From the Selected Works of William Osei-Poku

Spring April 10, 2009

Encoded Archival Description

William Osei-Poku

Available at: https://works.bepress.com/raspino/3/
The current Encoded Archival Description (EAD) is an SGML based data structure with a set of rules which are used to designate intellectual and physical elements of archival finding aids. Specifically, it is an XML based data structure, a simplified format of SGML. Some of the benefits of the EAD are to enable effective searching, retrieval, display and exchange of the information contained in an archival aid. It also helps to standardize how archival information are represented and or presented in an electronic environment and overall, it enhances the searching and finding processes. This approach encourages a platform-independent or a system-independent structure which in turn allows the exchange of data with little or no modifications of mapping techniques. Many repositories in the US, England, Australia and other places have adopted and implemented EAD with varying degrees of sophistication (Implementors currently listed, 2007). They include universities like Harvard, Johns Hopkins, Brown and the University of Maryland, libraries like the Yale University library and the Durham University Library, archives like Utah State Archives, Glasgow University Archives, American Heritage Virtual Archives and Iowa Women’s Archives. The complete list of institutions which have adopted the EAD are found here: [http://www.archivists.org/ssaagroups/ead/implementors.html](http://www.archivists.org/ssaagroups/ead/implementors.html)

HISTORY OF THE EAD:
The EAD was started in 1993 at the University of California in Berkeley library with funds from the Department of Education and under the leadership of Daniel Pritti, the principal investigator of the project. Sometimes he is referred to as the founder of the EAD. The developers wanted to look into the development of a non-proprietary encoding standard that went beyond what was then available for machine-readable finding aids. The finding aids referred to in this case, include inventories, registers, indexes, and other documents created by archives, libraries, museums and manuscript repositories. The developers hoped that the EAD, when completed, would support the use of their holdings and also make it less strenuous to exchange or transfer data in different formats. Perhaps the success of the EAD could be attributed to the fact that the development of the EAD received enormous technical as well as intellectual support from the Society of American Archivists (SAA) and is currently maintained by the SAA in partnership with the Library of Congress. In fact, the Society of American Archivists (SAA) owns the intellectual component of the EAD, and the SAA EAD Working Group (EAD Roundtable) oversees its development (EAD – Encoded Archival Description, 2007). I think this was as an important decision on the part of the developers to involve these professionals earlier in the development of the EAD.

OBJECTIVES OF THE EAD:
As presented on the EAD website, the developers of the EAD aimed at achieving the following with a fully functioning EAD, among other things: 1. The ability to present extensive and interrelated descriptive information found in the archival finding aids. 2. The ability to preserve the hierarchical relationships existing between levels of description. 3. The ability to represent descriptive information that is inherited by one hierarchical level from another. 4. The ability to move within a hierarchical informational structure. 5. Support for element-specific indexing and retrieval.

STRUCTURE OF THE EAD:
Underlying the development of the EAD is the logical segmentation of a finding aid, which states that at the basic level, a finding aid consists of two segments:

1. A segment which describes the information about the finding aid – For example, compiler, compilation date, title etc. 2. A segment which describes information about the archival material itself - For example, collection, record group, series etc. The segment which provides information about the described material (#2 above) could further be sub-divided into: 1. A hierarchy or multi-level description of organized information which describes a unit of record(s) along with component parts or divisions and 2. Other associated information that do not specifically describe the record(s) but acts as a facilitator (e.g. bibliography) for researchers.

High Level Elements:
In relation to the segmentation above, the EAD DTD divides the first segment into two high-level elements known as EAD Header <eadheader> and Front Matter <frontmatter> (They are discussed below). The second segment, which describes information about the archival materials, is contained within the <archdesc> element which is a third high-level element (also described below). All three of these high-level elements are contained within the <ead> element which is the outermost element and also wraps around the entire document. (EAD – Encoded Archival Description, 2007).

As one studies these differences, it becomes clear that they are key to understanding an EAD document, schema or DTD. To some extent, these differences can help you understand how all these elements fit together in an EAD document.

As stated above, some of the aims of the EAD are to ensure that, at a minimum, machines will be able to read and represent archival information accurately and efficiently; that software will be able make the searching process more efficient so that identifying documents does not become a tedious process, and also be able to make the exchange or merger of data elements.
between platforms more efficient and as seamless as possible. But in order to accomplish these goals, EAD needed to either adopt some conventional languages, technologies, principles, formats and conventions or the developers had to be able to develop an improved method of achieving the above goals. The developers of the EAD chose to go with the former, they based the structure of the EAD on a well known Schema and DTD combination like other SGML and XML based data structures or content management software. SGML authoring and viewing software need such structured data elements in order to transform and display them efficiently.

In structuring the EAD, the developers took advantage of the characteristic of SGML, which allows you to define attributes and associate them with particular elements. The EAD has many (146) elements, some of which are required, implied or optional. For people like me who have not been familiar with the EAD until now, perhaps one important thing about the organization of the EAD which we would find most beneficial is the emphasis placed on comprehensive documentation of the EAD Schema and DTD complete with examples and crosswalks. Detailed explanation of elements in the tag library makes it easier for people who are less acquainted with the EAD to quickly learn to implement it. Such comprehensive documentation is lacking in many systems development projects and usually what happens is that when the group of developers leaves the project, no one else is able to continue the work started because the new developers would be limited by the lack of adequate documentation. In the next section, I will be turning my attention to some of the information provided in the well documented EAD tag library.

**EAD SCHEMA, DTD and TAG SET.**

The EAD Schema acts as a model language or an acceptable common language for describing the structure of data (Walsh, 1999). This encourages standardization and also makes it possible to transfer data from one system to the other as it provides the basis for the creation of a platform independent transfer or merger. It also enables the control of the structure of the different types of documents through the implementation of custom or general validation rules.

The current version of the EAD DTD released in December, 2002 was designed to function as both SGML and XML DTD with the capability to use switches to turn off or on, specific features peculiar to XML or SGML. The EAD DTD specifies elements that need to be used to describe a collection. It also provides rules on how the elements are arranged and what is permitted or not within a set of elements, for a specific implementation of XML. The tag set of the EAD contains 146 elements which can be used to describe a collection as a whole or it can be used to encode detailed multi-level, hierarchical inventory of a collection. It could also be mapped to other content formats or standards such as the Dublin Core, ISAD (G) and USMARC etc.

**EAD HEADER <eadheader>**

To ensure that the implementation of the EAD would encourage uniformity across platforms, the developers of the EAD modeled the <eadheader> on the header element of the Text Encoding Initiative (TEI). The <eadheader> has 4 sub elements with some of them further divided into additional sub elements: <eadid> (required), <filedesc> (required), <profiledesc> (optional), and <revisiondesc> (optional). These elements are defined below:

1. EAD Identifier <eadid> - this provides a unique identification number or code for the finding aid and can indicate the location, source, and type of the identifier.
2. File Description <filedesc> - contains much of the bibliographic information about the finding aid, including the name of the author, title, subtitle, and sponsor (all contained in the Title Statement <titlestmt>), as well as the edition, publisher, series, and related notes encoded separately.
3. Profile Description <profiledesc> is used to record the language of the finding aid and information about who created the encoded version of the document, and when.
4. Revision Description <revisiondesc> summarizes any revisions made to the EAD document.

The <eadheader> which is a required element in an EAD document is also a wrapper element for bibliographic and descriptive information about the finding aid document rather than the archival materials being described. According the EAD website, "Wrapper element" indicates an element that cannot contain text directly; a second, nested element must be opened first. The 4 elements of the <eadheader> and their sub-elements provide a unique identification code for the finding aid. The <eadheader> element must be followed in a prescribed order in order to ensure uniformity across finding aids. Because of this, the <eadheader> and its child elements can be mapped to other standards for easy interchange of information. They are often mapped to the Dublin Core elements such as Author, Language and Creator. For example, in the example below the relatedencoding="DC" attribute of the eadheader element specifies that child elements will be mapped to Dublin Core; the child element <author encodinganalogue="Creator"> indicates that the EAD element <author> maps to the Dublin Core element <creator>. Here is an example of a portion of an EAD document which shows that it is being mapped to the Dublin Core:

```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE EAD PUBLIC "+//ISBN 1-931666-00-8//DTD ead.dtd (Encoded Archival Description (EAD) Version 2002)//EN">
<ead>
```
FRONT MATTER <frontmatter>
Because the <eadheader> must adhere to a relatively strict structure to ensure uniformity across platforms of finding aids, EAD also includes an optional <frontmatter> element which is used to generate a title page that can be customized to meet local needs. It can also be used to encode data elements such as prefaces, dedications, or other text concerning the creation, publication, or use of the finding aid.

ARCHIVAL DESCRIPTION <archdesc>
According to the EAD web site, this element describes administrative and descriptive information about the collection. It is a wrapper element for the bulk of an EAD document instance, which describes the content, context, and extent of a body of archival materials, including administrative and supplemental information that facilitates use of the materials. The <archdesc> element has several specialized attributes. The required LEVEL attribute identifies the character of the whole unit, for example, "class," "collection," "fonds," "recordgrp," "series," "subfonds," "subgrp," "subseries," or "otherlevel." This attribute is comparable to ISAD (G) data element 3.1.4 and MARC field 351 subfield c.

While everything about the EAD is remarkable, what I personally found useful was the extensive documentation of the EAD. This is especially true about the EAD TAG LIBRARY. The EAD Tag library contains a description of the 146 elements arranged in alphabetical order by their names. There are also links that provide examples for the usage of the elements. What impresses me about the tag library is its simplicity in describing elements to the understanding of people who have little exposure to markup languages or data structures. It starts with an explanation of conventions and continues with detail descriptions of the element names. Here is an example of the entry for attributes:

ATTRIBUTES
"Identifies all attributes that can be associated with an element. Attributes are represented in lowercase letters in XML coding, but the Online EAD Tag Library uses the convention of capital letters to distinguish attributes from elements within the tag library context. Attributes are explained further in detail with examples in the tags library:

Attributes are associated with most of the elements contained in EAD. These attributes reflect named properties of an element and may take on different values, depending on the context in which they occur. In order to set one or more attributes, an encoder should include the name of the attribute(s) within the same angle bracket as the start tag, together with the value(s) to which the attribute(s) is/are to be set. That is, <[tag] [attribute]=["value"]> or <[tag] [attribute1]=["value1"] [attribute2]=["value2"]] > For example: <unitdate type="inclusive">1937-1992</unitdate> or <origination label="Creator">Kenny, Elizabeth</origination> <unittitle encodinganalog="MARC 245" label="Title">Elizabeth Kenny Papers</unittitle>

In conclusion, I wish to say that there is no doubt in my mind that this is a success story and I think it is a success story because of a few important things: The involvement of the SAA at the appropriate time was a key factor in the success of the EAD. I think the
developers realized very early in the project that the EAD will significantly affect the archival community so they made a smart move of involving them. Barely 2 years into the development of the EAD, the SAA’s Committee on Archival Information Exchange (CAIE) agreed to take the responsibility for involving interested archivists. This collaboration has proved to be an invaluable one over the years. They also went ahead to involve the user communities all around the world. In early 2000 they solicited and received 67 suggestions for changes and additions from a web-based suggestions form on the EAD web site. The EAD working Group which also included representatives from Australia, Canada, France, UK and the US, met to discuss these suggestions in Washington DC, from April 27-29, 2000. This meeting resulted in the deprecation of only eight (8) EAD elements that had been part of the Version 1.0 (1998) EAD DTD.

Also, it was critical for the developers of the EAD to build it on existing languages, technology and conventions. This certainly made it possible for easy adoptions. The understanding and use of Schemas and DTDs applied to the EAD and borrowing from the TEI were critical to the success of the EAD because these are well known data languages, formats and conventions in the data and metadata world. Modeling the EAD on these languages enormously helped the developers not to re-invent the wheel in every single case. This also meant that they significantly reduced the need to experiment with new methods and all the effects that come with it. On the other hand, they cleverly chose some of the best parts of these languages to enhance the EAD. For example, borrowing from one of the characteristics of SGML which allows you to define attributes and associate them with particular elements was an important step in the right direction.

Continuously updating the EAD also ensured that it did not become a kind of a secret coding that needed privileged knowledge to crack it. The EAD Schema 1.0 and DTD 1.0 were based on SGML conventions but when suggestions were made to let it also model some specifics of XML, the developers agreed, solicited more suggestions and modeled the latest version-2002 on both SGML and XML conventions and provided a means to switch on-and-off features that were specific to either XML or SGML. This flexibility has helped in many ways to keep the EAD as current as possible even in the face of unprecedented advances in internet and database technologies. The advantages of maintaining a comprehensive tag library which provides unambiguous definitions with practical examples can never be overemphasized.

These and other important steps taken by the EAD developers including, making it possible to build data structures that were platform independent has made it possible for many universities, libraries, archives, repositories and digital libraries around the world to adopt it and there is no doubt that many more will use it in the future.

REFERENCES:


