Extensible Framework of Authoring Tools for Web Document Annotation

Masahiro Hori, Mari Abe, and Kouichi Ono
IBM Tokyo Research Laboratory
1623-14 Shimotsuruma, Yamato-shi
Kanagawa-ken, 242-8502, Japan
{horim, maria, onono}@jp.ibm.com

Abstract

Web metadata is crucial for providing machine-understandable descriptions of Web resources, and has a number of applications such as discovery, qualification, and adaptation of Web documents. While metadata is often embedded into a target document, metadata can also be associated externally by means of an addressing scheme such as the XPath language. However, creation and modification of external metadata solely with a conventional editor is not easy because metadata authoring involves the maintenance and elaboration of addressing expressions as well as editing individual documents. The objective of this study is to advance extensibility and variations in the configuration of annotation tools, taking account of different authoring methods as well as the different roles of annotations for assertion and transformation.

1 Introduction

Web metadata is crucial for providing machine-understandable descriptions of Web resources, and has a number of applications such as discovery, qualification, and adaptation of Web documents [14]. A remark attached to a particular portion of a document is called an annotation, and covers a broad range in the literature. Forms of annotations can be characterized by the dimensions: whether formal or informal, and whether tacit or explicit [16]. Metadata that follows structural specification resides at the most formal and explicit extreme. In this paper annotation and metadata are used interchangeably in this restricted sense.

While metadata is often embedded into a target document, metadata can also be associated externally by means of an addressing scheme such as the XPath language [22]. However, creation and modification of external metadata solely with a conventional editor is not easy because metadata authoring involves the maintenance and elaboration of addressing expressions as well as editing individual documents. Configurations of annotation tools depend on the annotation scenario. Browser-based annotation tools [3, 7, 17] are desirable when annotators are not allowed to edit target documents without document ownership. On the other hand, an annotation tool based on a WYSIWYG editor [9] is helpful when annotators are responsible not only for the creation of the annotations but also for the editing of the target documents. Regardless of the variety of emerging annotation tools, a significant limitation of the current annotation tools is the lack of extensibility, because the existing tools are developed solely for a particular annotation vocabulary, and provided with predefined views for the authoring.

In pursuit of Web content adaptation, we have been working for the development of an annotation-based page-clipping system [8, 9, 11], annotation authoring tools [1, 10, 13, 18], and empirical evaluation for the robustness of external annotations [2]. In particular, the page-clipping engine and clipping annotation tools are commercially available as software products of transcoding proxy [20] and portal server [4].

In this paper, we propose a comprehensive framework of authoring tools for Web document annotation. In particular, the extensibility of tool configuration is investigated on the basis of two authoring methods (annotation by selection and by example) as well as the different roles of annotations for assertion and transformation. In the next section, we explain a schema for external annotation which is specified as an XML Information Set [21]. Section 3 introduces an extensible framework of annotation authoring tools, and shows the three of typical tool configurations. Finally, we present practical applications of external annotations for Web document clipping, and show how annotation tools are used for annotation authoring.

2 XML Information Set of Annotation Document

The framework of external annotation prescribes a scheme for representing annotation files and a way of associating original documents with external annotations [9]. The basic ideas behind this framework are twofold. One is that it should not be introduced new elements or attributes into the document type definitions of the target documents to be annotated. The other is that annotations need to be created for arbitrary parts of annotated documents. External annotation files contain metadata that refers to a part of a document to be annotated. XPath [22] is used to
associate annotated portions of a document with annotating descriptions. An external annotation document refers to portions of an annotated document. A reference may point to a single element (e.g., an IMG element), or a range of elements (e.g., an H2 element and the following paragraphs). For example, /HTML/BODY/P[3] points to the third P element of the BODY element of the annotated document. If a target element has an id attribute, the attribute can be used for direct addressing without the need for a long path expression.

One annotation vocabulary would not fit for all the requirements of metadata representation. Moreover, it is also impractical to expect to provide a common basis for a generic portion covering all annotation vocabularies. This difficulty comes from the approach of defining document type specifications based on the grammatical aspects of specific vocabularies, because the same contents may be described with different grammars. Therefore, in order to clarify the edit-time constraints, it is important to characterize a class of annotation documents, without regard to rigorous grammatical definitions.

Information available from an XML document can be specified as an abstract data called XML Information Set (Infoset) [21]. Figure 1 shows a RDF schema [19] of the annotation document infoset. Features of the annotation document infoset are described as follows. Note that terms in bold face indicate information items, and the terms in italic are items specific to the annotation information set.

- A description element is an element item.
- An XPath attribute is an attribute item, and may take an XPath expression as its value.
- An XPath attribute is owned by a description element (owner description property).
- The parent of a description element is a document item (owner annotation document property).

![Nodes with bold outlines are instances of rdfs:Class.](image)

**Table 1: Variations in annotation tool design**

<table>
<thead>
<tr>
<th>Roles of annotation</th>
<th>Authoring methods</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>By selection</td>
</tr>
<tr>
<td>Assertion</td>
<td>(a)</td>
</tr>
<tr>
<td>Transformation</td>
<td>(b)</td>
</tr>
</tbody>
</table>

The annotation document infoset mentioned above provides constraints on the structure of annotation documents, and exploited as edit-time constraints to be satisfied by annotation tools. The annotation tool framework presented in the next section incorporates the specifications of the annotation information set into an annotation profile.

### 3 Framework of Annotation Tools

An annotation in general declares properties that qualify a particular portion of a target document. Moreover, annotations may indicate structural changes for the annotated portion of a target document. In order to clarify the distinction of these two roles, the former can be called *assertional* annotations, while the latter *transformational* annotations [10]. It is important to note that this distinction is not exclusive, because every annotation is intrinsically an assertion.

It is straightforward for annotation authors to indicate a location to be annotated and create an assertion as annotation content. This is an approach that we call *annotation by selection*, and is adopted by existing annotation tools [1, 3, 7, 9, 12, 17]. On the other hand, for transformational annotations, it is easier for the authors to modify a target document toward the desired results of the customization, rather than to indicate the ways of modifications declaratively as assertional annotations. This is a basic idea behind an approach what we call *annotation by example*, which was originally proposed for the automatic generation of document transformation rules [13] and also applied for automatic generation of transformational annotations on the basis of users’ editing operations for the content customization [10].

According to the distinctions of annotation authoring methods and roles of annotation, Table 1 summarizes variations in annotation tools. It is straightforward for annotation authors to select a portion of document to be annotated and declare properties on the selected portion as assertional annotation. This type of annotation tools support assertional annotation by selection [Table 1(a)]. Even when annotations are used for structural changes of a target document, it is possible for authors to create transformational annotations by selecting portions to be changed and declare instructions of transformations as annotations. This type of annotation tools support transformational annotation by selection [Table 1(b)].

In order to create transformational annotations, however, annotation authoring by example would be much easier
for users, because users can work with a concrete example and create a desired result interactively with the example. In particular, the example-based method allows users to automatically generate transformational annotations on the basis of users’ operations conducted to come up with a desired result. This type of annotation tools follows transformational annotation by example [Table 1(c)]. This example-based method is particularly useful for the transformational annotations, but would not make sense for assertional annotations, because it is not intuitive for users to indicate assertional annotations as results of structural changes of a target document to be annotated.

Figure 2 shows an overview of annotation tool framework, which realizes the three configuration of annotation tools explained above. The core part of this framework is independent of any particular views and editors, and prescribes the internal document models [6] and their relations common to all the annotation tool configurations. It is assumed here that the creation of an annotation document is a primary task of users, and the users are not allowed to modify a target document as well as the annotation document. The important point here is that the constraints imposed on the core part come from the annotation document infoset, and make the tool framework extensible allowing addition of authoring capabilities as needed. In the remainder of this section, the assertional annotation by selection and the transformational annotation by example are explained respectively in Sections 3.1 and 3.2. The transformational annotation by selection is explained in Section 4.2 together with an application to document clipping for portal pages.

3.1 Assertional Annotation by Selection

Figure 3 shows a tool configuration for the assertional annotation by selection. In addition to the core components, this configuration includes an XPath composer with a target document viewer [Figure 3(a)], and an annotation document editor [Figure 3(b)]. The annotation document editor is provided with an annotation profile, which allows customization of the editor for different annotation vocabularies. In addition, since the edit-time representation of external annotations can be saved as an inline annotation file, this tool configuration can be given with a component for importing annotations into a target document [Figure 3(c)].

Figure 4 shows a screen copy of an annotation tool, which is developed for the assertional annotation by selection. The main window is divided into two panels. The left pane is the area for the target document viewer, and also contains a target view and a source view in the tabbed pane. The right pane is for the annotation document editor containing a tree view. When the target is an HTML document, users can employ the browser view. The browser view is not embedded in the split pane because it is often helpful for users to compare the selection between the target tree view and the browser view.

The annotation tool shown in Figure 4 can be customized for different annotation vocabularies by selecting one of the annotation profiles. Figure 5 shows the profile selection dialog. When a profile is selected from the pull-down list, the dialog shows the definition of the current profile. An annotation profile, which is depicted in Figure 3(b), includes the location of a DTD file, an annotation file type, the location of a data directory, and the names of the information items shown in Figure 1. In the figure, the “EML Sample Annotation” profile is selected, and the annotation documents are characterized in terms of the root element name annot, the annotation description name description, and the XPath attribute name target. The annotation editor assumes that a target attribute is owned by a descriptions element that is an immediate child of the root element (see Figure 1).

According to the annotation document infoset, an annotation document consists of description elements. Every description element must be given with an XPath attribute and annotation contents. The annotation tool in Figure 4 provides an XPath composer. The key idea underlying the XPath composer is to improve flexibility of the creation of XPath expressions with a seamless integration of three authoring methods: automatic instantaneous creation, context-constrained creation, and manual creation. Further details of the XPath composer are reported in another article [1].
In addition to XPath composition, it is necessary for annotation authoring to create the contents of the description item. This part of authoring relies on a DTD given with a corresponding annotation profile (Figure 5). The contents of the description item can be edited in the same manner as with a DTD-based structure editor on a DOM-tree view. When a description node is selected in the annotation tree view, a popup menu can be brought up by clicking the right mouse button as shown in Figure 6. The first popup menu is common to all annotation vocabularies, and provides menu items to add/remove an element, add/remove an attribute, and edit an attribute value. The second-level menus are derived from the DTD information of the current profile. Figure 6 shows a popup menu that appears after choosing the "Add the first child element" item, and the valid elements such as importance, date, and author are given as candidates for the insertion as a child of the description node. When the user chooses the "author" item, a dialog appears with an edit field for typing the text content, namely, an author name.

3.2 Transformational Annotation by Example

Transformational annotation has been used for Web content adaptation, in which structural changes of a target document are needed [9, 17, 20, 23]. Among those annotation languages, XSL Transformation Language (XSLT) [23] is well-known, and the document type definition of XSLT is actually compliant with the constraints of annotation docu-
Figure 6: Popup menu for editing an annotation description info-set explained in the previous section. In contrast to assertional metadata such as Dublin Core Metadata [5], transformational annotation languages are more like programming languages. Learning an abstract language and writing programs are not easy tasks for most people. However, if a person knows how to perform a task to be executed by a computer, perhaps the person’s knowledge can somehow be exploited for the creation of a program to perform the task. This is the idea behind programming by example [15]. Programming by example is a natural approach to creating the transformational annotation for page designers or novice programmers, because users need only work with examples of how to transform a document at hand, and are given automatically generated annotations that can replicates the same transformation.

On the basis of the idea of annotation by example, we have developed annotation generation tools [13, 18]. A configuration of the example-based annotation tool is depicted in Figure 7. With this annotation tool, first a user opens a target document to be customized (e.g., an HTML file). The user then edits the document by using the full capabilities of the WYSIWYG authoring tool. Although a user’s editing actions are recorded into an operation history, the user does not have to care about the recording process behind the scenes. When the editing is finished, the user will have a customized document. At the same time, the annotation generator creates transformational annotation for the customization, which can also be used by a runtime engine (e.g., XSLT processor) to replicate the transformation from the initial target document to the customized document. Further details on the annotation generation procedure are reported in the other articles [10, 13].

All the above-mentioned editing operations actually modify the target document by means of DOM (Document Object Model) manipulation operations [6], and can be created as transformational annotations. However, assertional annotations cannot be created on the basis of the DOM manipulation operations, but need to be declared explicitly as assertions. Annotation tools that follow the annotation by selection approach play a complementary role in such situations instead of the example-based annotation generation for the transformational annotations.

4 Applications of External Annotation

Annotations provide additional information about Web contents, so that an adaptation engine can make better decisions on the content re-purposing. The role of annotations is to provide explicit semantics that can be understood by a content adaptation engine [11]. Figure 8 depicts an overview of an annotation-based transcoding process. Upon receipt of a request from a client, a Web document is retrieved from a content server. Taking account of the capabilities of the client specified in the HTTP request header, a transcoding proxy selects one or more transcoding modules. When a selected transcoding module requires an annotation document, an annotation file is also retrieved from a content server, which may or may not be the same server that retrieved the Web document. The transcoding module may simply return the original document, if a client agent has the rendering capabilities compatible with ordinary desktop computers [Figure 8 (a)]. Alternatively, the original document may be returned with modification, so that the original content can fit into a small screen device [Figure 8 (b)]. The decisions about the content adaptation are made taking account of the client capabilities specified in the HTTP request header.
Figure 9: Simple example of an HTML page clipping

4.1 Annotation-Based Document Clipping

Web pages for e-commerce, for example, contain a lot of information such as details of products, product images, and numerous links to other areas of the site, when the pages are created for the desktop computers. However, it may be necessary to deliver portions of this page for users to access through a Web-enabled phone rather than a desktop browser. In such a case, the images and nested HTML tables prepared for a nicely laid out page are a hindrance rather than help. The sheer amount of information becomes unwieldy in the small display, and potentially expensive depending on the user’s wireless service.

Content adaptation can be done by using an annotation-based page-clipping engine [20]. At content delivery time, the page-clipping engine may modify the original document with reference to page-clipping annotations and client profiles sent over HTTP. The main idea in the page-clipping annotation language is the notion of a clipping state. By using <keep> and <remove> elements in the annotation descriptions, users can specify the clipping state to indicate whether the content being processed should be preserved or removed.

As a simple example, an HTML page and its clipped results are shown in Figure 9. In this example, the header and the first paragraph are preserved as shown in Figure 9(a). The table element is modified by deleting the third column and the second row. The cell-padding attribute of the table is increased, so that each table cell can be provided with margin space [Figure 9(b)]. In addition, the whole of the second paragraph is removed as shown in Figure 9(c). All the structural changes in HTML documents can be easily done by using a WYSIWYG HTML editor.

Figure 10 shows an annotation document that realizes the page clipping illustrated in Figure 9. This transformational annotation can actually be automatically generated by using the example-based annotation generation tool [10]. The <description> element prescribes a unit of an annotation statement in the annotation language. The target attribute is set to an XPath expression, and identifies the node on which the annotation will be applied, and the take-effect attribute indicates whether the annotation is applied before or after the target node. By specifying target="/HTML[1]/BODY[1]/P[2]" as in Figure 10(a), the clipping state is activated after the first element after the first <BODY> element, which in this case is an <H1>. The <keep> element in Figure 10(a) indicates that all the document elements encountered are preserved, until otherwise instructed by another annotation statement. The clipping state is changed to ‘remove’ just before the second <P> element [Figure 10(c)], and changed back to ‘keep’ after the <P> element [Figure 10(d)]. As results, the second paragraph element indicated by "/HTML[1]/BODY[1]/P[2]" is removed while preserving the elements just before and after the removed element.

Since HTML tables can often be complex elements to clip, the annotation language provides special-purpose elements to make table clipping easier. The <row> and <column> elements allow user to clip rows and columns without relying on complicated XPath expressions. The table-clipping elements are used in the description shown in Figure 10(b). This description sets the clipping state to ‘keep’ just before the first table element, and also changes the value of cellpadding attribute to 4 by using the <insertattribute> element. The name attribute of <insertattribute> can be specified with an arbitrary name of an attribute available for a target document.

In addition, the description element [Figure 10(b)] declares that the third column, which is indicated by the index value of the <column> element, is discarded, while the remaining columns are preserved. Note here that the wildcard character to indicate multiple columns (index="*"). If a wildcard is specified, all rows (or
Automatic Generation of Annotation by Selection

Core Components

both selection-based and example-based approaches to annotation operations. To put it another way, it is possible for users' intension can be expressed solely with the simple semantically generate an annotation document, as long as the followed the transformation annotation by selection can auto-
desired results for customization. However, the tools that lowed to modify or edit the target document to come up the only selects the portions to be annotated, and are not al-
ner to simple clipping capability to remove headers and side menus in the original documents that were created for browsers on desktop computers.

4.2 Transformational Annotation by Selection

Figure 12 shows a screen of an annotation tool for clipping portlet in the left, and a portal page that includes the clipped page in the right. This annotation tool allows a user to select the portions of the original page to be removed in the portal page, and the annotation generator automatically creates page-clipping annotations from the selected nodes.

This tool follows the configuration for transformational annotation by selection [Table 1(b)], which is depicted in Figure 13. This type of annotation tools rely on a target document viewer rather than an editor in the case of annotation by example (see Figure 7). Therefore, users can only selects the portions to be annotated, and are not allowed to modify or edit the target document to come up the desired results for customization. However, the tools that follow the transformation annotation by selection can automatically generate an annotation document, as long as the users’ intension can be expressed solely with the simple selection operations. To put it another way, it is possible for both selection-based and example-based approaches to au-

5 Concluding Remarks

In this paper, we presented a comprehensive framework of annotation tools on the basis of an XML information set of annotation document. In addition to the fundamental tool configuration, namely, the assertion-based annotation by selection [Figure 3(a)], we explained the other two tool configurations for transformational annotation that allow automatic annotation generation by either selection or example [Figure 3(b), (c)]. Automatic generation is an innovative approach to helping users with the generation of transformational annotations, since users of the generator do not have to learn the annotation language at all. This approach is particularly suitable for environments of annotation authoring by page designers or novice programmers who are not necessarily familiar with annotation languages.
References


