A Common Annotation Framework

David Bargeron
Anoop Gupta
A.J. Bernheim Brush

November 15, 2001

Technical Report
MSR-TR-2001-108

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

A Common Annotation Framework

David Bargeron, Anoop Gupta, and A.J. Bernheim Brush
Microsoft Research
Redmond, WA 98052
{davemb,anoop}@microsoft.com, ajb@cs.washington.edu

1 Abstract

Annotating resources on the World Wide Web in a flexible manner is a key problem, yet there is no standard for annotations on the Web. We propose a design for CAF, a Common Annotation Framework that supports annotation across applications in a standard way. The framework is based on a principled logical model, and is designed to meet a wide set of annotation needs without overburdening any particular application with excessive functionality. It is flexible and extensible, and can support annotation of any media type with any media type. We discuss the structure and function of the CAF logical model, and we describe a proposed CAF schema. We explain how CAF can be extended and give examples. And we present a web-based annotation application built using CAF.

2 Keywords

Annotation, schema, framework, linking, multimedia.

3 Introduction

As the World Wide Web matures, annotating web resources in a common and adaptable way is a pressing issue. Shared annotations are playing an increasingly important role in collaboration across the Web. Index, search, recognition, and summarization algorithms consume and produce metadata on everything from text to video. Individuals want to bookmark ideas and record thoughts as they traverse the Internet. And web page authors want to customize their sites for target audiences.

Today, these and other annotation activities are handled in different, mostly incompatible ways. A plethora of web-based research annotation systems has been developed to explore the issues [1][6][7][12][13][20], and a few commercial systems have been brought to market [11][19][21][24]. In addition, a number of standards exist which support annotations to some degree [8][9][10][15][18]. But to the best of our knowledge, no one has attempted to establish a single foundational standard for annotations which can support the wide variety of common annotation activities that occur today, and there are no standards which are flexible enough to handle new annotation activities that may arise in the future.

Yet it is important to establish an annotation standard for several reasons. First, it is a key step in establishing a rich, common user experience for annotations across the Web. Second, it facilitates interoperability among applications, so that if annotated content is copied from a web page to an email message, for instance, or it is converted from HTML format to PDF, the annotations on the content can follow along seamlessly. And finally,

an annotation standard can support code reuse and a common programming model, freeing developers to focus on higher-level annotation issues such as anchoring techniques and user interfaces.

In this paper we propose the Common Annotation Framework (CAF), which is designed to support annotation features across applications in a common and consistent manner. The framework is based on a principled logical model, and consists of a simple XLink-compatible schema combined with extensibility conventions. It is designed to meet the needs of a large variety of annotation applications, but we kept it as simple as possible so that no one application or platform is saddled with unnecessary complexity. The framework is flexible enough to handle many common annotation scenarios without being altered, and it can be extended to accommodate future needs. It is designed to support sophisticated annotation across media types. And CAF annotations are self-contained, so they can be integrated into a wide variety of applications for private, shared, or mixed use.

We designed CAF to meet a wide spectrum of requirements from a broad range of groups. We consulted with software product groups which had built (or wanted to incorporate) annotation features into their products, and we found that simplicity is paramount. We discussed design principles with members of standards bodies, principally the W3C XML Linking Working Group [9], and this reinforced our conviction that flexibility is important. And we learned lessons from our experience developing and deploying the MRAS system [2] and the WebAnn client [5], which taught us that extensibility is crucial.

In addition, we rationalized our design against related technologies such as XLink, XPointer, RDF, and MPEG-7. Xlink [9] and Xpointer [8] provide important standards for flexible resource linking and intra-resource addressing, respectively, but they do not provide enough structure to fully support annotation by themselves. RDF provides a much fuller structure, but it may be too restrictive for many common annotation scenarios. And MPEG-7, which attempts to establish a metadata ontology for video data, is too specialized to serve as a general-purpose annotation framework. Thus a new common framework for annotations is timely and appropriate.

The remainder of the paper is organized as follows. In the next section, we present the high-level goals motivating CAF. Next we describe the logical model underlying CAF. Following that, we explain the structure of Annotation Markup Language (AML), the core CAF schema. Then we describe conventions for extending CAF to support novel annotation scenarios, and we present WebAnn, an application we have developed using CAF. Finally, we discuss related work, and we provide some concluding remarks.

4 Design Goals

Our proposal for the Common Annotation Framework is based on five broad design goals: First, we wanted to keep the core framework **small and simple**, so that no one application or platform is overburdened with features it does not need. We wanted CAF to work well on relatively constrained platforms like Palm OS, Win CE, and cell phones. We also wanted CAF to be useful for more powerful platforms like network-connected PCs running full-featured desktop applications.

This leads us to our second goal: For CAF to be relevant to applications requiring richer functionality, it was imperative that CAF be **extensible** via addition of new properties, objects, and methods. Establishing conventions for extending the framework and providing a set of examples was also important to support code reuse and a common programming model around CAF. If one group develops a better way to anchor CAF annotations to video content, for example, this method can be reused by others who adopt CAF and want to annotate video. Striking a good balance between functionality in the core framework versus that in extensions has been one of the toughest challenges in designing CAF.

Our third goal was to keep CAF **storage-neutral** and to avoid wedding it to any one particular configuration for storing annotations. Given that CAF consumers may require support for annotations on a variety of devices, the

storage models will necessarily vary considerably. Different applications will also have different feature requirements, such as storing annotations within an annotated document for portability, storing annotations in a local cache for privacy or offline operation, and storing annotations in a common repository for shared use.

Our fourth goal was to support **universal annotation**, so that CAF would support as wide a variety of annotation scenarios as possible. Implicit in the CAF "simple core plus extensions" model is the desire that with extension, CAF should be able to support the annotation of all or part of any resource with all or any part of any other resource (e.g., regardless of media type, structure, format, etc.).

Closely related to this goal was the desire to facilitate a common user experience for annotations across media types, document formats, and devices. Today the various experimental and commercial annotation systems available on the Web present an unnecessarily confusing and inconsistent user experience for commenting, revising, and discussing documents. Providing a consistent experience in this area will likely lead to more effective use of annotations, and will result in greater value for the user. This was not a direct goal of CAF, however it is hoped that widespread adoption of CAF will result in greater uniformity of the annotation user experience.

Finally, our fifth goal was to make CAF **standards-compliant**, and to avoid duplication and reinvention wherever possible. For this reason we based CAF on XLink, so that CAF annotations are entirely compatible with Xlink 'extended' links.

5 Logical Model

With these goals in mind, we designed a logical model to provide a principled underpinning for CAF. At a high level, the core CAF logical model describes the structures and relationships necessary for anchoring any annotation to any resource. Consider, for example, the left hand side of Figure 1, showing a handwritten note in the margin of a text document. Intuitively, we want to describe how the handwritten annotation is anchored to the margin of the document. In CAF, we consider the handwritten note and the text document as two separate resources, and the annotation as the relationship between them. The core CAF logical model uses combinations of resources, anchors, and annotations in this way to represent all annotation relationships.

A *resource* in CAF is anything that can be annotated or can serve as an annotation (in the common sense of the word, not the technical sense in which we use annotation in this context). A resource may refer to something a user wishes to annotate, for instance an article he is reading. Or, it can refer to the content he intends to use as part of an annotation, for instance a comment he wants to make on the article. A resource may be a simple block of ASCII text, a complex word processing document, an audio/video presentation, a web page, an electronic calendar item, digital ink data, etc. For our purposes, a resource is only constrained by the requirement that it be addressable by URI [3], thus the CAF concept of resource agrees with common usage in the hyper linking literature [9]. In Figure 1, the article being annotated and the digital ink in the margin are two separate resources.

A CAF *anchor* contains information pertaining to a single resource. The anchor can either be "by-reference", in which case it references a resource by URI and may contain additional information about a single position within the resource; or it can be "by-value" and contain a resource literally, in which case it contains no additional positioning information. anchors are also categorized as either "context" anchors or "content" anchors. A "context" anchor refers to or contains the resource which contextualizes the annotation; or in other words, that which is being annotated. In Figure 1, the address and positioning data for the paragraph in the article being annotated are captured in the annotation's "context" anchor. A "content" anchor encapsulates information about the resource which is being used to annotate the annotation's context. In Figure 1, the literal data representing the digital ink comment are captured in the annotation's "content" anchor.

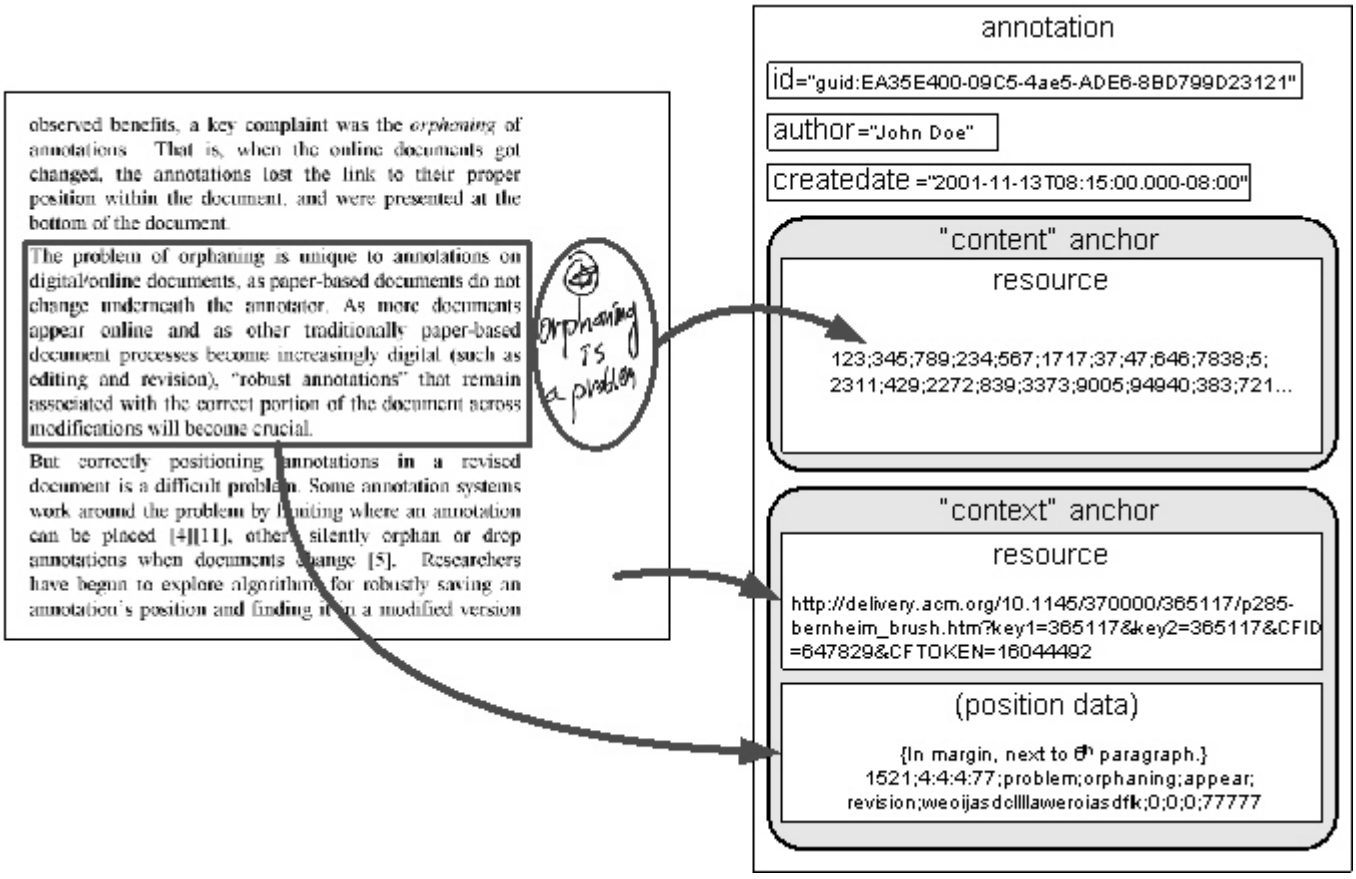


Figure 1: Common Annotation Framework (CAF) logical model for a handwritten annotation on a text document. CAF annotations are self-contained and symmetrically structured to support a wide range of annotation functionality. The numbers in the "content" anchor represent digital ink coordinates.

The CAF *annotation* object draws together a collection of anchors and encapsulates the annotation relationship among the resources they specify. It also holds metadata properties describing characteristics of the annotation relationship itself. In Figure 1, the annotation contains one "context" anchor and one "content" anchor, and properties representing the annotation's id, author, and creation date.

5.1 Model Implications

Defining annotations in this way has several interesting implications. First, annotation objects are totally self-contained. This means that they can be stored completely separately from the resources that are referenced by the annotation's anchors. This is particularly important when resources are read-only, for instance when a user annotates a web page or a streaming video. It is also useful for sharing and filtering annotations in flexible ways. And it does not preclude storing annotations directly in one of the resources being annotated, for instance for portability.

CAF Annotations are also flexible and expressive linking structures. Figure 1 shows an annotation with one anchor pointing to context and another containing some literal content, but other configurations are possible too. For instance, if our application instead stored all digital ink in a separate xml file, the "content" anchor could reference the file by its URI and use an XPointer statement to identify the appropriate block of digital ink. In this

case, both the "context" and "content" anchors would refer to resources and neither would store literal content. This is what the Xlink specification refers to as "third-party links" [9]. In yet another configuration, the annotation may be stored inline among document markup, as in Figure 4, in which case the "context" anchor could contain the literal text of the paragraph being annotated. Here, both anchors would store literal content.

Related to the linking character of CAF annotations is the fact that "by-reference" anchors support annotation of any part of any resource with any part of any other resource, regardless of the resources' media types. This is because each "by-reference" anchor can capture media type-specific position information for the resource it refers to. Thus the core model can support annotating video with audio, text with digital ink, audio with text, and so on.

Another significant implication is that a single CAF annotation may contain any number of anchors. For instance, anchoring a single annotation to more than one place in a document, or anchoring across multiple documents, can be achieved by including one "context" anchor in the annotation for each resource and/or intra-resource position. Storing multiple foreign-language translations of a user's comment can be achieved by including one "content" anchor for each translation.

Finally, the core logical model maintains a strict distinction between "annotations", which represent the annotation relationship, and "resources". This makes the model recursive in two distinct ways. First, the annotation relationship can be annotated. This is because CAF annotations are themselves addressable (and are thus resources). A 'reply' item in an in-context threaded discussion is an example of an annotation on an annotation relationship, since the reply pertains to both its parent and its parent's context. Secondly, "annotations are annotatable." That is, that which we commonly think of as an annotation, for example the handwritten note in Figure 1, can be annotated, even if it is stored within an annotation's "content" anchor. This is because CAF regards it as a resource, and thus by definition is must be independently addressable. A spelling correction made to the handwritten note in Figure 1 would be an example of this kind of recursion, since the spelling correction does not depend on the handwritten comment being displayed in the context of the text document.

5.2 Model Tradeoffs

We were faced with several tradeoffs in the design of the core CAF logical model. To keep the model simple, there are a number of interesting and valuable features that it does not support. For instance, flexible methods for grouping anchors and annotations are missing, and are described as extensions to the core later in the paper.

On the other hand, to retain expressive power in the model, we chose to allow annotations to contain any number of anchors that point at or contain any kind of resources. Obviously, this can lead to complex and ambiguous annotation relationships unless the core model is "subtyped" to specify more constrained types of annotations, such as "text-on-text" annotations that only support text annotation of text documents.

6 AML-Core Schema

In this section we describe the core CAF logical model and its tradeoffs in formal detail with Annotation Markup Language, the AML-Core schema. It is of central importance in the Common Annotation Framework, and can serve as a *lingua franca* for annotations on the Web. It is the result of a highly iterative process in which we composed a schema draft, circulated it among the groups with whom we have been collaborating, considered additional annotation scenarios and gathered input, revised the draft schema, and repeated. A full listing of the schema can be found in the Appendix.

The schema is quite simple. It defines <annotation> elements to represent CAF annotations, and <anchor> sub-elements to represent anchors. The <annotation> element governs the relationship among the <anchor> sub-elements it contains, and <anchor> sub-elements hold literal resource content or a resource URI plus positioning

data. <annotation> elements are the only top-level elements defined in the schema, and since the salient characteristics of resources are captured within <anchor> sub-elements, no additional <resource> sub-element is necessary. Figure 2 shows a summary of AML-Core structure.

```
annotation::=id anchor+ [author] [createdate] [property]+
anchor    ::=id label {{uri [position]}|{[cid] [content]}} [property]+
position  ::=position_data [format] [code_base]
content   ::=content_data [format] [code_base]
property  ::=name property_data [format] [code_base]
label     ::={"content"|"context"|*}
```

Figure 2: Abbreviated Baucus Naur Form (BNF) for AML-Core schema constituents.

In addition to being simple, AML-Core is XLink-compatible [9]. <annotation> elements encapsulate the properties of Xlink extended links, and <anchor> sub-elements encapsulate the properties of both the Xlink 'locator' and 'resource' element types. <annotation> elements also contain an <arc> sub-element which by default relates all "context" <anchor> elements to all "content" <anchor> elements. Thus, any application that understands Xlink links automatically understands the basic structure and function of AML-Core <annotation> elements.

```
<aml:annotation id="guid:EA35E400-09C5-4ae5-ADE6-8BD799D23121"
                author="John Doe"
                createdate="2001-11-13T08:15:00.000-08:00">
      <aml:anchor id="int:0"
                  label="context"
                  href="http://delivery.acm.org/10.1145/370000/365117/p285-
                     bernheim_brush.htm?key1=365117&key2=365117&CFID=
                     647829&CFTOKEN=16044492"
                  format="urn:robustTextAnchor"
                  codebase="http://www.johndoe.com/robustTxtAnchor.dll">
            {In margin, next to 6th paragraph}
            1521;4:4:4:77;problem;orphaning;appear;
            revision;weoijasdcllllaweroiasdflk;0;0;0;77777
      </aml:anchor>
      <aml:anchor id="int:1"
                  label="content"
                  cid="guid:EA33A5BF-0911-2201-AFB7-998B7AD23121"
                  format="urn:digital_ink"
                  codebase="http://www.johndoe.com/digitalInk.dll">
            123;345;789;234;567;1717;37;47;646;7838;5;
            2311;429;2272;839;3373;9005;94940;383;721...
      </aml:anchor>
</aml:annotation>
```

Figure 3: Annotation from Figure 1 expressed in AML-core. This annotation is completely self-contained, and can be stored separately from the document being annotated.

Figure 3 shows one way to express the annotation from Figure 1 according to the AML-Core schema. Here, an <annotation> element contains one "context" <anchor> and one "content" <anchor>, and its attributes record its id, author, and createdate. The "context" <anchor> is an XLink 'locator' (since it contains an href attribute), and records the URI of the document being annotated. Its content is interpreted as position data describing the place in

the document where the annotation belongs. Its format attribute identifies what kind of positional data is stored (in this case, "robustTextAnchor" data for robustly anchoring the annotation to text [4]), and its codebase attribute identifies the URL for a code module that knows how to handle the position data.

Contrast this with the "content" <anchor>. The format and codebase attributes still describe the structure and location of handling code for the data stored within the element. However, this <anchor> is an Xlink 'resource' element (since the href attribute is absent), and the data stored within it is interpreted not as positional data, but rather as literal content. In addition, an optional 'cid' attribute holds a URI that can be used to identify the literal content independently of the "content" <anchor> in which it is stored.

Together, the <annotation> and its constituent <anchor> elements in Figure 3 form a self-contained unit that can be stored separately from the document being annotated. Self-sufficiency may be inefficient, however, when annotations are stored within a document for portability or offline operation, for instance, so AML-Core also supports a variety of in-document configurations. If the annotation in Figure 3 were stored in-document in a "data island"-style block, for instance, we could simply leave the "context" <anchor>'s href attribute blank (but leave the positional data intact), and it would be clear that the annotation applies to the document in which it is stored (as per RFC 2396 [3]). Alternatively, if we wanted to store the annotation inline in document markup like an HTML <a> tag, we could convert the "context" <anchor> to an Xlink 'resource' element (by leaving out the href attribute and dropping the positional data) and use it to wrap the text in the document being annotated. This is illustrated in Figure 4.

```
<p>...and were presented at the bottom of the document.</p>
<aml:annotation id="guid:EA35E400-09C5-4ae5-ADE6-8BD799D23121">
      <aml:anchor id="int:0"
                  label="context"
                  format=”text/html”>
            <p>The problem of orphaning is unique to annotations on
            digital/online documents, as paper-based ... of the document
            across modifications will become crucial.</p>
      </aml:anchor>
      <aml:anchor id="int:1"
                  label="content"
                  cid="guid:EA33A5BF-0911-2201-AFB7-998B7AD23121"
                  format=”urn:digital_ink”
                  codebase=”http://www.johndoe.com/digitalInk.dll”>
            123;345;789;234;567;1717;37;47;646;7838;5;
            2311;429;2272;839;3373;9005;94940;383;721...
      </aml:anchor>
</aml:annotation>
<p>But correctly positioning annotations in a revised...</p>
```

Figure 4: The annotation from (Figure 1) expressed in AML-Core, and configured to be stored directly inline among the annotated document's markup. The paragraph being annotated is wrapped by the AML-Core markup.

6.1 Schema Tradeoffs

We made a number of tradeoffs in the design of AML-Core to keep it as flexible and extensible as it is. For instance, we refrained from defining anchoring algorithms as part of CAF, and we avoided defining positional data formats as part of the core schema. Working these details out is undoubtedly necessary, and having them already worked out for some set of media types would have added a lot of value to CAF. However, we felt it

would unduly constrain the framework if we fully specified the methods by which annotations can be anchored, since it would preclude developing new methods. Instead, we designed <anchor> elements that can contain any kind of positional data and can support anchoring algorithms with their format and codebase attributes. This flexibility has paid off, and several anchoring algorithms and positional formats have been defined elsewhere [4].

7 Extensibility

Because we deliberately left things out of the AML-Core schema, we established conventions for extending CAF. It is via extension that new anchoring algorithms and positional data formats can be supported, annotations can be "sub-typed" to constrain their structure, and new abstractions such as composite anchors and annotation sets can be added to the logical model.

7.1 Anchoring

We have already seen an example of a new anchoring scheme. In Figure 3, the "context" <anchor> element contains rather inscrutable positional data that was generated by a robust text anchoring algorithm. This data records keywords, character offsets, and other features from the paragraph to which the handwritten comment corresponds, and it can be used by a re-positioning algorithm to correctly place the handwritten comment back in the document, even if the document gets reflowed or modified [4]. The <anchor>'s format attribute provides a processing hint to applications by labeling the element's content as "robustTextAnchor" data. If an application does not know how to parse "robustTextAnchor" data, it can use the <anchor>'s codebase attribute to download a code module that encapsulates the anchoring algorithm that produces and consumes this data.

The contents of <anchor> elements are unconstrained, so applications are free to store whatever kind of data supports the algorithms they define. In fact, this same basic mechanism is used in "content" <anchor> elements to support the user interface. In Figure 3, for instance, digital ink data is stored in the "content" <anchor>, and the codebase URI references a user interface module that can display and support interaction with the ink.

7.2 Sub-Typing

A second CAF extensibility convention is sub-typing. Sub-typing is a useful way to specify the structural constraints for a task-specific annotation type, and it is achieved through XML Schema inheritance. Sub-typing allows applications to produce and consume CAF data without having to handle all the arbitrarily complex annotation structures that CAF makes possible. For instance, we may want to define an "audio-on-video"-type annotation for recording narration on a television program. In this case the <annotation> should have one "context" <anchor> that can anchor the annotation to a segment of the program's timeline, and one "content" <anchor> that stores the audio narration for that segment. The AML-AOV schema is included in the Appendix to illustrate the example.

7.3 New Abstractions

Another important way that CAF can be extended is through adding new objects to the logical model. New abstractions can be added simply by inheriting from the AML-Core schema.

For example, grouping <anchor> elements together into "composite anchors" can allow us to specify more structure in an annotation relationship than we could with just the "context" and "content" distinction. We can use them to group redundant "context" <anchor>s, for instance, where each <anchor> supports a different method for positioning the annotation in a resource; or we could use them to group multiple "content" <anchor>s, each of which represents a different translation of a user's comment.

Figure 5 shows the basic structure of composite anchors as we propose them. The 'order' attribute specifies how the <anchor> elements stored within a composite anchor are to be used. "all" indicates that all of the <anchor>s must be used, "any" means the application can choose any one of the <anchor>, and "sequential" means application must use each <anchor> specified in the order they are listed. The order attribute can also be any other string value an application wishes to define, and composite anchors can contain other composite anchors. We include the AML-CA schema in the Appendix to formally define composite anchors.

```
composite_anchor   ::=  order {anchor_id | composite_anchor}+
order              ::=  {all|any|sequential|*}
anchor_id          ::=  UTF-8 encoded anchor id
```

Figure 5: BNF for AML-CA schema constituents.

The annotation "set" is another useful abstraction we may want to define. Collecting annotations into groups can be used to organize them and control sharing [1]. As we propose them, sets can contain annotations and other sets, and each set has a friendly name in addition to an id. Figure 6 shows the basic structure of sets as we propose them, and we include the AML-Set schema in the Appendix.

```
set            ::=  id name [parents] [children] [createdate] [property]+
parents        ::=  UTF-8 encoded, space-delimited list of id's.
children       ::=  UTF-8 encoded, space-delimited list of id's.
```

Figure 6: BNF for AML-Set schema constituents.

8 Applications

To test and validate the proposed AML-Core schema and CAF extensibility conventions, we used them to build a web page annotation client that plugs into Microsoft Internet Explorer, called WebAnn. The WebAnn user interface is illustrated in Figure 7. It has been deployed to support in-context threaded discussions around research papers in a graduate course at the University of Washington [5], and it has served as a test bed for experimenting with robust anchoring algorithms [4].

WebAnn users can see others' annotations on any web page. They can create their own highlights and text notes simply by selecting some text in the page and choosing the type of annotation they want to create from an in-place menu. Alternatively, they can add "summary" comments that apply to the entire page, or they can "reply" to existing annotations. Annotations can be marked public or private, and all annotations are displayed in an index to the left of the page. Regular annotations (those which are not summary comments or replies) are also displayed directly in the web page. Each user is assigned a separate color, in which all of their annotations are displayed. And all annotations are stored separately from the page, either in a local cache on the user's machine, or in a shared Internet annotation server.

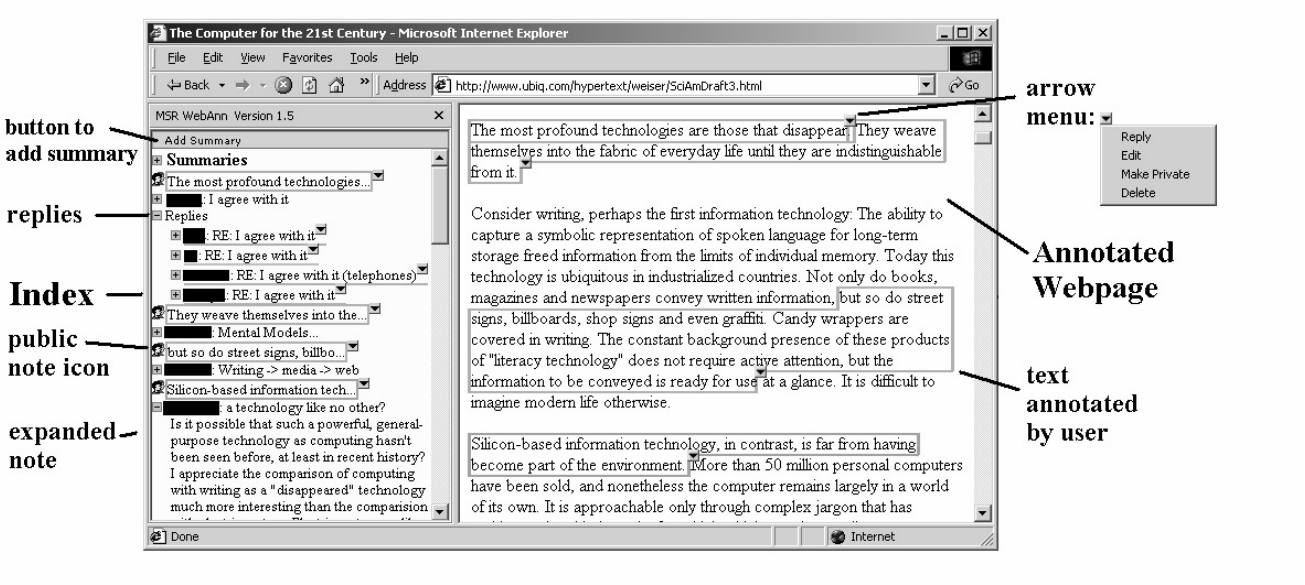


Figure 7: The CAF-based WebAnn tool. User names have been covered.

The CAF core supported all of the features we included in WebAnn, with three interesting exceptions. First, we defined and tested several different anchoring algorithms and positional data formats for our "context" <anchor> elements, and we settled on Keyword Anchoring [4].

Second, there is no distinction between public and private annotations in AML-Core, so we included a <property> element in each <annotation> to encode this value.

And third, AML-Core provides no way to record the color in which an annotation should be displayed. Since color corresponds to author, storing a separate <property> element in every <annotation> to encode color would have been inefficient. Instead, we annotated the class with a color for each user. That is, we invented a simple anchoring algorithm and positional data format to anchor annotations to an abstract “CSE 510” entity representing the graduate course in which all of the users were enrolled. Then, when each user started WebAnn for the first time, WebAnn automatically chose a color for the user and annotated “CSE 510” with it. We used these annotations behind the scenes to control the display.

8.1 Object Model

As we developed and deployed the WebAnn client we found it very helpful to have a common software layer that encapsulates CAF concepts. The API layer we developed includes objects corresponding to the core CAF logical model, an "annotation store" object that supports a flexible storage-independent query and filter syntax, and a software framework for coordination of anchoring algorithms and storage modules. We also implemented two annotation stores to which the API layer could connect, including a local store based on Microsoft SQL Server for WinCE, and a shared Internet annotation server based on Microsoft SQL Server. Our software architecture is illustrated in Figure 8.

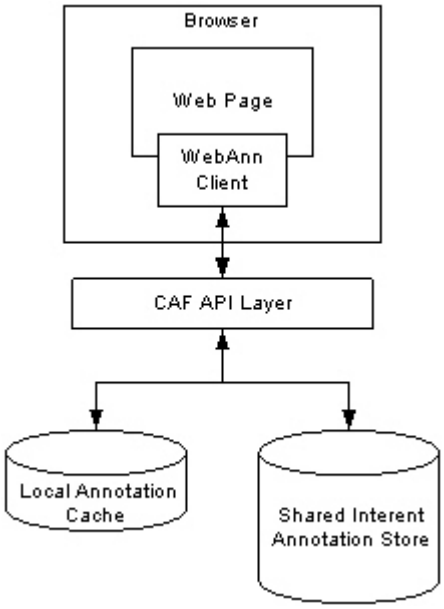


Figure 8: The CAF-based WebAnn tool. User names have been covered.

WebAnn and other applications currently in development have all been based on our API layer with great success. Indeed, in the future it may be useful to establish a standard software layer to support CAF much like the Document Object Model supports HTML. Such a layer could further normalize CAF extensibility conventions, for instance by establishing standard mechanisms for pluggable anchoring, UI, and storage handling modules. It could also support a standard query syntax for retrieving and filtering annotations, such as XML Query [16]. And it could support facilities which the CAF core intentionally avoids, for instance access control, scalable storage, and data exchange protocols.

9 Related Work

There is a broad range of work that has been done on Web-based annotation, though to our knowledge no one has attempted to establish a single standard for annotations which can support the wide spectrum of activities that "annotation" represents. Roughly, the work that has been done falls into four categories, including existing standards, proposed architectures, research systems, and commercial systems.

9.1 Standards

Several technology standards touch on annotation or can be considered to support annotation in one way or another. The most closely related to CAF is Xlink. Xlink sets forth standard abstractions and attributes that can be used to structure linking in schemas [9]. Xlink in and of itself does not provide sufficient structure to support annotations, however we used the abstractions it defines forming the core CAF logical model, and we employ the attributes it defines in the AML-Core schema. As a result, CAF can be regarded as an application of Xlink to the annotation domain.

Another standard which by itself does not constitute a full-fledged annotation system is XPointer. XPointer specifies a standard for pointing to nodes within XML documents [8]. In the CAF context, XPointer comprises a flexible and reasonably robust positional data format for anchoring annotations to XML. Thus it can be regarded as an enabling technology for CAF.

Several efforts have attempted to establish standard ontologies for describing resource characteristics, including MPEG-7 [18] and the Dublin Core Initiative [10]. While CAF is a framework for annotating resources, and in particular could be used for storing metadata descriptions of resources, it does not attempt to define a metadata ontology with which resources should be annotated. CAF is similar to RDF in this respect, and in fact can be viewed as a generalization of RDF.

RDF is a framework that supports the use of metadata ontologies to describe resources in standard ways. Its main goals are to facilitate machine understanding and automated processing of resources [15], however some have also used it to support annotation [13]. CAF and RDF are in fact similar, but RDF imposes some fundamental restrictions which limit it as a platform for annotations: First, RDF embeds directionality by assuming properties belong to web resources, and there is no way to modify the underlying "property of" relationship that an RDF property has with a web resource. Second, intra-resource locations can only be expressed as URI extensions (e.g. by including data after the #). This is fine where standardized methods such as XPointer exist, however the URI extension mechanism is too otherwise unstructured. CAF provides a well-defined way to include arbitrary positional data in <anchor> elements and describe it with the <anchor> element's format and codebase attributes. Third, while RDF provides facilities for grouping resources and literals, it does not support addressing individual members of a collection for purposes of anchoring. That is, while a single RDF property may refer globally to every member of a collection of resources, there is no way to express how the property refers individually to each member of the collection. This makes it impossible to anchor a single annotation to multiple places in a document, for instance, which is a scenario AML-Core was designed to handle.

9.2 Proposed Architectures

Previously proposed architectures for the support of annotations on the Web have mostly focused on scalability, access control, and other basic system issues. Laliberte and Braverman propose a scalable protocol for annotation retrieval and discovery, for example, but do not specify what annotations are, what data they can contain, how they are anchored to a resource, etc. [14]. Likewise Schickler et al focus on network architecture and data delivery issues for an annotation system without going into detail about the structure and capabilities of the annotations themselves [23]. And Roscheisen et al discuss the ComMentor system's architecture and annotation operations (such as adding, deleting, and retrieving annotations) at great length. They include a brief definition of their "PRDMitem" annotation structure, which bears some resemblance to the CAF <annotation> element, but the system is focused on annotating text documents and does not address a wider class of annotation activities [22].

9.3 Research Systems

A wide variety of research systems have been developed over the past decade to experiment with various annotation features, and some have claimed to establish standards for annotations on the web. The W3C Annotea system, for example, is an RDF-based annotation system. It employs Xlink to some extent, it uses XPointer for anchoring, and it supports text annotations on text web pages [13]. But because it relies on RDF, it requires each annotation to be classified according to a type hierarchy as either a comment, a change, an example, a question, etc. This makes the system overly complex for support of end-user annotation, since many such annotations are personal and transient in nature and may not retain their meaning for very long (even to the author) [17], and this makes classifying them an unnecessary burden. CAF makes it possible to classify annotations via extension, but it does not require it as part of the core CAF model.

Other systems support text annotation of text web pages with various more constrained approaches. The CoNote system exploits HTML to allow annotations at predefined positions on a web page [7]. The CritLink system also exploits HTML for anchoring, but allows annotations to be created anywhere on a web page [6]. These approaches are somewhat similar to the special case of storing CAF <annotation> elements inline in document markup. The Annotator [20] and WebVise [12] systems support more flexible non-HTML methods for anchoring

their annotation items, however the annotations they support have a constrained structure. None of these systems offer a model as powerful, flexible, and extensible as CAF for annotating diverse digital resources.

9.4 Commercial Systems

Finally, several widely-available commercial systems have popularized web-based annotation. ThirdVoice, now out of business, pioneered this effort with a system that allowed users all over the world to author and share annotations on web pages [24]. E-Quill supports web-based collaboration with high-quality graphics [11]. The PageSeeder system is aimed at encouraging community around web page content, and displays threaded discussions inline in the annotated web page [21]. And Microsoft Office Web Discussions supports inline threaded discussion on web pages among small to medium sized work groups [19]. However, none of these systems has been able to establish a de facto standard for annotation structure and function, and the user experience of authoring and viewing annotations varies dramatically. CAF is a proposed standard and could facilitate a more uniform user experience for annotations on the Web.

10 Concluding Remarks

A single, foundational standard for annotation structure and function can be a significant step forward for annotations on the Web. To this end we propose a simple, extensible, storage-neutral Common Annotation Framework that is capable of annotating any resource. CAF is standards-based, and supports many annotation scenarios that other annotation-related standards cannot.

Throughout the course of our work on CAF, we have heard time and again how valuable it would be to establish a *lingua franca* for annotations. A common programming model, application interoperability, and a consistent user experience are just a few of the advantages such a framework can bring, and the time is right to agree upon such a foundation. That said, our work is just beginning. It is the hope of the authors to gather input from the research community on the value of the proposed CAF standard.

11 Acknowledgements

Thanks to our many colleagues who provided valuable input into the design of CAF, including Scott Cottrille, Andy van Dam, Steve DeRose, Bert Keely, Axel Kramer, Butler Lampson, Vikram Madan, Jonathan Marsh, Mike Morton, Darryl Rubin, Sam Sengupta, Susi Strom, Steve Weil, and Yoram Yaacovi.

12 Appendix

In this section we present the proposed CAF core schema AML-Core. We also present extensions to AML-Core, including AML-CA for support of composite anchoring, and AML-Set for support of annotation grouping.

12.1 AML-Core Schema

AML-Core is the CAF core schema. It encapsulates the structure, functionality, and interrelationships described in the CAF logical model. Note that simple types such as 'date', 'format', and 'codebase' alias built-in XML Schema simple types so that they may be extended in a consistent manner. Also note that <aml:annotation> elements are constrained to have unique ids within any document instance, and <aml:anchor> elements must have unique ids within the scope of the annotation to which they belong.

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="aml-core"
            xmlns:xsd="http://www.w3c.org/2001/XMLSchema"
```

```
      xmlns:xlink=”http://www.w3.org/1999/xlink”
      xmlns:aml=”urn:aml-core”>

<xsd:element name="annotation" type=”aml:AnnType” minOccurs="0"
maxOccurs=“unbounded”/>
<xsd:complexType name=”AnnType” mixed=”false”>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:anyAttribute processContents="lax"/>
      <xsd:attribute type=”xlink:type” fixed=”extended”/>
      <xsd:attribute type="aml:id" use=”required”/>
      <xsd:attribute type="aml:author" use=”optional”/>
      <xsd:attribute name=”createdate” type="aml:date" use=”optional”/>
      <xsd:element type=”aml:arc” minOccurs=”1” maxOccurs=”1”/>
      <xsd:element type="aml:anchor" minOccurs="1" maxOccurs=“unbounded”/>
      <xsd:element type="aml:property" minOccurs="0" maxOccurs=“unbounded”/>
</xsd:complexType>

<xsd:complexType name=”arc”>
      <xsd:complexType>
            <xsd:complexContent>
                  <xsd:restriction base=”xsd:anyType”>
                        <xsd:attribute type=”xlink:type” fixed=”arc”/>
                        <xsd:attribute type=”xlink:from” default=”context”
                           use=”optional”/>
                        <xsd:attribute type=”xlink:to” default=”content”
                           use=”optional”/>
                        <xsd:attribute type=”xlink:arcrole” default=”annotates”
                           use=”optional”/>
                        <xsd:attribute type=”xlink:role” default=”annotation”
                           use=”optional”/>
                        <xsd:attribute type=”xlink:title” default=”annotation”
                           use=”optional”/>
                        <xsd:attribute type=”xlink:show” default=”default”
                           use=”optional”/>
                        <xsd:attribute type=”xlink:actuate” default=”default”
                           use=”optional”/>
                  </xsd:restriction>
            </xsd:complexContent>
      </xsd:complexType>
</xsd:complexType>

<xsd:complexType name=”anchor” mixed=”true”>
      <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:anyAttribute processContents="lax"/>
      <xsd:choice>
            <xsd:group>
                  <xsd:attribute type=”xlink:type” fixed=”resource”/>
                  <xsd:attribute name="cid" type="aml:id" use="optional"/>
                  <xsd:attribute type=”xlink:href” use=”prohibited”/>
            </xsd:group>
            <xsd:group>
                  <xsd:attribute type=”xlink:type” fixed=”locator”/>
                  <xsd:attribute type=”xlink:href” use=”required”/>
            </xsd:group>
      </xsd:choice>
      <xsd:attribute type="aml:id" use=”required”/>
      <xsd:attribute type=”xlink:label” use=”required”/>
```

```
            <xsd:attribute type="aml:format" use=”optional”/>
            <xsd:attribute type="aml:codebase" use=”optional”/>
            <xsd:element type="aml:property" minOccurs="0" maxOccurs=“unbounded”/>
      </xsd:complexType>

      <xsd:complexType name="property" mixed=”true” >
            <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:anyAttribute processContents="lax"/>
            <xsd:attribute name="aml:name" use=”required”/>
            <xsd:attribute type="aml:format" use=”optional”/>
            <xsd:attribute type="aml:codebase" use=”optional”/>
      </complexType>

      <xsd:simpleType name=”date” type=”xsd:dateTime”>
      </xsd:simpleType>

      <xsd:simpleType name=”author” type=”xsd:string”>
      </xsd:simpleType>

      <xsd:simpleType name=”format” type=”xsd:anyURI”>
      </xsd:simpleType>

      <xsd:simpleType name=”codebase” type=”xsd:anyURI”>
      </xsd:simpleType>

      <xsd:simpleType name=”name” type=”xsd:string”>
      </xsd:simpleType>

      <xsd:simpleType name=”id” type=”xsd:anyURI”>
      </xsd:simpleType>

      <xsd:unique name=”uniqueAnchorIdsInsideAnnotations”>
            <xsd:selector xpath=”aml:annotation/aml:anchor”/>
            <xsd:field xpath=”@id”/>
      </xsd:unique>

      <xsd:unique name=”uniqueAnnotationIds”>
            <xsd:selector xpath=”.//aml:annotation”/>
            <xsd:field xpath=”@id”/>
      </xsd:unique>

</xsd:schema>
```

12.2 AML-AOV Schema

The AML-AOV "audio-on-video" schema defines a specific sub-type of the AML-Core <annotation> element for the purpose of annotating video with audio. It is provided to illustrate how the core CAF annotation can be sub-typed to support specific tasks.

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="aml-aov"
            xmlns:xsd="http://www.w3c.org/2001/XMLSchema"
            xmlns:aml=”urn:aml-core”
            xmlns:xlink=”http://www.w3.org/1999/xlink”
            xmlns:aov=”urn:aml-aov”>
```

```
<xsd:element name="annotation" type=”aov:AoVType” minOccurs="0"
maxOccurs=“unbounded”/>
<xsd:complexType name=”AoVType” mixed=”false”>
<xsd:complexContent>
        <xsd:extension base=”aov:restrictedAnn”>
               <xsd:element type="aov:audioContent" minOccurs="1"
               maxOccurs=“1”/>
               <xsd:element type="aov:videoContext" minOccurs="1"
               maxOccurs=“1”/>
               </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name=”restrictedAnn” mixed=”false”>
<xsd:complexContent>
        <xsd:restriction base=”aml:annotation”>
                  <xsd:attribute type=”xlink:type” fixed=”extended”/>
                  <xsd:attribute type="aml:id" use=”required”/>
                  <xsd:attribute type="aml:author" use=”optional”/>
                  <xsd:attribute name=”createdate” type="aml:date"
                  use=”optional”/>
                  <xsd:element type=”aml:arc” minOccurs=”1” maxOccurs=”1”/>
               </xsd:restriction>
        </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name=”audioContent” mixed=”true”>
<xsd:complexContent>
        <xsd:restriction base=”aml:anchor”>
                  <xsd:attribute type=”xlink:type” fixed=”resource”/>
                  <xsd:attribute name="cid" type="aml:id" use="optional"/>
                  <xsd:attribute type="aml:id" use=”required”/>
                  <xsd:attribute type=”xlink:label” fixed=”content”/>
                  <xsd:attribute type="aml:format" fixed=”binary/wav”/>
                  <xsd:attribute type="aml:codebase" use=”optional”/>
               </xsd:restriction>
        </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name=”videoContext” mixed=”false”>
<xsd:complexContent>
        <xsd:extension base=”aov:cxtAnchor”>
               <xsd:element name=”beginTime" type=”aov:videoTime”
               minOccurs="1" maxOccurs=“1”/>
               <xsd:element name=”endTime" type=”aov:videoTime”
               minOccurs="1" maxOccurs=“1”/>
               </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name=”cxtAnchor” mixed=”true”>
<xsd:complexContent>
        <xsd:restriction base=”aml:anchor”>
                  <xsd:attribute type=”xlink:type” fixed=”locator”/>
                  <xsd:attribute type=”xlink:href” use=”required”/>
                  <xsd:attribute type="aml:id" use=”required”/>
                  <xsd:attribute type=”xlink:label” fixed=”context”/>
```

```
                        <xsd:attribute type="aml:format" fixed=”text/xml”/>
                        <xsd:attribute type="aml:codebase" use=”optional”/>
                    </xsd:restriction>
            </xsd:complexContent>
      </xsd:complexType>

      <xsd:complexType name=”videoTime” mixed=”false”>
      <xsd:complexContent>
            <xsd:restriction base=”xsd:anyType”>
                        <xsd:attribute type=”aov:ms” use=”required”/>
                  </xsd:restriction>
            </xsd:complexContent>
      </xsd:complexType>

      <xsd:simpleType name=”ms” type=”xsd:integer”>
      </xsd:simpleType>

</xsd:schema>
```

Below is an example of an "audio-on-video" annotation which associates an audio comment (represented in binary form) with a 30-second interval toward the beginning of a "Friends" TV episode:

```
<aov:annotation id=”int:0”>
      <aov:videoContext id=”int:0”
      href=”http://www.nbc.com/entertainment/shows/friends/13Oct2001.mpeg”>
            <aov:beginTime ms=”53453”/>
            <aov:endTime ms=”83453”/>
      </aov:videoContext>
      <aov:audioContent id=”int:1”>
            asodii23roaiw34r90acf9a03245
            wje4350w9j34rasdv890wefwd…
      </aov:audioContent>
</aov:annotation>
```

12.3 AML-CA Schema

The AML-CA composite anchoring schema extends the AML-Core schema by adding anchor grouping functionality. It is provided as both an example of how the core CAF schema can be extended, and as a proposed standard for anchor grouping. Note that the <companc> element has no specified Xlink meaning.

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="aml-ca"
            xmlns:xsd="http://www.w3c.org/2001/XMLSchema"
            xmlns:aml=”urn:aml-aml”
            xmlns:aca=”urn:aml-ca”>

      <xsd:element name="annotation" type=”aca:AnnType” minOccurs="0"
      maxOccurs=“unbounded”>
      <xsd:complexType name=”AnnType” mixed=”false”>
            <xsd:complexContent>
                  <xsd:extension base=”aml:amlAnnType”>
                        <element type=”aca:companc” minOccurs=”0”
                           maxOccurs=”unbounded”/>
                  </xsd:extension>
            </xsd:complexContent>
```

```
	</xsd:complexType>

	<xsd:complexType name="companc" mixed="false">
		<xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
		<xsd:anyAttribute processContents="lax"/>
		<xsd:attribute type="aca:order" default="any" use="optional"/>
		<xsd:element type="aca:anchorid" minOccurs="0" maxOccurs="unbounded"/>
		<xsd:element type="aca:companc" minOccurs="0" maxOccurs="unbounded"/>
		<xsd:element type="aml:property" minOccurs="0" maxOccurs="unbounded"/>
	</xsd:complexType>

	<xsd:complexType name="anchorid">
		<xsd:complexType>
			<xsd:complexContent>
				<xsd:restriction base="xsd:anyType">
					<xsd:attribute type="aml:id" use="required"/>
				</xsd:restriction>
			</xsd:complexContent>
		</xsd:complexType>
	</xsd:complextType>

	<xsd:simpleType name="order">
		<xsd:union memberTypes="aca:enumOrder xsd:string"/>
	</xsd:simpleType>

	<xsd:simpleType name="enumOrder">
		<xsd:restriction base="xsd:string">
			<xsd:enumeration value="any"/>
			<xsd:enumeration value="all"/>
			<xsd:enumeration value="sequential"/>
		</xsd:restriction>
	</xsd:simpleType>

</xsd:schema>
```

Below is an example of how the AML-CA composite anchoring schema can be used to group "context" <anchor> elements within an <annotation>. The first context anchor contains anchor data for a fast but fragile anchoring algorithm, while the second contains data for a slow but robust algorithm. The <companc> element instructs applications to try the fast but fragile anchor first, then the slow but robust algorithm if that fails:

```
	<aca:annotation   xmlns:aca="urn:aml-ca"
			  xmlns:aml="urn:aml-core">
		<aml:anchor id="0" label="context"
			  href= http://www.foo.com/info.htm
			  format="fragileButFastBookmark">
			234;kj23dfoiuoi2348979ds8fvasdaw23434534;;;we3;
		</aml:anchor>
		<aml:anchor id="1" label="context"
			  href= http://www.foo.com/info.htm
			  format="slowButRobust">
			332;234;;;234;233;879089;;dog;fox;purple;45;;345;00;0;0
		</aml:anchor>
		<aml:anchor id="2" label="content" format="text/plain">
			what does this sentence mean?
		</aml:anchor>
		<aca:companc order="sequential">
```

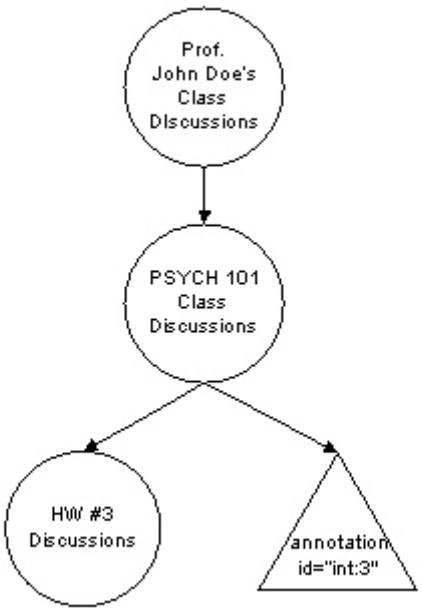
```
                <aca:anchorid id=”0”/>
                <aca:anchorid id=”1”/>
          </aca:companc>
    </aca:annotation>
```

12.4 AML-Set Schema

AML-Set extends the AML-CA schema by adding annotation grouping functionality. Like AML-CA, it is provided as both example and proposed standard, and the <set> element has no specified Xlink meaning.

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="aml-set"
            xmlns:xsd="http://www.w3c.org/2001/XMLSchema"
            xmlns:aca=”urn:aml-ca”
            xmlns:amls=”urn:aml-set”>

      <xsd:element name="set" type="amls:setType" minOccurs="0"
      maxOccurs="unbounded"/>
      <xsd:complexType name="set" mixed=”false”>
            <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:anyAttribute processContents="lax"/>
            <xsd:attribute name="id" type="aml:id" use=”required”/>
            <xsd:attribute name="parents" type="amls:idrefs" use=”optional”/>
            <xsd:attribute name="children" type="amls:idrefs" use=”optional”/>
            <xsd:attribute name=”createdate” type="aml:date" use=”optional”/>
            <xsd:attribute type="aml:name" use=”required”/>
            <xsd:element type="aml:property" minOccurs="0" maxOccurs=“unbounded”/>
      </xsd:complexType>

      <xsd:element name="annotation" type=”amls:AnnType” minOccurs="0"
      maxOccurs=“unbounded”>
      <xsd:complexType name=”AnnType” mixed=”false”>
            <xsd:complexContent>
                  <xsd:extension base=”aca:AnnType”>
                        <attribute name="sets" type="amls:idrefs" use=”optional”/>
                  </xsd:extension>
            </xsd:complexContent>
      </xsd:complexType>

      <xsd:simpleType name=”idrefs”>
            <xsd:list itemType=”aml:id”/>
      </xsd:simpleType>

</xsd:schema>
```

Below is an example of how AML-Set can be used to group <annotation> and <set> elements into a meaningful organizational hierarchy. The diagram on the left represents how Professor John Doe's class discussions are organized. Circles represent annotation sets, and triangles represent annotations. The hierarchy is represented according to AML-Set on the right.




```
<s:aml-set xmlns:s="urn:aml-set">
      <s:set id="int:0"
             name="Prof. John Doe's
             Class Discussions"
             parents=""
             children="int:1">
      </s:set>
      <s:set id="int:1"
             name="PSYCH 101 Class
           Discussions"
             parents="int:0"
             children="int:2 int:3">
      </s:set>
      <s:set id="int:2"
             name="HW #3 Discussions"
             parents="int:1"
             chidren="">
      </s:set>
      <s:annotation id="int:3"
      sets="int:1">...</s:annotation>
</s:aml-set>
```

13 References

[1] Bargeron, D., Gupta, A., Grudin, J., and Sanocki, E., "Annotations for Streaming Video on the Web: System Design and Usage Studies." *Proceedings of the Eighth International World Wide Web Conference,* Toronto, May 11-14, 1999. http://www8.org/w8-papers/1b-multimedia/annotations/

[2] Bargeron, D., Grudin, J., Gupta, A., "General and Specific Interfaces: Experiences with a Multimedia Platform." *Microsoft Technical Report MSR-TR-2001-90*, October 2, 2001. http://www.research.microsoft.com/research/coet/MRAS/TRs/01-90.doc

[3] Berners-Lee, T., Fielding, R., Masinter, L., "Uniform Resource Identifiers (URI): Generic Syntax." *RFC 2396*, Internet Engineering Task Force, August 1998. http://www.ietf.org/rfc/rfc2396.txt

[4] Brush, A.J., and Bargeron, D., "Robustly Anchoring Annotations using Keywords." *Microsoft Technical Report MSR-TR-2001-107*, November 2001.

[5] Brush, A.J., Bargeron, D., Grudin, J., Borning, A., and Gupta, A., "Supporting Interaction Outside of Class: Anchored Discussions vs. Discussion Boards." *To Appear in Proceedings of Computer Support for Collaborative Learning 2002,* Boulder, CO, January 7-11, 2002. http://www.research.microsoft.com/research/coet/Annotations/CSCL02/paper.pdf

[6] CritLink, http://crit.org/http://crit.org/critlink.html

[7] Davis, and Huttonlocker. "CoNote System Overview." Cornell University, 1995. http://www.cs.cornell.edu/home/dph/annotation/annotations.html

[8] DeRose, S., Maler, E., and Daniel, R. (eds), "XML Pointer Language (XPointer) Version 1.0." *W3C Candidate Recommendation*, September 11, 2001. http://www.w3.org/TR/xptr/

[9] DeRose, S., Maler, E., and Orchard, D. (eds), "XML Linking Language (XLink) Version 1.0." *W3C Recommendation*, June 27, 2001. http://www.w3.org/TR/xlink/

[10] Dublin Core Metadata Initiative. http://dublincore.org/

[11] E-Quill. http://www.e-quill.com/

[12] Grønbæk, K., Sloth, L., and Ørbæk, P., "Webvise: Browser and Proxy Support for Open Hypermedia Structuring Mechanisms on The WWW." *Proceedings of the Eighth International World Wide Web Conference*, Toronto, May 1999. http://www8.org/w8-papers/3a-search-query/webvise/webvise.html

[13] Kahan, J., Koivunen, M.R., Prud'Hommeaux, E., and Swick, R.R. "Annotea: An Open RDF Infrastructure for Shared Web Annotations." *Proceedings of the 10th International World Wide Web Conference*, Hong Kong, May 1-5, 2001. http://www.iw3con.org/cdrom/papers/pdf/p488.pdf

[14] Laliberte, D., and Braverman, A., "A Protocol for Scalable Group and Public Annotations." *Proceedings of the Third International World Wide Web Conference,* Darmstadt, Germany, April 10-14, 1995. http://www.igd.fhg.de/archive/1995_www95/proceedings/papers/100/scalable-annotations.html

[15] Lassila, O., and Swick, R.R. (eds), "Resource Description Framework (RDF) Model and Syntax Specification." *W3C Recommendation*, February 22, 1999. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

[16] Malhotra, A., Robie, J., and Rys, M. (eds), "XML Syntax for XQuery 1.0 (XQueryX)". *W3C Working Draft 07*, June 2001. http://www.w3.org/TR/xqueryx

[17] Marshall, C.C., "Toward an ecology of hypertext annotation," *Proceedings of HyperText '98*, Pittsburgh, PA, June 1998. http://www.csdl.tamu.edu/~marshall/ht98-final.pdf

[18] Martinez, J.M. (ed), "Overview of the MPEG-7 Standard (version 5.0)." *ISO/IEC JTC1/SC29/WG11 N4031*, Singapore, March 2001. http://mpeg.telecomitalialab.com/standards/mpeg-7/mpeg-7.htm

[19] Microsoft Office 2000 Web Discussions. http://office.microsoft.com/assistance/2000/wWebDiscussions.aspx

[20] Ovsiannikov, I., Arbib, M., McNeill, T. "Annotation Technology." *International Journal of Human Computer Studies*, 1999, pp. 329-362.

[21] PageSeeder, http://ps.pageseeder.com/ps/topic/ps

[22] Roscheisen, M., Mogensen, C., Winograd, T. "Shared Web Annotations as a Platform for Third-Party Value-Added, Information Providers: Architecture, Protocols, and Usage Examples." *Technical Report CSDTR/DLTR*, Stanford University, 1997. http://www-diglib.stanford.edu/rmr/TR/TR.html

[23] Schickler, M.A., Mazer, M.S., and Brooks, C., "Pan-Browser Support for Annotations and Other Meta Information on the World Wide Web." *Proceedings of the Fifth International World Wide Web Conference*, Paris, France, May 1996. http://www5conf.inria.fr/fich_html/papers/P15/Overview.html.

[24] ThirdVoice Annotation System, http://www.thirdvoice.com