A data model supporting conflict resolving in cooperative text editions – HypereiDoc *

Péter Bauer and Zsolt Hernáth
Dept. of Programming Languages and Compilers
and Dept. of Information Systems
Faculty of Informatics, Eötvös Loránd University
Pázmány Péter sétány 1/C H-1117 Budapest, Hungary

E-mail: {bauer_p|hernath}@inf.elte.hu

Abstract

HypereiDoc [1] is an XML based framework that has been designed to support multi-layered processing of epigraphical, papyrological or similar texts in a cooperative, and distributed manner for modern critical editions. Creating an edition, philologists may however face the problem that a prepared edition is semantically unjust. The reason behind semantically damaged editions is merging virtual text-documents that annotate the same piece of text differently, and occasionally made by different scholar teams independently of each other. Current initiatives like TEI and Epidoc have definitive limitations both in expressional power and the way how individual results can form a cooperative product. The HypereiDoc framework provides XML schema definition for a set of annotation-based layers connected by an extensive reference system, validating and building tools, and an editor that makes on-line visualization of the base text and the annotations. The framework that makes scholars able to work on the same text in a cooperative and distributed way has been successfully tested by philologists working on the Hypereides palimpsest.[1]

Keywords: data model, XML, cooperative text edition

1. Introduction

The XML document format [16] is a well-respected solution for the document processing domain. A wide range of applications are based on XML from *DocBook* to *Microsoft Office Open XML*. The flexibility with the ease of machine processing makes XML an ideal format for document handling. Very special but increasingly important areas of document handling are *epigraphy* and *papyrology*. Epigraphy and papyrology include both the interpretation and translation of ancient documents. Such historical relics are often damaged and their study produces controversial results by nature. Scholars have long standing solution for the situation: the system of critical annotations to the text. Annotations may mark missing, unreadable, ambiguous, or superfluous parts of text. They should also carry information about the *reason* of the scholar's decision e.g. other document sources, well-accepted historical facts or advances in technology. Annotations also provide meta-information about the author of the individual critical notes and expose the supposed meaning according to the given scholar. Preserving any information during transcription processes has primary inportance, even those remarks which will never appear in any critical edition should be kept either. The *Leiden Conventions* are the most accepted set of rules and symbols to indicate annotations in literary, epigraphical or papyrological texts [8].

Our aim was to equip the scholarly teams with general and flexible tools, which enable them to create both consistently tagged source files and pretty printed output. While most epigraphycal systems, similarly to classical paper-based books, focus on the creation of a single document, HypereiDoc covers the full distributed process of scientific activity. In order to follow and join the related standards, for the realization of our goals the Text Encoding Initia-

*Supported by Nemzeti Kutatási és technológiai Hivatal under TECH_08-A2/2-2008-0089.

[1] The text edition of Hypereides' speech against Diondas is based on the above described editor. The transcript was published in the Zeitschrift für Papyrologie und Epigraphik vol. 2008 (October). Similarly, this editor was used to publish the revised edition of Hypereides' Against Timandros (cf. [13, 14]) in AAHung vol. 2008. After the above mentioned publications the entire Greek texts together with the editor is accessible in [17].

tive (TEI) Guidelines [18] and its epigraphical customization Epidoc [15] that provide detailed recommendations for storing documents in XML format has been chosen as start-up point. As our goals due to supporting embedded and overlapped annotations seemd far beyond the possibilities of the above standards, we establish a suitable extension of those. Meanwhile TEI released the new version P5 [20], and Epidoc was correspondently changed.

As the detailed description of the HypereiDoc algebraic datamodel as well as the application tool set far exceed the acceptable size of the paper, here we pay a great importance to conflict detection and resolution instead. The priors, both, can be found in [1]. For HypereiDoc XML schema, and also a list of philological applications of the framework, see [17].

The rest of the paper is organized as follows: section 2 presents a detailed description on the design and implementation of our *Virtual Text-document Annotation Model* based on our extension and customization of TEI P5 and Epidoc Version 5 standards. Section 3 reports on a successful application of the HypereiDoc framework in the Archimedes Palimpsest project. In section 4 we give an overview of XML based projects from the epigraphical and papyrological domain and approaches on XML related merging problems. Our results are summarized in section 5.

2. Virtual Text-document Annotation Model

Virtual Text-document Model stores documents as a set of TEI P5 conform XML documents distributed over different *document-layers* and provide low-level semantic operations on texts. Document-layers are not simply joined XML document fragments like XML general entities, but rather an XML document hierarchy where document-layers are connected via an extensive reference system. The model provides the means for stepwise adding of basic semantic information, summarizing the scholar team's knowledge base, team work, cross-checking, and proof-reading. Editions may be based upon one or more previously published layers, so that creating critical editions is also supported. We defined a *Base Text Layer* where only the original text and its physical structure is stored and which may not be modified later, one or more *Annotation Layers* with the attached philological metadata, and an *Ordering and Indexing Layer* defining the pages' order and place in the codices. Philologists can define their own Annotation Layers which may refer to the Base Text Layer or one or more Annotation Layers, so they can add notes and annotations to the original text and also to already exsisting annotations. To handle these references and to support the distributed and collaborative work with using more Annotation Layers in one edition, an adequate schema has been designed.

2.1. Document-layers

The primary structure of the text is its logical structure according to the TEI Guidelines[2]. As for the physical structure of the text, empty tags are suggested according to XML's milestone technique that follows the natural physical structuring provided by the palimpsest. The latter is a well-identifiable frame for the document processeing, and therefore considered as the primary structure for our Reference System. References are used to mark off and annotate particular part of the documents to establish mappings between the images and the transcription. The Base Text Layer is established to store the transcription of the text including both its logical and physical structure.

For philologists the most important aspect is the annotation facilities by using the Leiden Conventions, and the application of the critical apparatus. One weakness of TEI P4 and EpiDoc is that these pieces of information are stored within the document as XML element type instances [2, 5]. On the one hand, unless an element type content model allowes nested instances of the same element type, implementing embedded annotations encoded by the same element type instance is impossible without loosing semantics. On the other hand, due to the requirements for well formed XML documents, overlapping annotations done by the philologists, and encoded by different element type instances cannot be implemented at all as illustrated below. The string $omen$ is readable, however, aut beside it is missing due to a flaw in the material of the codex. The transcriber has succeeded at reconstructing the missing part. According to the Leiden conventions the respective annotation is $[aut]omen$. Nevertheless, the transcribing philologist observes that the t and o characters are superfluous, and probably got into the text as an error on the part of the original copyist of the document. This can be annotated as $[au\{t]o\}men$, but this annotation cannot be encoded with the element type instance suggested by TEI P4 and Epidoc. To remedy scantinesses of TEI P4 and Epidoc, our solution proposed was that annotations of the text have to be stored separately from the Base Text Layer in several Annotation Layers, and to develop an extensive Reference System built on the physical structure of the document. With this missing words and sentence boundaries can easily be described, even if interpreted differently by various philologists. Punctuations missing from the document can also easily be encoded, so that our extended model enable us to handle any overlapping or embedded annotation.

The Ordering and Indexing Layer is an especial document-layer. A detailed discussion of its necessity, and the layer itself are described in section 3.

[2] For the problems caused by structuring the text this way see [19].

2.2. Reference System

Most annotations may have two different meaning. It is possible that the character string we refer to is present in the base text or in a previous annotation. We call this type of annotation *Marking Annotation*. It is also possible that the annotation inserts new characters in the text. This type of annotation is called *Inserting Annotation*. The inserted text may only be referred relatively to this annotation. From this point of view, due to the embedded and overlapped annotations and our multi-layered document approach we define three types of references. Operations supporting editions are annotation-level operations, and therefore an important issue to mark off annotations themselves. That needs, however an additional kind of reference. References implemented by TEI P5 extended XPointers [21] are as follows.

The *Absolute References* point at a character position in the Base Text. The *Internal Relative References* point at a character position in text inserted or marked by a previous annotation in the same Annotation Layer. The *External Relative References* point at a character position in text inserted or marked by an annotation in a previous Annotation Layer. To mark out annotations *External Annotation References* are used.

Concernig External Annotation References, please note, there is no internal annotation reference, since there is no need to select annotations from the layer being edited. Philologists can easily insert and remove annotations to and from their unpublished Annotation Layer.

2.3. Virtual Text Semantic Validity

Virtual Text Semantic Validity is References' Satisfiability. Our model guarantees that annotating semantc valid virtual texts keeps semantic validity. Considering a Base Text Layer, and its all available annotation layers, a selection of an arbitrary subset of those including the Base Text Layer does not necessarily constitute semantic valid virtual text-document. As creating an edition takes place by selecting a subset of Annotation Layers for some Base Text Layer, it may result in semantic invalid virtual tetxt-document. There is only one common case that can, and in general does cause virtual text semantic validity problems: differently annotating the same piece of base or virtual (i.e. annotated) piece of text. The possible results of that, however, may be very various, as shown by the examples below.

First, let us suppose that the base text was unreadable, and the first annotation layer stated that the word "$master$" was restored there, which according to the Leiden conventions appeared as "$[master]$", then the second layer revised this annotation as "$[mater]$", while the third layer revised it as "$[magister]$". In this situation one can select the first and the second annotation layer or the first and the third one without conflict, but cannot select the first, the second and third one. In our second example the string "$omen$" is readable and is present in the Base Text Layer. Philologist 1 recognizes that "aut" is missing before "$omen$", and publishes this in Annotation Layer 1. This is encoded as "$[aut]omen$". Philologist 2 states that "to" is superfluous and publishes it in Annotation Layer 2 encoded as "$[au\{t]o\}men$"[3], then if revising the "$[aut]$" to "$[a]$" in Annotation Layer 3, or hiding the "$[aut]$" annotation leads to very similar conflicts.

2.4 Hiding annotations

To help creating editions, we want to make the philologist able to select an annotation to publish from a conflicting set and hide the others. Since in our model all annotations are shown by default, the only change we have to store is making an annotation invisible. Therefore in the layer describing the edition we store a $Hide$ operator with a reference to the annotation to be hidden.

However, hiding an annotation may cause new conflicts. If there are annotations in a published layer referring to the hidden annotation, and we include that layer in our new edition, all the referring annotations are in conflict with the Hide operation. Therefore we collect these conflicting annotations and create a conflict selection, which is represented by a new layer with references to those annotations which have references to a hidden annotation or references to any annotation which are already in this selection. This is needed because hiding an annotation makes unpublishable all the annotations which are refferring it, and if we hide these unpublishable annotations, this will cause new annotations to be unpublishable. Therefore the selection consists of exactly those annotations which are referring to the hidden annotation or any annotation in the selection.

After creating the selection of conflicting annotations, the philologist can use one of the options below to resolve the conflict. Since these operations resolves conflicts between two or more annotations, a new selection is to be created from the conflicting annotations left. The scholar can remove some new conflicts with another operation and these steps are repeated until the selection is empty.

The selected annotations are referring the hidden one with an External or Internal Relative Reference or an External Annotation Reference. These types of references are like a foreign key in a database system. Hiding the referred annotation may cause two kinds of effect: cascade-hiding all the referring annotations or changing them to point to NULL.

[3] A superfluous character in a codex is present, readable, but considered erronously inserted by the ancient scribe, thus "$[au\{t]o\}men$" has different semantics from "$[au]men$".

However the latter is not acceptable in a document processing system, since it is impossible to publish an annotation without knowing where to insert it in the text. Philologist agree that the cascade-hide mechanism is not acceptable either. They want to make decisions on which annotation to hide and which to show in place of the hidden annotation.

Splitting annotations To support this, it is possible to split up annotations, when a referring annotation overlaps the hidden one. Using the SPA operation, the referring annotation can be split into two consecutive annotations, of which one is embedded in the annotation to be hidden while the other one is independent. After the split, we can apply cascade-hide to the embedded annotation, while leaving the other part shown. Following the second example in section 2.3, if the annotator choses to hide "$[aut]$", it is a possible option to split up the "$\{to\}$" annotation to "$\{t\}\{o\}$" with hiding the "$\{t\}$", giving "$\{o\}men$".

Relocating annotations In case of embedded annotations, we need another logic. If the philologist hides an annotation in which another annotation was embedded, a fall-back mechanism can be used. In this case, we apply a REL (relocation) operation on the embedded annotation, after that it acts like it was inserted to the location, where the hidden annotation was. This operation can revise the Location of the referring annotation to add a reference to the point where the hidden annotation was inserted. For instance, if "$aumen$" was deleted and later restored (this is displayed as "$[[aumen]]$") and then a philologists makes an annotation that "to" was restored from a gap: "$[[au[to]men]]$". If this is published and another philologist wants to hide the "$[[aumen]]$" annotation, it is possible to leave "$[to]$" by relocating it to the point where "$[[aumen]]$" was insterted.

2.5 Automating the annotation-level operations

In real cases, when there are hundreds of annotations attached to a single page of original text, creating editions leads to many conflicts. We had a detailed research on how the decisions in conflict resolution can be automated. Working with philologists, we found that the decisions in many cases are predictable due to the semantics of operations.

Since inserting Annotations always refer to an empty text, an Inserting Annotation never overlaps with previously added annotations, it is always stand-alone or embedded. While this annotation type always adds new characters to the text which should not be hidden, using the fall-back mechanism when hiding referred annotations is the best choice in most cases.

Embedded Marking Annotations interprets only the text of the annotation they are embedded into. Therefore if the referred annotation was an Inserting Annotation, hiding it always makes the embedded Marking Annotation uninterpretable, therefore the cascade-hide is a good choice. In most cases, when the referred annotation was a Marking Annotation, the embedded annotation still makes sense after hiding the referred one, therefore the fall-back mechanism should be used. In case of overlapping annotations, when the referred annotation is an Inserting Annotation, the referring annotation should be split up. When the referred annotation is a Marking Annotation, the fall-back mechanism should be used.

Please note that hiding a Marking Annotation involves the fall-back mechanism in most cases, since the embedded and overlapped annotations does make sense without the interpretation of the hidden annotation. Our conclusions can be seen in table 1. These defaults are set in the editor but can be easily overridden in every single case.

Referred annotation	**Referring annotation**		
	Inserting	Marking	
	embedded	embedded	overlapped
Inserting	fall-back	cascade-hide	split
Marking	fall-back	fall-back	fall-back

Table 1. Automating operations

2.6 Meta-layer

When a philologist creates an edition, document-, layer- and annotation-level operations are done. Selecting published Annotation Layers to include in an edition is a layer-level operation, while selecting conflicting annotations and resolving the conflicts are annotation level operations. However these are very similar operations to the philologists annotations, therefore we call them meta-annotations. An edition is described by the meta-annotations needed to create it from published Annotation Layers. We store these meta-annotations in a *Meta-Layer*. Meta-Layers are very much like Annotation Layers, the difference is that references in Meta-Layers are pointing to annotations and meta-annotations, not character positions.

An edition which consists of only one Annotation Layer without reference to other Annotation Layers is called a *Basic Edition*. An edition which is based on more than one Annotation Layer is a *Complex Edition*. Every Complex Edition – even if it has no conflicts to resolve – has a Meta-Layer which may refer to Annotation Layers of Basic Editions and Meta-Layers of complex editions. A Basic Edition can be treated as having an empty Meta-Layer.

2.7. Tools

Our XML format contains a flexible XPointer scheme which is not easily editable by simple text editors. To support user friendly editing of the texts, we developed a What

You See Is What You Get editor. It is not only an editor but also helps with the publishing of the finished document.It supports working with layered and flat XML files: it has a Base Text mode which is used when one start working with a new codex. For the layered structure the editor has an Annotation mode. In this mode editing the base text is disabled, but adding, modifying and deleting annotations are still possible.

3. The Hypereides palimpsest – a sample project

The HypereiDoc system has been created with the application to the decipherment of the now famous Archimedes Palimpsest [11] in view. It consists of remains of at least five former codices. One of those discarded and reused books contained speeches of Hyperides [13, 14]. Here we faced a special situation: the Archimedes Palimpsest is a secondary product, it has been created from reused sheets of former manuscript books. Before the secondary usage the leaves must have been cleaned as much as possible to make them fit for bearing the new texts. Scholars are interested in both the old texts (hardly visible remains of a lower layer on the surface of the pages, as called *undertext*) and the new texts (an upper layer, as called *overtext*).

As the undertext has not yet been exactly identified on all leaves, and finding new leaves the whole codex has to be occasionally reordered, the Base Text Layer's physical structure is based on the overtext, the pages are identified with the overtext leaf and side while columns, lines are marked regarding the undertext, thus the undertext lines are exactly identifiable. Since the undertext can only be interpreted or even displayed in its original page order if the exact structure of the undertext is known, we defined the Ordering and Indexing Layer independently from the Base Text Layer. It had to be defined as an independent layer, because philologists may not agree on the page order, and perhaps want to use Ordering and Indexing Layer of their own. The Ordering and Indexing Layer assigns the overtext leaves and sides to undertext quires, leaves and sides.

Ending the project not only the final result has been documented, but also important steps in the scholarly process of creating the transcription, ie. the XML source files that contain much such information has beeen made public along with the printed version.

4. Related work

Robin La Fontaine has established the Delta Format for XML [6] which stores the changes between versions of XML files in XML format. The system also gives a DTD to validate the delta file based upon the DTD of the original XML file. Unfortunately this system cannot be used when only the differences are stored as annotations in XML layers.

Robin La Fontaine's approach to merging XML datasets in an intelligent way [7] is a well-suitable system when there's no semantic conflict between the data sets. However in our case it is not suitable, since it can't handle semantically conflicting annotations.

Grégory Cobéna gives an overview on XML change detection systems in [4]. None of the systems reviewed is suitable for us, because we are working with published, frozen editions which are very different. We are not looking for changes or similarities, but semantic conflicts, therefore we cannot use change detection.

Tancred Lindholm introduces the three-way merging technique for XML files in [9]. This technique cannot deal with semantic conflict resolution.

Gerald W. Manger describes a tree-based algorithm for megring SGML and XML files with respect to document validity in [10]. This approach has an accent on keeping a valid syntax, but does not have solutions to keep static semantics, which is needed to resolve annotation conflicts.

Peter Buneman and his co-authors gives an overview on how to deal with keys in XML documents in [3]. The paper discusses the difference between pointers used in the XML standard and keys used in database systems. It is a good starting point for our system, however this paper does not deal with validity in terms of statical semantics.

Md. Sumon Shahriar and Jixue Liu introduces referencial integrity constraints in means of dependencies and foreign keys in [12]. This is very close to our work, since Relative References and Annotation References are foreign keys in our XML layers. However, the fallback mechanism used in HypereiDoc is not described here.

5. Conclusion

XML-based frameworks are widely used in editing epigraphical, papyrological texts. Most common systems, like TEI [18] and Epidoc [15] are able to describe the larger part of the annotations required by the scholars, but lack support for overlapping annotations, cooperative and distributed work of teams of scholars as well as creating annotations. HypereiDoc is an XML based framework supporting distributed, multi-layered processing of epigraphical, papyrological or similar texts in a modern critical edition. With the extensions described in the paper, philologists can create editions from scratch and also based upon their previous and other teams' published work. Semantical conflicts in multi-layered documents can be detected and resolved using our model. This makes scholars able to summarize their knowledge in editions composed of many

previously published Annotation Layers on the same Base Text.

In the last months HypereiDoc has been proved to be an efficient epigraphical system used in creation of large amount of papyrological results. In this paper we reviewed the experiences regarding the framework. The growing community using HypereiDoc revealed a number of features where the system should be improved. We discussed the most interesting problems, including the merging problem which arises when overlapped annotations are later modified causing conflicts. We extended the model describing the HypereiDoc annotation system to capture the problem. We proposed solutions based on annotating the annotations.

References

[1] Péter Bauer, Zsolt Hernáth, Zoltán Horváth, Gyula Mayer, Zsolt Parragi, Zoltán Porkoláb, Zsolt Sztupák: HypereiDoc - An XML Based Framework Supporting Cooperative Text Editions, In: Paolo Atzeni, Albertas Caplinskas, Hannu Jaakkola (Eds.): Advances in Databases and Information Systems, 12th East European Conference, ADBIS 2008, Pori, Finland, September 5-9, 2008. Proceedings. Lecture Notes in Computer Science Vol. 5207, Springer Verlag 2008, ISBN 978-3-540-85712-9, pp. 14-29.

[2] Bauman, Syd.: TEI HORSEing Around. In Proceedings of Extreme Markup Languages, 2005.

[3] Peter Buneman, Susan Davidson, Wenfei Fan, Carmem Hara, WangChiew Tan: Keys for XML, Computer Networks, Volume 39, Issue 5, August 2002, pp 473 - 487.

[4] Grégory Cobéna, Talel Abdessalem, Yassine Hinnach: A comparative study for XML change detection, Institut National de Recherche en Informatique et en Automatique, Rocquencourt, France, July 2002.

[5] DeRose, Steven.: Markup Overlap: A Review and a Horse. In Proceedings of Extreme Markup Languages, 2004.

[6] Robin La Fontaine: A Delta Format for XML, XML Europe 2001, Berlin, 2001
http://www.gca.org/papers/xmleurope2001/papers/html/s29-2.html

[7] Robin La Fontaine: Merging XML Files: A New Approach Providing Intelligent Merge of XML Data Sets, XML Europe 2002, Barcelona, 2002
http://www.deltaxml.com/dxml/93/version/default/part/AttachmentData/data/merging-xml-files.pdf

[8] B. A. van Groningen: De signis criticis in edendo adhibendis. Menemosyne 59 (1932), pp. 362-365.

[9] Tancred Lindholm: A three-way merge for XML documents, Source Document Engineering, Proceedings of the 2004 ACM symposium on Document engineering, Milwaukee, Wisconsin, USA, 2004, pages: 1-10, ISBN:1-58113-938-1

[10] Gerald W. Manger: A Generic Algorithm for Merging SGML/XML-Instances, XML Europe 2001, Berlin, 2001
http://www.gca.org/papers/xmleurope2001/papers/html/s29-1.html

[11] Reviel Netz, William Noel: The Archimedes Codex. Revealing The Secrets Of The World's Greatest Palimpsest. ISBN-13: 9780297645474, London 2007.

[12] Md. Sumon Shahriar and Jixue Liu: Towards a Definition of Referential Integrity Constraints for XML, International Journal of Software Engineering and Its Applications, Vol. 3, No. 1, January, 2009

[13] Tchernetska, N.: New Fragments of Hyperides from the Archimedes Palimpsest. Zeitschrift für Papyrologie und Epigraphik 154 (2005) 1–6.

[14] Tchernetska, N., Handley, E. W., Austin, C. F. L., Horváth, L.: New Readings in the Fragment of Hyperides' Against Timadros. Zeitschrift für Papyrologie und Epigraphik 162 (2007) 1–4.

[15] Epidoc Guidelines.
http://www.stoa.org/epidoc/gl/5/toc.html

[16] Extensible Markup Language (XML) 1.0 (Third Edition)
http://www.w3.org/TR/2003/PER-xml-20031030

[17] HypereiDoc Project Homepage.
http://hypereidoc.elte.hu

[18] TEI P4 Guidelines.
http://www.tei-c.org/Guidelines/P4/index.xml

[19] TEI P4 Multiple Hierarchies.
http://www.tei-c.org/release/doc/tei-p4-doc/html/NH.html

[20] TEI P5 Guidelines.
http://www.tei-c.org/Guidelines/P5/index.xml

[21] TEI XPointer Supplements.
http://www.tei-c.org/release/doc/tei-p5-doc/en/html/SA.html