XML Transclusions: a Feasible Path

Angelo Di Iorio, Silvio Peroni, Fabio Vitali
Dept. of Computer Science
University of Bologna
Mura Anteo Zamboni, 7
40127, Bologna, Italy
{diiorio | speroni | fabio}@cs.unibo.it

John Lumley, Tony Wiley
HPLabs Bristol
Filton Road, Stoke Gifford
Bristol BS34 8QZ
United Kingdom
{john.lumley | anthony.wiley}@hp.com

ABSTRACT

The idea of *transclusion* has been at the same time the strength and weakness of Xanadu: some people considered it as an extremely powerful mechanism to get any version of any fragment of any document in a global shared document space, others as a very complex solution too difficult to be actually implemented and delivered. We believe transclusions are still worth implementing and would allow designers to build very sophisticated hypermedia applications. On the other hand, we are aware that the original design of Xanadu cannot be implemented uprooting the current systems, protocols and technologies - *in primis* the World Wide Web and XML. In fact, there is a great distance between the original data model of transclusions - strongly based on external referencing mechanisms - and the XML data model - strongly based on hierachical structures and embedded markup.
This paper investigates to what extend the concept of transclusion can be shaped for the world of XML, and studies simplified models for building functionalities inspired to Xanadu. Particular attention is given to the support for tracing fragments provenance in multi-source documents and for synchronizing distributed content through transclusions. The paper also traces a roadmap to actually implement transclusions for XML - identifying three incremental steps - and briefly describes some experimental prototypes.

Categories and Subject Descriptors

H.5.4 [**INFORMATION INTERFACES AND PRESENTATION (I.7)**]: Hypertext/Hypermedia

1. INTRODUCTION

The concept of *transclusion* is rooted in the early days of hypertext[10]. A transclusion is a very advanced inclusion, whose content is not actually copied but stored as a virtual reference to the original source. There is only one copy of each fragment in the whole document space and transcluded data is permanently connected to the original.

Transclusions were the core idea of Xanadu. Xanadu documents were built on-the-fly from fine-grained references so that users could access, modify and reuse any fragment from any document in a safe and controlled way. The implementation of transclusions relies on external referencing to locations in a text data stream, through complex addressing mechanisms. Mark-up information, links and metadata are expressly distinct from the flow of text in order to guarantee flexibility and expressiveness.

The world of XML (and SGML) relies on a completly different strategy: mark-up is embedded, documents are strictly hierarchical and assertions about text fragments are made by wrapping them with elements, in case enriched by attributes. Ted Nelson himself pointed out disadvantages of such an approach in relation with transclusions[9]. The three objections he raised can be summarized as: (1) SGML approach interposes a 'forced' structure between users and actual content while editing, (2) SGML approach only supports well-formed inclusions and does not allow users to change included content, (3) SGML approach does not support overlap and non-hierarchical relationships.

Thus, these two positions seem to be irreconciliabile. The goal of this paper is to investigate whether and how transclusions can be implemented for XML documents and trace a possible course towards that goal.

The preliminary step is to understand what we mean for 'XML transclusions'. Our goal is not to re-implement a revised version of Xanadu based on XML technologies, rather to support users in creating and editing composite XML documents that make some xanalogical functionalities possible. In particular, we are interested in: *tracing fragment provenance* and *remote manipulation*. First of all, composite XML documents would benefit rich information about the origin of each fragment. That makes it possible to identify single contributions in collaboratively edited documents, to display multiple changes in a single document, to go back to original resources and navigate documents in a free and powerful way. The permanent connection between transcluded content and original sources would also make possible sophisticated forms of editing. Changes to remote documents could be propagated through transclusions or - the other way round - local modifications could actually update remote resources. Yet, such scenarios also require other tricky issues to be addressed such as content merging, synchronization, access permissions, reliability and so on.

The core of this paper discusses three approaches for implementing XML transclusions, in section 4. These solutions use different syntaxes and are progressively expressive and powerful. Before that, we briefly review the recent literature about transclusions and XML. The paper also mentions some prototypes we are developing.

2. TRANSCLUSIONS AND XML

The two most recent research efforts for implementing transclusions are outside the XML universe. Nelson and his team proposed Transliterature[8] a revision of the original Xanadu project built on newer technologies. Transliteral documents are dinamically built on top of transcluded fragments, so that rich and fine-grained connections between documents are permanently available. Two prototypes are worth mentioning: *Transquoter*, allowing users to hide/hightlight/surf multi-source quotations and *XanaduSpace*, providing users a 3D view of the overall document space and showing some of the advanced surfing and displaying functionalities envisioned for such inter-connected documents.

Kolbitsch at al.[5] investigated transclusions for HTML. Authors presented a prototype that allows users to select content from web pages and transclude them into new documents. Transclusions directives are stored as in-line elements and very complex URLs. Interesting issues are still open about content addressing and merging, especially considering that web pages may change often and may have complex (and bad-formed) internal structures.

The closest solution to transclusions in the XML universe is XInclude[7]. XInclude is a W3C standard for merging XML documents, by writing inclusion directives and retrieving other (parts of) documents. The focus of XInclude is on well-formed XML fragments. Although even text fragments can be included (`parse="text"`), XInclude does not allow users to include bad-formed fragments or ranges. That makes it impossible to implement full-fledged transclusions. Simplified forms of transclusions can be possible through XInclude anyway. XIPr[13] is an implementation of XInclude in XSLT 2.0, very efficient and simple to be integrated in other XML applications. XSLT technologies could be also used straightforwardly for simplified transclusions: a general approach is discussed in [3].

XLink[1] could also be cited as a way to transclude pieces of content. The `@show` and `@actuate` attributes allow users to define *hot-links*, that are similar to transclusions apart from implementation details. On the other hand, such a solution is only partial and XLink does not seem to succeed as expected.

3. A CASE-STUDY: COLLABORATIVE REVIEWS

The idea of transcluding fragments from and to XML documents is still under developed. Hereinafeter we discuss a road-map to make that development possible, by using a case study throughout the paper.

Let us suppose we are building a system for supporting multiple users to collaboratively write reviews about movies. It would be indeed useful to let reviewers quote fragments from other reviews, keep trace of their source and mantain a 'live' channel between connected fragments. In fact, a review process can be improved by quoting opposite criticism, by letting reviewers to access related reviews, by automatically update distributed reviews with new material and by fostering discussion among reviewers. It is worth noting that such an idea is both rooted in the early days of hypermedia (the essence itself of hypermedia is the inter-connection between documents) and central in the recent trend of the World Wide Web (one of the milestones of the so-called Web 2.0 is just collaboration).

Consider now that reviewers are writing their comments about 'Australia', starring Nicole Kidman and Hugh Jackman[1]. The film is one of the most controversial of the early 2009. David wrote a quite positive review: "Baz Luhrmann's Australia is good, but not a masterpiece".[2] Brad has an opposite opinion: "A major miscalculation if there ever was one"[3]. Assume that Brad replied to David quoting his original note. In fact, a third reviewer - say Mike - might be interested in quoting both these opinions, even by citing a different comment by a fourth reviewer (for instance, Rebecca saying "I actually regret having seen the film through to the end."[4]). Figure 1 shows a possible view of such a composite review, highlighting multiple contributions:

I am now looking forward to watch this movie. Too controversial points of view in our small group…Just read Brad's review: "I have to fully disagree with David. How can you say that 'Australia is good, but not a masterpiece'? I would rather define it 'a major miscalculation if there ever was one'". And Rebecca's review is even worse, if she says that she actually "regret having seen the film through to the end".

Figure 1: A possible view of Mike's review, highlighting multiple contributions

The multi-contribution view is only one of the applications of such an advanced quoting. Users might also be given the possibility to surf to the original review, in order to collect more information about the movie. Moreover, permanent connections between fragments could be enriched with metadata (stating, for instance, that the remote review is 'positive' or 'negative'), so that advanced search could be performed over the network of documents. Note also that such a composite view makes it possible to rebuild links between all reviews (even the first two) from the document itself, without requiring remote documents to be available.

Last, but not least, connections across fragments can be used as 'channels to make documents communicate'. Such an approach can be considered as a *simplified form of transclusions*: transclusion as a 'live and bidirectional channel'

[1]http://www.australiamovie.com/
[2]http://www.theaustralian.news.com.au/story/0,25197,24670334-601,00.html
[3]http://www.ropeofsilicon.com/article/movie-review-australia-2008
[4]http://movies.about.com/od/australia/fr/austral-review.htm

between pieces of content. The term 'channel' stresses the fact that such a mechanism would allow automatic updates of content, in both directions: from a remote review to all reviews including that fragment and from a transcluded fragment to the original review it belongs to. Yet, the automatic update of content opens tricky issues of synchronization, conflict resolution, priority, content merging and so on. On the other hand, it opens fascinating perspectives for collaborative documents accessing and editing.

The natural candidate to mark-up these composite documents is XML. It is standard, universally supported and very powerful. Surprisingly enough, none of the XML applications we are aware of provides users all these xanalogical functionalities together. The problem is paradoxically the same hierachical structure of XML, that makes impossible to address and overlap fine-grained transclusions[9]. The question is now at a different level: to what extend XML transclusions can be implemented? Is it possible to create a full-fledged xanalogical environment even for XML documents and use it for 'collaborative reviews'?[5]

4. CHARTING A COURSE FOR XML TRANSCLUSIONS

Intermediate results are also interesting, towards a full implementation of XML transclusions. This section first discusses some partial objectives - that can be achieved today - and then focuses on long-term developments.

4.1 Step 1: Embedding simplified forms of XML transclusions

The extension of XInclude makes possible to support simplified forms of XML transclusions. Let us encode the use-case reviews as DocBook documents, whose quotations are actually XInclude inclusions. Figure 2 and 3 show simplified source codes for Mike's review (including Brad's and Rebecca's reviews) and Brad's one (including David's).

```
<para>Just read Brad's review: "<xi:include
href="brad.xml" xpointer="..." parse="..."/>".
And Rebecca's review is even worse: she actually
"<xi:include href="rebecca.xml" xpointer="..."
parse="..."/>".</para>
```

Figure 2: XInclude instructions in *mike.xml*

```
<para>I have to fully disagree with David. How
can you say that `<xi:include href="david.xml"
xpointer="..."  parse="..."/>'? I would rather
define it  `a major miscalculation if there
ever was one'</para>
```

Figure 3: XInclude instructions in *brad.xml*

An XInclude processor transforms *mike.xml* into a final document where all inclusions are resolved and content fragments are merged together. The (though partial) result we want to achieve is enriching that document with detailed information about the origin of each fragment. It is crucial that *all* inclusions traversed to get the content are stored in the document.

XInclude supports two types of inclusions: inclusions of (1) text fragments (`parse="text"` and `xpointer` addresses a sequence of characters) or (2) XML elements (`parse="xml"` and `xpointer` addresses a well-formed XML fragment). The second case obviously requires that the source documents already contain the XML elements to be included. In the example, all quotations are encoded through the DocBook `quote` element in *brad.xml*, *rebecca.xml* and *david.xml*.

A first solution consists of introducing in the output new elements wrapping the included content. For instance, we could use a new element `<xi:included>` decorated with metadata about the inclusion. On the top of that information, we can build applications to let users access metadata, surf to original content, identify contributions, update content, etc.

There are two main drawbacks for such an approach. First of all, any tool processing the final document has to know the element `xi:included`. It is then impossible to directly reuse legacy tools such as DocBook converters or renderers. Yet, tools that directly process children of unknown elements (basically 'ignoring' the presence of `xi:included`) could be used as well, but we cannot take such a 'transparent' behavior for granted. A second issue is related to the way users define inclusions. Consider a fifth reviewer quoting fragments from (the expanded view of) *mike.xml*: how can she/he set the `xpointer` attribute of inclusion instructions? She/he has to also consider the `xi:included` elements and write very complex expressions, unless XPointer addesses are filtered to 'ignore' those elements. In any case, a tangled and error-prone management of internal addresses is required. A detailed discussion about these issues can be found in[4], along with a possible solution based on Architectural Forms[12]. A running prototype is also presented in the same paper.

The idea is to resolve all inclusions and 'embed' information in qualified attributes (belonging to a reserved namespace), so that the main structure of the XML document is not altered. The presence of these attributes does not impact the basic processing, parsing and integrity of the document. On the other hand, full expressive information about inclusions is available. Figure 4 shows such a solution applied to the use case.

```
<para>Just read Brad's review: "<quote
xi:inclusion_history="brad.xml#xpointer(...)">I have
to fully disagree with  David. How can you say that
`<quote xi:inclusion_history="brad.xml#xpointer(  ),
david.xml#xpointer(...)"> Australia is good,but not
a  masterpiece </quote>'? I would rather define it
`a major miscalculation if there ever was one'
</quote>". And Rebecca's review is even worse: she
actually "<quote xi:inclusion_history="rebecca.xml
#xpointer(...)">regret having seen the film through
to the end</quote>".</para>
```

Figure 4: Resolving and embedding inclusions in *mike.xml*

[5]Note that many other scenarios would benefit such collaborative reviews approach. Just think about the creation of multi-source reports in a company that collect information from existing resources, internal documents and external publications.

The attribute `xi:inclusion_history` is an extension property of the XInclude standard to record data about inclusions. The example uses that attribute to store information about nested inclusions and spread that information all over the XML tree. Consider, for instance, the quotation 'Australia is good, but not a masterpiece': the document knows that it has been included from *brad.xml* but it was actually originated in *david.xml*. Thus, all copies of a given fragment are very well connected in a complex network of inclusions, available to both users and applications.

In practice, `xi:inclusion_history` was never given great importance and it was never really supported by the XInclude implementations. In fact, the prototype presented in [4] uses a different syntax. Syntactic details are not relevant here: what is really important is that record of inclusions are hidden within documents and can be activated on-demand. That information can be exploited to build the aforementioned simplified form of XML transclusions, allowing users to navigate transclusion metadata, to retrieve original content, to highlight multiple contributions and to automatically update content.

There many reasons why such a solution is only a partial step towards our goal. First of all, it only applies to the inclusion of well-formed XML, since qualified attributes - meant to be added to the included `infoset` - can be only added on elements. One of the consequences is that no information about 'dangling' inclusions is eventually stored in the document. Solutions based on XML comments or decoration of preceding/following siblings might be investigated but seem to be tortuous and ackward. The second - and more important - issue is that such a model does not allow a lot of very common inclusions. Consider, for instance, a reviewer interested in quoting only few words of an existing quotation, or a sentence that partially spans over two paragraphs, or an interval that spans over a bad-formed XML fragment. Such selections can be supported only by exploiting external referencing mechanisms, that go beyond the scope of the strictly hierarchical organization of XML.

4.2 Step 2: Externalizing XML transclusions

A further step to overcome the aforementioned limitations consists of storing data about transcluded content in external and ad-hoc data structures, instead of embedding that information in the document itself. The prototype presented in [4] exports a module handling such externalization, although only some transclusions are actually supported. Figure 5 shows the example document *mike.xml* represented through externalized transclusions.

The content - even the transcluded one - is stored as a plain stream of text while the `ted:out-of-band` data-structure contains pointers to text fragments and metadata about each fragment. Several advantages of such a representation can be outlined. First of all, it contains rich information about the nesting of transclusions as well as rich metadata about original sources, without interfering with users and applications that interact with the document. In fact, all transclusions data are stored externally and can be ignored by transclusion-unaware processors. The position of the `ted:out-of-band` data-structure is also worth discussing: it can be placed within the XML document itself

```
<ted:out-of-band>
  <ted:transclusion
       transclusion-id="d6e7"
       source="brad.xml#xpointer(...)"/>
  <ted:transclusion
       transclusion-id="d9e11"
       source="david.xml#xpointer(...)"/>
</ted:out-of-band>

<para>Just read Brad's review: "I have  to fully
disagree with  David. How can you say that
`Australia is good, but not a  masterpiece'?I would
rather define it `a major miscalculation if there
ever was one'". And Rebecca's review is even worse:
she actually "regret having seen the film through
to the end".</para>
```

Figure 5: Resolving and externalizing inclusions in *brad.xml*

(for instance, as a first child of the root) or even in a completely external file. The first approach produces a single self-contained resource that can be moved from one system to another without requiring any management of relative links; on the other hand, it adds extra-structures to the original XML tree. The second approach is completely transparent and decoupled from the document tree-structure but it requires applications to process links and set of resources. Both these approaches are valid, according to users needs and preferences.

However, the main strength of such an externalized solution is its capability of defining transclusions that span over text fragments, ranges or even overlapped transcluded content. While Architectural Forms only work for transcluding well-formed XML fragments - as information is embedded into attributes of the transcluded elements - external pointers may refer to any location in the document, through proper XPointer expressions. For instance, it is possible to identify a transcluded fragment by simply indicating a pointer to the *start-* and *end-* offset of that fragment, and without altering the document itself.
Note also that this approach allows users to store information about dangling transclusions (whose content is not available because of network errors or simply because of different regimes in accessing content) by simply adding a reference to the point where the transcluded content was supposed to be. There is no 'intruder' structure in the XML tree but very rich information is available about nested or empty transclusions.

On the other hand, such an approach opens very tricky issues of content merging and manipulation of references. The main problem is related to the 'bad-formedness' of included fragments. External references - and in particular XPointer expressions - allow users to also include *any* piece of an XML tree. Consider, for instance, a user selecting the last sentence of a paragraph and the first one of the following paragraph (a possible HTML source code: `` `concluding paragraph</p><p>And starting a new one' ``). Several questions arise from that simple selection: which is the best way to include that fragment into a new document? Is it a plain string (without the `p` elements), or does it generate two para-

graphs or is it a completely different structure? The adoption of an externalized approach require us to adress these issues and to build a general and flexible document architecture for manipulating fine-grained fragments. The consistence of external data structures is another very difficult issue to face. Consider as example the implementation of an editor for such documents: it requires any edit operation to be propagated to the external data structure in a consistent way. Even small changes require a complex network of statements and relations to be updated and manipulated.

4.3 Step 3: Further generalization and externalization via RDF and EARMARK

The aforementioned approach can be further improved, although it is enough to provide users with a full-fledged xanalogical environment. The idea is to exploit existing XML standards: instead of using ad-hoc data structures and operations, XML transclusions could be described through RDF (and OWL) statements. RDF is a language for representing information about resources in the World Wide Web[6], that allows users to make any assertion on any identifiable resource: statements on any fragment of text, multiple statements on the same resource and overlapped statements are all possible in RDF. The additional benefit of RDF lies on its standardization: expressing transclusions in RDF would allow us to exploit legacy tools of the Semantic Web community and to create sophisticated services of searching and resoning over transcluded content. Nevertheless, the adoption of RDF leaves still open all the very complex issues related to consistent update and manipulation of externalized transclusions.

We finally propose a further generalization of RDF to integrate in a single approach advantages of both embedded markup (*a la* XML, presented as a first step of our roadmap) and external annotations. A detailed discussion of that proposal - called EARMARK (Extreme Annotational RDF Markup) - can be found in [11]. EARMARK derived from the analysis of some limitations of RDF. The main problem is that RDF assertions need URIs to address resources: statements only apply to whole documents, fragments provided with an identifier, or fragments addressable by some URI schema. The XPointer standard (actually, the full XPointer schema of XPointer [2]) makes it possible to refer to arbitrary pieces of text and XML documents. The adoption of that standard in conjunction with RDF would solve all the URI-related issues. The problem is that the XPointer full schema has never been confirmed by the W3C and its approval seems to be quite far.

The second interesting aspect of RDF is the fact that all URIs are considered 'opaque' strings regardless of the type of resource they refer to. It is very difficult to differentiate an assertion about a text fragment from an assertion about an element, to differentiate classes of assertions and relations among assertions. The solution we propose, EARMARK, defines an ontologically sound model that formalizes the concepts of the XPointer schema, and a natural way to express *externally* both RDF assertions and embedded markup constructs in the same framework. We also implemented a first prototype translating documents from and to EARMARK format, RDF and embedded XML. Although the implementation of EARMARK is at a very early stage, such a 'semantic layer on top of transclusion' is very flexible and promising.

5. CONCLUSIONS

Transclusions have contributed to shape the most successful hypermedia projects, although they were never fully implemented. After several years from their invention - dated back to the early 60s - they still keep the original attractiveness and potentialities. The challenge for the community is now to adapt the original xanalogical design to the current technologies and systems, in particular to the World Wide Web and XML. The contribution of this paper is manifold: (i) a preliminary discussion of simplified forms of transclusions, viable for the world of XML, (ii) a roadmap to actually implement XML transclusions and (iii) a brief presentation of early prototypes we are developing in that direction. It is our intention to foster the discussion within the community and to actually integrate xanalogical funcionalitities in a renewed generation of Xanadu-like systems.

6. REFERENCES

[1] S. DeRose, E. Maler, and D. Orchard. XML Linking Language (XLink) Version 1.0. http://www.w3.org/TR/xlink/, 2001. W3C Recommendation.

[2] S. J. DeRose, E. Maler, and R. Daniel. Xpointer xpointer() scheme. World Wide Web Consortium, Working Draft WD-xptr-xpointer-20021219, December 2002.

[3] B. DuCharme. Transclusion with XSLT 2.0. http://www.xml.com/pub/a/2003/07/09/xslt.html, 2003.

[4] A. D. Iorio and J. Lumley. From xml inclusions to xml transclusions. ACM Hypertext 09 or technical report.

[5] J. Kolbitsch and H. Maurer. Transclusions in an html-based environment. *Journal of Computing and Information Technology*, 14(2):161–174, 2006.

[6] F. Manola and E. Miller. RDF primer. http://www.w3.org/TR/rdf-primer/, 2004. W3C Recommendation.

[7] J. Marsh, D. Orchard, and D. Veillard. XML Inclusions (XInclude) Version 1.0. http://www.w3.org/TR/xinclude/, 2006. W3C Recommendation.

[8] T. H. Nelson. Transliterature: A Humanist Format for Re-Usable Documents and Media. http://translit.org/.

[9] T. H. Nelson. Embedded markup considered harmful. *World Wide Web J.*, 2(4):129–134, 1997.

[10] T. H. Nelson. Xanalogical structure, needed now more than ever: parallel documents, deep links to content, deep versioning, and deep re-use. *ACM Comput. Surv.*, page 33, 1999.

[11] S. Peroni and F. Vitali. Earmarking documents for arbitrary, overlapping and out-of-order annotations. ACM Hypertext 09 or technical report.

[12] J. K. Truss. International organization for standardization. a.3 architectural form definition requirements (afdr. In *In ISO/IEC 10744:1997, Annex A, SGML Extended Facilities*, pages 1601–1608, 1997.

[13] E. Wilde. XIPr: XInclude Processor. http://dret.net/projects/xipr/, 2007.