Transcluded Documents

Advantages of Reusing Document Fragments

Harald Krottmaier

Institute for Information Processing and
Computer Supported new Media (IICM)
Inffeldgasse 16c, A-8010 Graz, Austria
hkrott@iicm.edu

Abstract. Quoting parts of a document is a very common task when writing scientific papers. Usually an author cut-and-pastes a source quote to a new document, which inherently causes problems. Transclusions – invented in 1960 by Ted Nelson – are a possible solution: they are a kind of function which returns the data to be included in a new document and also adds some links to and from the original context of the quote.
This paper presents some ideas and problems when using transclusions in an existing digital journal environment. The creation process of transclusions with LaTeX-macros is described and tools supporting this task are listed in this paper. After uploading a special 'markuped' document to the server, a serverprocess is responsible for assembling the content. This includes the transfer of up-to-date content to the client, management of changes etc.
Additional navigation (i.e. hyperlinks) is also generated on the server side, therefore users are not forced to use a dedicated browser for the journal articles. Authors and readers of a document using transclusions profit from this architecture and the original context of a quote is just a mouse-click away.
The paradigm of transclusions is useful for all participants involved in the creation and reading of articles.

1 Introduction: Some Problems of Cut-and-Paste

Cut-and-paste parts of a document is "reuse of document fragments with duplication" [1]. *Reuse* of document fragments causes a lot of problems. In the first section of this paper we take a look at some of them. Thereafter we introduce the concept of transclusions which was first mentioned by Ted Nelson.

It is obvious that reusing parts of existing documents is an essential issue in producing new documents. New documents are usually based on existing documents, therefore it is necessary to refer to existing documents. It is also obvious that fine granularity is needful especially when reusing parts of very large documents.

Let us now discuss some of the problems of the "cut-and-past approach" related to text-based document development. The reuse of parts of a document in a new generated document introduces a lot of problems for both authors:

– The author of the original document is *not automatically* informed about the reuse of some parts of the document.
– The author using a quote is *not automatically* informed about any changes in the original document because of the lost original context of the quote.

The first problem is very obvious: while reusing some fragments of a document via cut-and-paste, the connection between the original document and the new created document is lost. Furthermore the author of the original document has no idea that a fragment of the written document is beeing reused in another document. This situation is very unsatifactory. Think of copyrighted material, where the author of the original document *must* permit any reuse of the document. Especially *freely and online* available articles are cited much more often than traditional documents or non-freely available articles (i.e. articles in e.g. submission based systems, [2]).

Unfortunately it is very common to reuse parts of a document without permission of the original author. Although there are some technical aids such as similarity search and fulltext search algorithms available, it is very difficult to automatically find documents which unlawfully reuse other documents.

With cut-and-paste it is not possible for the author of the original document to explore which parts of a document are most important for the community. It would be very helpful for authors to get this information! This information would:

– help readers to get the most important information out of the document
– increase the quality of upcoming publications by the author

Readers who are aware of the parts which are referred by other authors are able to explore the "orginal" contribution much quicker.

Additionally to very common annotation features which provides the author with some feedback about a publication knowing which parts of the document are important for other scientists matters and will influence the style of future publications.

Transclusions may be seen as 'additional meta-data' to parts of a document because the information: "this part of the document is used in another publication..." is in fact a kind of meta-data. Conventionally meta-data are of great use to the user – we are convinced that this additional *meta-data connected to parts of a document* will make an impact on the scientific community!

On the other hand, also the author who reuses parts of a document has to fight against the disadvantages of cut-and-paste: since the connection between the original document and the reused fragment of the document is lost, it is very difficult to get an automatic notification, if there are some modifications in the original document. This fact is a great drawback especially when selecting up-to-date reference lists or any other very dynamic resource.

There is not just an impact on authors, but also on readers of documents using the cut-and-paste approach: they are not informed about the original context of the quote! If a quote is declared as a quote in some other document, then usually an unidirectional reference to the original document is generated (very often via the reference section). If the document is also available online, usually a uniform resource identifier (URI, [3]) is given in the reference section. It is much more helpful to get a reference to the *exact position* of the quote (paragraph and/or page) instead of the whole document referenced by a URI. While there is a fragment identifier defined in the specification of the URI, fragments are not part of the URI itself but a property of the data resulting from a retrieval action.

The issue of disk-space must not be neglected. Although the hardware situation has changed dramatically since the early 1960's when transclusions were introduced, disk-space is an issue if transclusions are used e.g. in version control systems and personalized documents.

It has been shown that there are enormous problems with cut-and-paste of document fragments. In the next section we take a closer look at transclusions and compare them with the cut-and-paste approach.

2 Are Transclusions a Solution?

To simplify the following discussion we restrict the concept of transclusions to *'easy computer readable text-based and online available documents'* as source of transclusions. The reasons for these restrictions are:

easy computer readable: The format of the original document must be easy to read for a computer program, because some parts must be extracted and included in a new document. This will limit our first prototype to plain-text, HTML- and XML-formatted text documents.
Other open and very common document formats – like PostScript and PDF – and proprietary document formats – like the MS-Word `.doc`-format – are not considered in our first prototype. Nevertheless, it is very likely that adapters to these formats will be available in the future.

text-based: For our first prototype it will be easier to refer to parts of text than to refer to some part of a multi-media document. Multi-media documents are much more difficult in this respect: although images are easier to handle (e.g. 'include the upper left part of the image in my document' or 'compose an image out of X and Y...') than other multi-media document formats for videos, audios and 3D-scenes, we are going to limit the inclusions to text-based content.
For future projects it will be possible to adapt the prototype to handle much more sophisticated multi-media documents.

online available: If a document which uses transclusions is assembled on-the-fly, the content of the document fragments must be available to the assembling system by means of the http-request/response protocol. The hypertext-transfer protocol is quite easy to implement and widely available. If one wants to include locally stored data, some modifications have to be implemented (i.e. the content must be provided by a simple http-server and the objects must be referenced by some URI).
Due to possible server- or connection breakdowns, it is possible that a document cannot be constructed. Document fragments may not be available to the system. There are some possible solutions to this problem:

1. Transclusions are used just for intra-server documents, therefore the availability of document-fragments is guaranteed. This approach limits the field

of transclusions but will work fine for some kind of applications (e.g. for a discussion forum).
2. Extra-server documents (or the transcluded parts of an extra-server document) are cached on the assembling server system. This approach will prevent possible problems when assembling documents but some advantages of transclusions are lost.

If we limit the applications of transclusions to intra-server applications, we can guarantee proper assembly of the document.

It is quite easy for the system – when using transclusions with these constraints – to assemble a document out of document fragments and send this assembled document to the client-side. Let us now start the discussion of the advantages of transclusions compared to the cut-and-paste approach from the creation process to the delivery of the assembled document.

2.1 Creation of Documents Using Transclusions

When using transclusions, the connection between the quote and the original document is not lost. The connection is established via highly sophisticated reference linking between the original document and the reused quote in the new document. In our prototype we are going to use a Hyperwave Information Server (HIS, see e.g. [4] and [5]) to 'connect' these different parts of a document.

The creation process of trancluded documents is a follows:

1. Write document source. In our first implementation we are going to support documents written in LaTeX.
2. Identify quotes from online available documents in the specified format.
3. Describe (i.e. refer to) the quote with a special LaTeX-macro.
4. Build document output format (HTML, PDF, XML,. . .).
5. Upload document to the server system.

The identification of the quote to be included in the new document will be supported by additional software: Much work has been done to define exactly some part of the HTML-page (see e.g. [6] or [7]). Graphical tools like the WysiWyg Web Wrapper Factory (W4F, [8]) make it quite easy for the user to specify the textregion to be included in the new document. The quote is identified by some XPointer-expression ([9]).

Building the document (i.e. the creation of the output-format of the document) is very similar to the convenient build-process of LaTeX-documents. Evaluation of the specified transclusion-macro e.g.

```
\tranclude[some parameters]{URI}{XPointer-Expression}
```

is performed depending on the output format. LaTeX was chosen because of several advantages over other publishing systems:

– LaTeX is well known in the scientific community and (quite) easy to use.
– LaTeX is very often the source-format of high-quality publications.

– It is very easy to produce different output-formats of LaTeX-formatted input, including PDF, PostScript, HTML and XML without any user-interaction.
– LaTeX-formatted documents are easy to parse because they are written and stored in plain-ASCII format.

After the document is written, the document must be uploaded to the system. The uploading process is responsible for:

creation of links: Link objects must be created in order to refer the selected quote

notification of the original author of a quote: if authors of contributions are interested to know whether some new documents are quoting their articles, they may request an automatically created email.

In HIS all hyperlinks to *and* from documents are automatically stored in a link-database and are represented by link objects. Every link is bidirectional i.e. it is possible to reach the source link if the destination link is given. With this link-database some features are system immanent:

links are stabel: If the destination of a link is moved from one location to another (in an intra-server environment), all links to this document are automatically updated. Therefore the most common error "`404: Document Not Found`" is avoided by the system and the user is not confronted with this error.

links have attributes: Arbitrary and system dependent link attributes may be added to link objects. It is possible to add e.g. a Right-attribute to link objects. The right-concept is very much straight forward (including read-, and write-access to objects). A link may therefore be a kind of 'private' if the appropriate attribute is set to 'just visible to a specific user'.

links are separated from the content: Since links are not stored in documents, it is possible to add links to arbitrary documents in the system. The document must not be altered when creating links from the document to some other document.

In our concrete implementation of the transclusion concept, we are going to use these link objects with special attributes to create the connection between quote and original document. These special attributes are evaluated during visualization of both documents: those using transclusions and transcluded documents.

2.2 Visualization of Documents Using Transclusions

As noted above we consider in our first implementation of the transclusion concept *'easy computer readable text-based and online available documents'* as the sources of new documents. Independently of the location of the 'source' document the visualization process can be described as follows:

1. visualize links to the original context of the quote
2. get and include the current content of the transclusion

Since the idea of transclusion is implemented with standard features available in HIS, just some simple modifications have to be implemented: Links to original context of the quote must be visualized. Thereby we use attributes stored in the link object. These attributes are evaluated during the display of a document.

If the original document is stored on some HIS, necessary features for transclusions such as version control, links from the original document back to the usage of the quote etc. are available to readers. Nevertheless, we are not going to restrict the *location* of the source documents. Simple HTTP requests are performed to get the content of the quote. All necessary information of the quote – including creation date, some data finger-print to detect changes in the content etc. – are stored in the attributes of the link object!

If the format of the document is HTML the visualization process is exactly as described above. When some other output formats are used, some additional work has to be done. Depending on the speed of the creation of the document this can be done on-the-fly or in a batch-job performed by the system in a regular manner.

2.3 Features and Problems

In the creation process the author of a document using transclusions decides which notifications are appropriate for each used transclusion. While or after uploading, it must also be decided, which action must be taken if another author is going to use some parts of this document. Some features are very much dependent on the features supported by the source system providing the original document. In this section we provide a short listing of features and possible problems.

Notification of changes in the original document: In [10] some methods on how to detect changes in content and layout of HTML-pages are described. Using these methods makes it easy to observe the relevant parts (i.e. the parts used in transclusions) of the original document. Since link objects are bidirectional in HIS it is possible to actively inform every author of documents using parts of the documents via the transclusion technology. Possible actions of such a change notification are:

- **Accept the new version:** If the content of the transclusion is still appropriate in the context of the transclusion, the author may accept the new version.
- **Refuse the new version:** If the content of the original document changes completely, the author may delete the transclusion or – if the old version of the original document is still available – refer to the old version. Highly sophisticated publishing systems like HIS provide the author of documents with version control facilities.

  If the publishing system does not provide version control of documents, some caching mechanism on HIS may serve the original document or parts of the original document.

Nevertheless, an author of a document may ignore changes in the original document.

Frequent changes: If the original document changes very often, it might not be suitable as source for transclusions. It depends very much on the application of transclusions. Integrating a dynamic and rated list of references might be a good choice for transclusions.

Deleted source document: If the source document of a transclusion is deleted, the authors using parts of that document must be informed by the system. Nelson suggested the write-once Xanadu system where deletion of objects is not possible. Similarly we suggest to prohibit deletion of documents used by transclusions, if the documents are under the control of the server administrator. Nevertheless, it is generaly not possible to protect any document from deletion.

Moved source document: If the source document is moved to some other location the transclusions must be updated. Please note that links are stable in HIS and therefore no additonal action must be undertaken for intra-server documents. Extra-server documents must be handled separately (by *remote objects*).

Reuse-notification: If some part of a document is reused by another author, the author of the original document may be notified by email.

Changehistory: Again, if the system provides version control, the author and the user may browse different versions of the documents.

As one can see, transclusions depend very much on the offered features of the publishing system. Version control and bidirectional links are required features to make an implementation of transclusisions work. It has been shown that transclusions are most powerful when used in an intra-server environment on some high sophisticated server system.

3 Conclusion and Future Work

With the knowledge of the transclusion concept many existing applications can be improved. When using transclusions, readers of a document using transclusion are able to jump directly to the context of the qute (i.e. to the original paper). It is therefore easily possible to explore the original context of the quote. On the other hand readers of the transcluded document are able to see the most important parts of this document.

The next steps are to continue the implementation of the presented ideas in a prototype using a Hyperwave Information Server, and to make usability studies whether transclusions are accepted by the community and how they are used.

References

1. Ted Nelson. *Literary Maschines.* Ted Nelson, 1987.
2. Steve Lawrence, C. Lee Giles, and Kurt Bollacker. Digital libraries and autonomous citation indexing. *IEEE Computer*, 32(6):67–71, 1999.
3. T. Berners-Lee, R. Fielding, U.C. Irvine, and L. Masinter. Uniform Resource Identifiers (URI). RFC 2396, August 1998.

4. Hyperwave Informatino Server, 2001. `http://www.hyperwave.com` .
5. Hermann Maurer. now Hyperwave — The Next Generation Web Solution, 1996.
6. Frankie Poon and Kostas Kontogiannis. i-Cube: A Tool-Set for the Dynamic Extraction and Integration of Web Data. *LNCS 2040*, pages 98–115, 2001.
7. G. Huck, P. Fankhauser, K. Aberer, and E. Neuhold. JEDI: Extracting and Synthesizing Information from the Web. In *Cooperative Information Systems (COOPIS)*, 1998.
8. A. Sahuguet and F. Azavant. Wysiwyg web wrapper factory. In *WWW Conference 1999*, 1999.
9. Steve DeRose, Eve Maler, and Ron Daniel Jr. XML Pointer Language (XPointer) Version 1.0, 2001. available online `http://www.w3.org/TR/2001/CR-xptr-20010911` .
10. Luis Francisco-Revilla, Frank Shipman, Richard Furuta, Unmil Karadkar, and Avital Arora. Managing change on the web. In *Proceedings of the first ACM/IEEE-CS joint conference on Digital libraries*, pages 67–76. ACM Press, 2001.