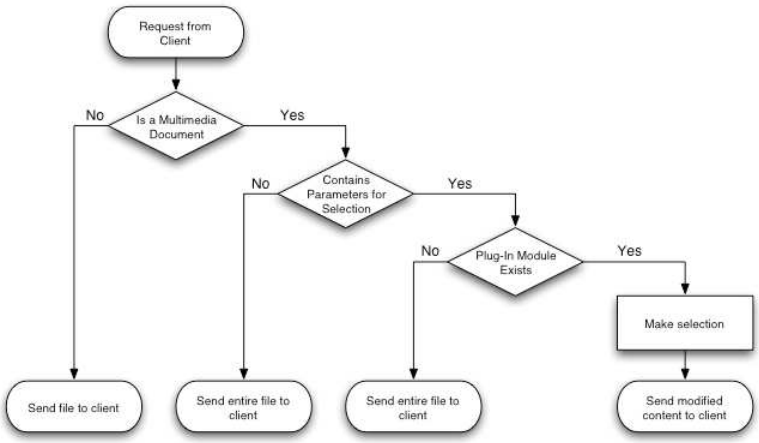

Figure 4: Retrieval strategy for multimedia documents with the ability to make fine-grained transclusions.

The implementation of a plug-in is relatively uncomplicated for simple, open document formats such as JPEG/JFIF and PNG images or MP3 sound files. The prototype illustrated in section 3.6, for instance, implements a plug-in module for JPEG, PNG and GIF images. The implementation is somewhat more complex, though, when media types such as MPEG-1 are to be handled: the encoding is frame-based, where frames only contain the segments different from a particular base-frame. So when a temporal selection of an MPEG-1 video file starts at an arbitrary frame, it might be necessary to reconstruct the entire frame first. An implementation becomes even more difficult when proprietary document formats such as Microsoft's WMV video files or complex media files such as animations based on Macromedia Flash are to be handled.

3.6 Prototype Implementation

The prototype implemented follows the CGI-based approached introduced above. It consists of two components: a script for generating a URL that can be used to include the fine-grained transclusion into an HTML document, and a second CGI program that actually produces the requested portion of content.

Currently, users can make fine-grained transclusions of three frequently employed document types: JPEG, GIF and PNG images. It can be assumed that a small number of content providers will create fine-grained transclusions, and a large number of users will retrieve them. Therefore only a very simplistic interface for that task is provided: When users want to create a fine-grained transclusion they are asked to supply the URL of the original file as well as the coordinates that describe the region to be extracted (see Figure 5). Only rectangular selections can be made; more

complex geometric or even arbitrary shapes are not supported at the moment. In the course of time, tools that simplify the generation of fine-grained transclusions and offer enhanced functionality should be made available.

```
<img src="/images/test.jpg" alt="my image">
<img src="/images/test.jpg?shape=rect&coords=0,0,100,150"
 alt="my image">
<img src="/cgi-bin/get_mm_transclusion?url=/images/test.jpg&
shape=rect&coords=0,0,100,150" alt="my image">
```

Listing 1: A conventional image tag (top). A tag with a fine-grained transclusion is transparent to users if an HTTP server extension module is employed. A fine-grained transclusion through a CGI script is not transparent to users (bottom).

Upon submitting these data to the server, the first CGI program creates a URL and displays an example of how to use the URL to integrate the fine-grained transclusion into HTML documents. Whenever a fine-grained transclusion is retrieved, i.e., an image whose source URL contains certain parameters is requested, the second CGI script is invoked. It loads the original file from its location and extracts the requested spatial region of the image according to the given parameters (see also Listing 1). Finally, a file with the selected region is sent to the client.

The prototype implemented impressively demonstrates the power and ease-of-use of the proposed technology. Along with several examples, it is available online at [Kolbitsch 2005].

A drawback of the current implementation is that it is realised as CGI application; hence it is not transparent to users. This means users can recognise that images are loaded through a CGI program. An advantage of the approach is, though, that even users that do not have the two CGI programs installed on their HTTP servers can make use of the functionality because the software allows the use of images with remote URLs.

A further shortcoming of the implementation is that original resources have to be loaded onto the server that hosts the CGI scripts because image files need to be present in order to be able to crop the desired region. This makes retrieval of a fine-grained transclusion of an image rather inefficient and slow. Moreover it causes increased network traffic.

Therefore, ideally the CGI program for retrieving the content resides on the same server as the content. In this case, downloading the original file can be reduced to a simple “file open” function call.

4 Application Areas

Fine-grained transclusions of multimedia documents in web-based environments can have numerous applications. The prototype implementation, for instance, is designed as part of a larger system that offers communities tools to work actively with content from electronic encyclopaedias and digital libraries in general (see [Kolbitsch and Maurer 2005a]). In this environment users have the ability to make text-based transclusions in order to quote, refer to, and re-assemble existing articles. With the

implementation of this proposal they will also be able to make transclusions of multimedia content at a high granularity.

The following sections give a brief overview of further, potential applications in differing areas from museums to news and learning environments.

4.1 News Providers

News channels usually retain a large number of audio and video interviews. In many cases, they are not made available on the web site of the news providers because the files are too large, the content is simply too long, or for economic reasons.

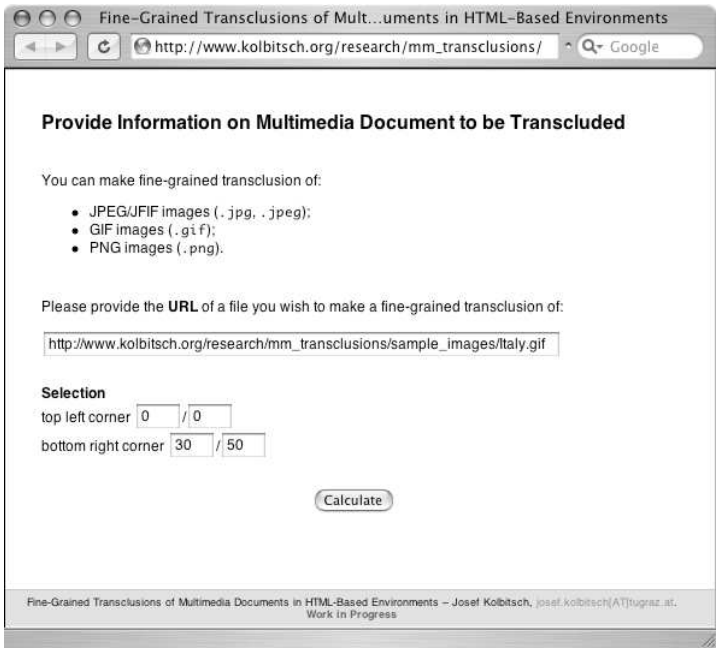

Figure 5: Screenshot of the prototype implementation. The user has to provide the URL of an image and the coordinates necessary to select a spatial region.

With fine-grained transclusions news providers could store one large file with the original interview in their internal database. On their web site they insert a fine-grained transclusion of the original content. The transclusion is, for instance, a temporal selection of an audio file that contains the most significant key issues of the interview. The entire interview (in full length) can, for example, be made available only to subscribers of the news service.

Thus news providers have hardly any extra effort in creating the short version of the interview, more or less no additional disk space is required, and providers are able to offer a complimentary service, while having the capability to retain full control of the content.

4.2 Galleries and Museums

Visitors in museums and galleries frequently encounter situations in which they require professional help:

- they look at paintings and do not know what to "see";
- they look at paintings and do not know how to interpret them;
- they are not familiar with the artist's life and social surroundings;
- they do not have adequate historical background; etc.

Therefore audio guides, and more recently PDA-based multimedia guides, are readily available in museums in order to assist visitors in their striving for comprehension.

Basically the same is true of people looking at paintings in online galleries or online museums (e.g., [Getty 2004]). With fine-grained transclusions it becomes possible to store the original painting *once*—as high quality image. When certain parts of the painting are explained, a transclusion of a spatial region is made, and the selection is described in detail. Thus, it is sufficient to store an image once, and when only a small section is referred to, a transclusion can be made.

4.3 Learning Environments

Similar to museums and galleries, web-based learning environments can utilise fine-grained transclusions to explain details of drawings and other media documents. In medical education, details of X-rays can be explained thoroughly, in biology in-depth descriptions of parts of photos resulting from electron microscopy can be given, etc.

In case of a sound file, only one minute of an entire opera can be transcluded into an e-Learning lesson. The selection contains the key scene of the work and is supplemented with the historical background, information on the composer, and a reference to the score of the opera.

In addition to this, lesson authors could virtually include a part of an external multimedia resource into a local lesson. A lesson for highschool students that explains a chemical experiment, for instance, can transclude a particular temporal selection of a related video available on the server of a university.

4.4 Movie Archives and Music Stores

In large archives of movies such as the Prelinger Archive (see [Prelinger 2005]) fine-grained transclusions can be employed for producing trailers and previews of the content on-the-fly. An entire movie, for instance, is stored in the system. When a user requests a description of the movie, a temporal or spatio-temporal selection of the movie is made and transcluded into an HTML page that gives a summary of the movie, lists a number of comments and provides a rating.

The same technology can also be employed in online music stores such as the iTunes Music Store (e.g., [iTunes 2005]). Previews of songs are created from the original sound file when users request it. Although the short preview clips would usually not be stored they can be retained within the system in order to increase performance. So when a preview is requested for the first time, the fine-grained transclusion is generated and cached in the system. On subsequent requests, the

transclusion is not newly generated but the preview is retrieved from the server cache instead.

Note that the use of cached files does not limit the generality of the concept of fine-grained transclusions. The relationship between the transclusion and its cached version is identical; caches are merely utilised in order to reduce loading time and improve overall system performance (see also [Nelson 1996; Nelson 1999]).

5 Conclusion

In this paper, an approach to introducing fine-grained trans-clusions of multimedia documents in HTML was presented, which constitutes a consistent extension to the concept of transclusions in HTML. With this technology, users have the ability to virtually include only a small region of an image, only a short clip of an entire movie, etc. into their web pages.

Two approaches to an implementation were discussed: a CGI-based implementation and an extension module for HTTP servers. Both methods offer the same functionality in terms of fine-grained transclusions while the levels of transparency vary, and their integration into existing systems (HTTP servers) is different. The suggested plug-in architecture offers an easy way to make an implementation open to third-party developers and facilitate the support of new document types.

A prototype implementation proves that the concept is reasonable and that the proposed technology can make reusing multimedia documents on the Web more efficient. Various application areas from news to museums and learning environments can have immediate benefits from the techniques presented in this paper.

Acknowledgements

This paper was supported by the Styria Professorship for Revolutionary Media Technologies.

References

[AIFF 1989] Apple Computer, Inc.: "*Audio Interchange File Format: 'AIFF'. A Standard for Sampled Sound Files Version 1.3"* (1989), http://www.tsp.ece.mcgill.ca/MMSP/Documents/AudioFormats/AIFF/Docs/AIFF-1.3.pdf, Accessed June 13th, 2005.

[Apache 2005] Apache HTTP Server, http://httpd.apache.org/.

[Foley et al. 1997] Foley, J. D., van Dam, A., et al., *"Computer Graphics: Principles and Practice",* Second Edition in C, Addison Wesley (1997).

[Getty 2004] The Getty Museum, http://www.getty.edu/art/.

[HTML 1999] Raggett, D., Le Hors, A., and Jacobs, I.: "*HTML 4.01 Specification",* (1999) http://www.w3.org/TR/html4/, Accessed April 28th, 2005.

[iTunes 2005] iTunes Music Store, http://www.apple.com/itunes/store/.

[Kappe 1995] Kappe, F.: "Maintaining Link Consistency in Distributed Hyperwebs", *Proceedings of the INET'95 Conference,* Honololu, HI, U.S.A. (1995). See also http://www.isoc.org/HMP/PAPER/073/html/paper.html.

[Kolbitsch 2005] Fine-Grained Transclusions of Multimedia Documents in HTML, http://www.kolbitsch.org/research/mm_transclusions/.

[Kolbitsch and Maurer 2005a] Kolbitsch, J., and Maurer, H.: "*Community Building around Encyclopaedic Knowledge"* (2005), to appear.

[Kolbitsch and Maurer 2005b] Kolbitsch, J., and Maurer, H.: "*Transclusions in HTML-Based Environments"* (2005), to appear.

[Krottmaier 2002] Krottmaier, H.: "Transcluded Documents: Advantages of Reusing Document Fragments", *Proceedings of the 6th International ICCC/IFIP Conference on Electronic Publishing (ELPUB2002),* Karlovy Vary, Czech Republic (2002), 359-367. See also http://hkrott.iicm.edu/docs/publications/elpub-2002.pdf.

[Macromedia 2005] Macromedia, Inc.: "*Macromedia Flash File Format (SWF) Specification License"* (2005), http://www.macromedia.com/software/flash/open/licensing/fileformat/, Accessed June 16th, 2005.

[Maurer 1996] Maurer, H.: "*Hyperwave—The Next Generation Web Solution",* Addison-Wesley Longman, London (1996).

[Maurer and Lennon 1996] Maurer, H., and Lennon, J. A.: "Flexible Link Architecture in Hypermedia Systems", *Proceedings of ED-MEDIA'96,* Boston, MA, U.S.A. (1996) 384-389.

[Nelson 1965] Nelson, T. H.: "A File Structure for the Complex, the Changing and the Indeterminate", *Proceedings of the ACM 20th National Conference,* Cleveland, OH, U.S.A. (1965) 84-100.

[Nelson 1981] Nelson, T. H.: "*Literary Machines",* Mindful Press (1981).

[Nelson 1995] Nelson, T. H.: "The Heart of Connection: Hypermedia Unified by Transclusion", *Communications of the ACM,* 38, 8 (1995), 31-33.

[Nelson 1996] Nelson, T. H.: "*Generalized Links, Micropayment and Transcopyright"* (1996), http://www.almaden.ibm.com/almaden/npuc97/1996/tnelson.htm, Accessed May 1st, 2003.

[Nelson 1998] Nelson, T. H.: "*Transcopyright: Pre-Permission for Virtual Republishing"* (1998), http://www.aus.xanadu.com/xanadu/transcopy.html, Accessed May 3rd, 2005.

[Nelson 1999] Nelson, T. H.: "Xanalogical Structure, Needed Now More than Ever: Parallel Documents, Deep Links to Content, Deep Versioning and Deep Re-Use", *ACM Computing Surveys,* 31, 4es (1999). See also http://xanadu.com.au/ted/XUsurvey/xuDation.html.

[Pam 1995] Pam, A.: "Where World Wide Web Went Wrong", *Proceedings of the AUUG'95 & Asia-Pacific World Wide Web '95 Conference & Exhibition,* Sydney, Australia (1995). See also http://www.csu.edu.au/special/conference/apwww95/papers95/apam/apam.html.

[Prelinger 2005] Prelinger Archives, http://www.archive.org/details/prelinger/.

[QTVR 2005] Apple Computer, Inc.: "*QuickTime VR"* (2005), http://developer.apple.com/documentation/QuickTime/InsideQT_QTVR/insideqt_qtvr.pdf, Accessed June 13th, 2005.

[SVG 2003] Ferraiolo, J., Fujisawa, J., and Jackson, D. (Eds.): "*Scalable Vector Graphics (SVG) 1.1 Specification"* (2003), http://www.w3.org/TR/SVG/, Accessed June 12th, 2005.

[Thau 2003] Thau, R. S.: "*Apache API notes"* (2003), http://modules.apache.org/doc/API.html, Accessed May 21st, 2005.

[Tichy 1985] Tichy, W. F.: "RCS—A System for Version Control", *Software—Practice & Experience,* 15, 7 (1985), 637-654. See also http://www.hpcc.ecs.soton.ac.uk/hpci/tools/cvs/ps/rcs.ps.

[X3D 2005] Web3D Consortium: "Information technology – Computer graphics and image processing – Extensible 3D (X3D)", *International Standard ISO/IEC FDIS 19775:200x* (2005), http://www.web3d.org/x3d/specifications/ISO-IEC-19775-IS-X3DAbstractSpecification/, Accessed June 12th, 2005.

[Yocom 2004] Yocom, M.: "*Mac OS History"* (2004), http://www.macos.utah.edu/Documentation/MacOSXClasses/macosxone/macintosh.html, Accessed May 3rd, 2005.