An XML based General Document Algebra supporting conflict resolving in cooperative text editions – HypereiDoc revisited *

Zsolt Hernáth[1], Péter Bauer[2], and Zoltán Porkoláb[2]

[1] Dept. of Information Systems
[2] Dept. of Programming Languages and Compilers
Faculty of Informatics, Eötvös Loránd University
Pázmány Péter sétány 1/C H-1117 Budapest, Hungary
{bauer_p|hernath|gsd}@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 made by different scholar teams that may annotate the same piece of text independently of each other. As nor detection, neither resolution of such semantic casualties is currently supported by the framework, in this paper we explore and analyze possible semantic problems, and extend the mathematical model of HypereiDoc in order to capture, and present possible solutions for them.

1 Introduction

HypereiDoc is an XML [4] based framework established to support distributed, multi-layered, version-controlled processing of epigraphical, papyrological or similar texts in a modern critical edition. Such studies are typically based on independent work of philologists using annotation systems like the Leiden Conventions [2]. Most of the epigraphycal systems focus on the creation of a single document, similar the classical paper-based books. HypereiDoc does, however, cover the full distributed process of scientific activity. Scholars can refer each others results: support, modify, and contradict observations of earlier editions in form of annotations. When this process reaches a certain point a new edition can be created. The HypereiDoc framework provides XML schema for a set of annotation-based layers connected by an extensive reference system, validating and building tools, and an editor on-line visualizing the base text and the annotations.

The HypereiDoc framework is based on a multi-layer data model. We defined a

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

Base Text Layer where only the original text and its physical structure is stored and which may not be modified later, an *Ordering and Indexing Layer* defining the pages' order and place in the codices and one or more *Annotation Layers* with the attached philological remarks.

To physically structure the text and to implement philological metadata, the HypereiDoc framework provides an XML schema, in which not only the physical structure, but also a large scale of annotation metadata could be expressed. The latter, among others, makes also possible to follow also the ownership and the full timeline of the text. HypereiDoc supersedes other initiatives like TEI [6] and Epidoc [3] by introducing a multi-layered document model together with an extensive reference system, the latter in order to make exact references to any piece of both the base text in the *Base Text Layer* and texts inserted by annotations in any *Annotation Layer*. The layered document model is able to express both embedded and also overlapping annotations.

Our framework has been successfully used by philologists. The text edition of Hypereides' speech against Diondas has been created[1] using the editor of the framework. Currently the entire Greek texts together with the editor is available on the project's home page [5]. The creation of 10 pages of greek text with more than 1500 annotations, scholars sometime made conflicting annotations, that unintentionally caused some of the possible validity problems mentioned in [1], proving an exhausted test not only for the XML schema and the tools but also for the scholar's cooperative working methodology. Validating the document with our static semantics validation tool revealed these errors, but finding the exact conflicting annotations in different layers and dissolving them has been found laborious. To avoid executing the full semantical validity check in the editor repeatedly and then expressing the bogus positions in an unacceptable complex way, we concluded that both the detection and resolving of such conflicts should be supported on the level of the data model.

The rest of the paper is organized as follows. In section 2 we present two interesting and possible cases as a result of concurrently annotating the same piece of text. In section 3 our data model VITAM, see subsection 2.3 in [1], is extended. The extension is strongly governed by the sample cases presented in section 2. Section 4 details conflict handling and resolution by applying the VITAM extension. In section 5, we give an overview of approaches on XML related merging and foreign key problems, and last, our results are summarized in section 6.

2 HypereiDoc Revisited

Establishing new editions from old ones using the Hypereidoc editor, philologists were frequently facing the fact that a set of selected annotations made by dif-

[1] The text edition of Hypereides' speech against Diondas was transcribed with the above described editor. The publication is forthcoming in the Zeitschrift für Papyrologie und Epigraphik vol. 2009 (April). Similarly, this editor is applied in the revised edition of Hypereides' Against Timandros in AAHung vol. 2008.

ferent philologist groups procuced an annotated text, where some annotations were occasionally simple lost i.e. disappeared, others became uninterpretable. The reason behind the above phenomena is the lack of a forced full-validation of merged Virtual text-documents reported by [1].

Though the prototype of the HypereiDoc editor offers forced full validity check of any edition of annotated text, a real support of philologists would be an editor-controlled and automatic full validation of any edition (i.e. a base text and selected annotations). To support philologists with editor-controlled automatic full validation, basically data model controlled means are on need. Such means need to extend the Hypereidoc base algebra VITAM[2], so that first VITAM's notions and definitions are summarized below.

2.1 VITAM Revisited

Our data model VITAM lies basically on the notions Raw Text, Base, and Virtual Text-documents, the validity of the latter, primitive text operations on Base and Virtual Text-documents, and what are called annotations, which are considered as philological operations, and which are implemented by primitive operation sequences, each by one.

Raw Text and Base Text-document:
Given a text R as a particular sequence of UNICODEs (UTF8) called Raw Text, an (R-based) Base Text-document is a TEI P5 [6] conformable XML document being valid against the document grammar [5] made from R, where XML tags are to indicate page-, column-, and linebreaks. The idea behind the notion of base text-documents was to establish a frame of reference in order to locate and reference any piece of text inside R.

Primitive operations on raw texts:
LO, that locates some piece of text inside a base text-document by specifying the first and the last character of the sub-text. Marking off characters takes place by TEI P5 extensions of XPointers [7], and instead of marking out character positions Xpointers are used to trace out positions between neighbouring characters. An LO operator is called performable, if an existing piece of text, or a position between existing neighbouring characters is in fact located.

IN, that interprets (i.e. assigns semantics to) a piece of text located by an **LO** inside a base text-document.

RE, that revises a piece of text located by an **LO** inside a base text-document, i.e. deletes, or overwrites the located subtext, or inserts a piece of text at the position located.

Homogenous Operation Sequence:
A (possible empty) sequence $\{o_0, \ldots, o_n\}$ of primitive operations is called homogeneous, if operands of each **LO** occurrence inside the sequence refers implicitly

[2] VIrtual Text Annotation Model

either to the same base text-document, or to a raw text literal operand of some preceding operation.

Annotation:
A possible empty homogeneous operation sequence that refers implicitly to a base text-document X_R, and established as a TEI P5 [6] conformable XML document being valid against document grammar [5] is called an annotation. There exists the empty annotation denoted by $A_\emptyset$.

Virtual Text-document:
Given X_R base text-document and $\{A_{t_0}, \ldots, A_{t_r}\}$ a non-empty annotation sequence, $(X_R, A_\emptyset)$ is an X_R-rooted virtual text-document, identical with X_R. Given X_R-rooted virtual text-document V_R, $(V_R, \{A_{t_0}, \ldots, A_{t_r}\})$ is an X_R-rooted virtual text-document, defined by the expression

$$((\ldots(V_R, A_{t_0})\ldots), A_{t_r}).$$

The raw text content of virtual text-document $(V_R, \{A_{t_0}, \ldots, A_{t_r}\})$ results in from V_R, by processing annotations $A_{t_0}, \ldots, A_{t_r}$ in the given order. Processing an annotation means performing its primitive operations in the order of the operation sequence that it implements. The idea behind the notion of virtual text-documents is to avoid committed results of executing arbitrary sequences of the above operations on any Base Text-document.

Merging Virtual Text-documents:
Given $V_R = (X_R, \{A_{t_{i_0}}, \ldots, A_{t_{i_r}}\})$, $V_R^{'} = (X_R, \{A_{t_{k_0}}, \ldots, A_{t_{k_s}}\})$ X_R-rooted virtual text-documents, and $\{B_{t_0}, \ldots, B_{t_n}\}$ annotation sequence. Suppose, for each natural number m, for that $0 \leq m \leq n$ holds, there exists $0 \leq j \leq r$, or $0 \leq l \leq s$ such that either $B_{t_m} = A_{ti_j}$ or $B_{t_m} = A_{t_{k_m}}$ hold. A virtual text-document of form $(X_R, \{B_{t_0}, \ldots, B_{t_n}\})$ is called a merge of V_R and $V_R^{'}$.

Well-formedness and Validity:
Virtual text-documents' well-formedness formally declares the conformance of XML documents that implements base text-documents and annotations against document grammar [5]. Virtual text-documents' validity is a semantics issue.

Annotation $A_\emptyset$ is valid with respect to any base and virtual text-document. Given V_R virtual text-document, an annotation A is valid with respect to V_R, iff all occurrences of operations **LO** inside A is performable. An annotation sequence $\{A_0, \ldots, A_s\}$ is valid w.r.t. V_R, iff A_0 is valid w.r.t. V_R, and $\forall 1 \leq i \leq s$, annotation A_i is valid w.r.t. $(V_R, \{A_0, \ldots, A_{i-1}\})$.

Given X_R base text-document, $(X_R, A_\emptyset)$ is a valid virtual text-document. Given V_R valid virtual text-document, and $\{A_0, \ldots, A_r\}$ annotation sequence being valid w.r.t. V_R, $(V_R, \{A_0, \ldots, A_r\})$ is a valid virtual text-document.

In VITAM generally, any homogeneous operator sequence is called *annotations*. Here, from now on, we restrict the general notion of annotations to particular homogeneous operation sequences by assigning particular homogeneous opera-

tion sequences to particular philological annotations, one to each in harmony with Leiden Conventions, as it is seen in table 1[3].

Leiden Conventions	**Explanation**	**Operations**
[...]	a lacuna or gap in the original text, not restored by the editor	LO to position and IN:missing on the selected empty text
[*abc*]	letters missing from the original text due to lacuna, restored by the editor	LO to position, RE:insert, LO to select the inserted text and IN:restored gap
a(*bc*)	abbreviation in the text, expanded by the editor	LO to position, RE:insert, LO to select the inserted text and IN:expanded abbreviation
< *ab* >	characters erroneously omitted by the ancient scribe, restored or corrected by the editor	LO to position, RE:insert, LO to select the inserted text, IN:restored omission
{*ab*}	letters in the text considered erroneous and superfluous by the editor	LO to select the text and IN:superfluous
.*ab*.	characters damaged or otherwise unclear in the text, ambiguous outside of their context	LO to select the text and IN:damaged
...	traces of letters on the surface, insufficient for restoration by the editor	LO to position and IN:unrestorable on the selected empty text
[[*abc*]]	deleted letters	LO to position, RE:insert, LO to select the inserted text and IN:restored deletion
Vac.	space left empty (vacat) on the surface	LO to position and IN:empty on the selected empty text

Table 1. Representing the Leiden Conventions

2.2 Conflict Analysis

There is only one common case that can, and in general does cause virtual text validity problems: differently annotating the same piece of base or virtual (i.e. annotated) piece of text. The possible results of those, however, may be very

[3] Please note that an operation sequence starting with an LO followed by an RE is representing an Inserting Annotation. If the text to be interpreted is already present in the Virtual Text Document, one can use a Marking Annotation which is represented by the same operation sequence without the starting LO and RE. (see section 3.2 in [1])

various, as the following examples show. For instance, if the base text was unreadable, the first annotation layer stated that there "*master*" was restored, which appeared "[*master*]" according to the Leiden conventions. After that 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.

Let us consider another example: The string "*omen*" is readable and is present in the Base Text Layer. Philologist 1 recognizes that "*at*" is missing before "*omen*", and publishes this in Annotation Layer 1. This is encoded as "[*aut*]*omen*" in Leiden Conventions. After that philologist 2 states that "*to*" is superfluous and publishes it in Annotation Layer 2: "[*au*{*t*]*o*}*men*".[4] In this case revising the "[*aut*]" to "[*a*]" in Annotation Layer 3, or hiding the "[*aut*]" annotation leads to very similar conflicts.

3 Extended VITAM – VITAM^{E}

For arbitrary X_R base text-document, an edition can be considered as a merge of X_R-rooted virtual text-documents, and basically, there is only one source of conflicts: annotating the same piece of text in different ways. To support conflict detection, our base idea is that given an edition of form $E = (X_R, \{E_0, \ldots, E_s\})$, the sequence $\{E_0, \ldots, E_s\}$ can always be rearranged a to a sequence of subsequences such that E is yield by an expression

$$((\cdots(X_R, \{E_{1_1}, \ldots, E_{k_1}\}), \cdots), \{E_{m_1}, \ldots, E_{k_m}\}),$$

where annotations of subsequence $\{E_{1_1}, \ldots, E_{1_{k_1}}\}$ are valid w.r.t $(X_R, A_\emptyset)$, and annotations of subsequence $\{E_{j_1}, \ldots, E_{j_{k_j}}\}$ for any $2 \leq j \leq m$ are valid w.r.t. $(X_R, \{\{E_{1_1}, \ldots, E_{k_1}\}, \ldots, \{E_{(j-1)_1}, \ldots, E_{(j-1)_{k_{j-1}}}\}\})$. As for conflict resolution, subsequences are checked if more than one annotations annotate the same text region, and if so, groups of those by common text regions are to be considered as one annotation with the semantics that each group represents any, but only one annotation on the text region in question.

3.1 Extension Basics

Definition 1 (Null Annotataion)
Given X_R base text-document, and V_R X_R-rooted virtual text-document, for arbitrary $\mathbf{LO}_{V_R}$ that locates a piece of raw or annotated text within V_R, the operation sequence $\{\mathbf{LO}_{V_R}\}$ is called a *null-annotation* of the text located, and denoted by $A^0_{\mathbf{LO}_{V_R}}$.

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

Definition 2 (Denominations and Notations)
For arbitrary annotation A, let $\mathbf{LO}_A$, and $\mathbf{LO}^A$ denote the text to be, and already annotated, respectively. The text denoted by $\mathbf{LO}_A$ and $\mathbf{LO}^{\mathbf{A}}$ is called the *source domain* and *annotated source domain*, respectively.

In extended VITAM annotations can be considered as characters from some alphabet that disjoint from that of the base test-document. Annotations are therefore supposed, that they can be located by applying operation **LOA**, just like any pieces of texts inside virtual text-documents, by applying operation **LO**. $\mathbf{LOA}_A$ denotes the **LOA** that locates annotation A.

Definition 3 (Orthogonality and Common Domain)
Given V_R virtual text-document, and annotations $A_1, \ldots, A_r$, each of them being valid w.r.t V_R, an annotation sequence $\{A_1, \ldots, A_r\}$ is called *orthogonal* w.r.t. V_R.

Suppose, V_R, and $\{B_1, B_2\}$ denote a virtual text-document, and annotation sequence, respectively, the latter being orthogonal w.r.t. V_R. Annotations B_1 and B_2 are said *common-domained* w.r.t. V_R, iff

- there exist a raw text t such that t is a common subtext of $\mathbf{LO}_{B_1}$ and $\mathbf{LO}_{B_2}$;
- both $\mathbf{LO}_{B_1}$ and $\mathbf{LO}_{B_2}$ locate the same single position, or
- single position $\mathbf{LO}_{B_1}$ ($\mathbf{LO}_{B_2}$) is inside raw text $\mathbf{LO}_{B_2}$ ($\mathbf{LO}_{B_1}$).

Tags of annotation sequence $\{A_0, \ldots, A_s\}$ being orthogonal w.r.t. V_R are said *common-domained* w.r.t. V_R, iff for each pair A_j and A_k, $0 \leq j \neq k \leq s$, either A_j and A_k are common-domained w.r.t. V_R, or, one can select a sequence $A_{k_1}, \ldots, A_{k_r}$ from annotations $A_0, \ldots, A_s$ such that $A_j = A_{k_1}$, $A_{k_r} = A_k$, and neighbouring tags of the sequence selected are common-domained w.r.t. V_R.

Definition 4 (Orthogonal Validity)
Let V_R, and $\{A_0 \ldots, A_r\}$ denote a valid virtual text-document, and annotation sequence being orthogonal w.r.t. V_R, respectively. Sequence $\{A_0 \ldots, A_r\}$ is called *orthogonally valid* w.r.t V_R, iff for each $\{A_{j_0}, \ldots, A_{j_m}\} \subseteq \{A_0 \ldots, A_r\}$, $(V_R, \{A_{j_0}, \ldots, A_{j_m}\})$ is a valid virtual text-document.

Remark 1
Notice, for any virtual text-document V_R, and annotations $A_0, \ldots, A_s$ being common-domained w.r.t. V_R, there always exists the shortest piece of text t in V_R denoted by $\mathbf{LO}_{\{A_0,\ldots,A_s\}}$ in harmony with definition 2. An annotation sequence consisting of annotations that are common-domained w.r.t. some virtual text-document can't be orthogonally valid w.r.t. the same virtual text-document, and vice versa.

Definition 5 (Orthogonally layered Virtual Text-Documents)
Let X_R be a base text-document. An X_R-rooted virtual text-document V_R of form $(X_R, \{\{B_{1_1}, \ldots, B_{1_{k_1}}\}, \ldots, \{B_{s_1}, \ldots, B_{s_{k_s}}\}\})$ is said it is *ortogonally layered* or given in a canonical form, if

– annotation sequences $\{B_{1_1}, \ldots, B_{1_{k_1}}\}, \ldots, \{B_{s_1}, \ldots, B_{s_{k_s}}\}$ are pairwise disjoint,
– annotation sequence $\{B_{1_1}, \ldots, B_{1_{k_1}}\}$ is orthogonal w.r.t. $(X_R, A_\emptyset)$,
– for each $2 \leq j \leq s$, $\{B_{j_1}, \ldots, B_{j_{k_j}}\}$ is orthogonal w.r.t. virtual text-document yield by the expression $((\cdots(X_R, \{B_{1_1}, \ldots, B_{1_{k_1}}\}), \cdots), \{B_{j-1_1}, \ldots, B_{j-1_{k_j}}\})$,

Remark 2
Keeping notations of Definition 5, one can constructively prove that arbitrary X_R-rooted virtual text-document $(X_R, \{B_0, \ldots, B_u\})$ can be transformed into a canonical form. Creating a canonical form, first all those annotations has to be selected as the subsequence $\{B_{1_1}, \ldots, B_{1_{k_1}}\}$ that are valid w.r.t. $(X_R, A_\emptyset)$, next those as $\{B_{2_1}, \ldots, B_{2_{k_2}}\}$, which are valid w.r.t. $((X_R, A_\emptyset), \{B_{1_1}, \ldots, B_{1_{k_1}}\})$, and so on until each B_j $(j = 0, \ldots, u)$ has been selected.

It is also easy to see, that if V_R is an X_R-rooted virtual text-document given in a canonical form as $(X_R, \{\{B_{1_1}, \ldots, B_{1_{k_1}}\}, \ldots, \{B_{s_1}, \ldots, B_{s_{k_s}}\}\})$, then V_R is valid, iff $\{\{B_{1_1}, \ldots, B_{1_{k_1}}\}$ is orthogonally valid w.r.t. $(X_R, A_\emptyset)$, and for each $j = 2, \ldots, s$, $\{B_{j_1}, \ldots, B_{j_{k_j}}\}$ is orthogonally valid w.r.t. virtual text-document $(X_R, \{\{B_{1_1}, \ldots, B_{1_{k_1}}\}, \ldots, \{B_{(j-1)_1}, \ldots, B_{(j-1)_{k_{(j-1)}}}\}\})$.

3.2 Operations on Annotations

Operations on annotations may also be considered as annotations on annotations, or with other words meta-annotations. The operations introduced below are to support conflict detection and their resolution.

Definition 6 (Selection)
Suppose, V_R is a virtual text-document, $A_1, \ldots, A_r$ are common-domained annotations w.r.t. V_R. $\{A_1, \ldots, A_r\}^S$ is an annotation issued the semantics that it represents any, but only one annotation from the sequence $\{A_1, \ldots, A_r\}$, and called a *selection* of annotations $A_1, \ldots, A_r$.

Remark 3
Suppose, V_R is a virtualtext-document, $\{B_0, \ldots, B_r\}$ is an annotation sequence being ortogonally valid w.r.t V_R, annotations $A_1, \ldots, A_s$ are common domained w.r.t V_R, and source domain $\mathbf{LO}_{\{A_1, \ldots, A_s\}}$ is disjoint from any $\mathbf{LO}_{B_j}$ for each $j = 1, \ldots, r$. Under such conditions the sequence $\{\{A_1, \ldots, A_s\}^S, B_0, \ldots, B_r\}$ is orthogonally valid w.r.t. V_R.

Definition 7 (Prepare Selection of Annotations)
Suppose, V_R is a virtual text-document, operation $\mathbf{LO}_{V_R}$ locates some raw or annotated text inside V_R, and annotation sequence $\{C_0, \ldots, C_s\}$ is orthogonal w.r.t. V_R. Operation $\mathbf{PSA}(\mathbf{LO}_{V_R}, \{C_0, \ldots, C_s\})$ results in a LOA that locates a (occasionally empty) selection $\{C_{i_0}, \ldots, C_{i_m}\}^S$, where

– $\{C_{i_0}, \ldots, C_{i_m}\} \subseteq \{C_0, \ldots, C_s\}$,
– annotations $C_{i_0}, \ldots, C_{i_m}$ are common-domained,

– domains $\mathbf{LO}_{V_R}$ and $\mathbf{LO}_{\{C_{i_0},\ldots,C_{i_m}\}}$ are in relation $\mathbf{LO}_{V_R} \subseteq \mathbf{LO}_{\{C_{i_0},\ldots,C_{i_m}\}}$, or $\mathbf{LO}_{V_R} \supseteq \mathbf{LO}_{\{C_{i_0},\ldots,C_{i_m}\}}$.

Definition 8 (Complete selections For Undo)
Given selection $\{S_0,\ldots,S_r\}^S$, operation $\mathbf{CFU}(\{S_0,\ldots,S_r\}^S)$ results in a **LOA** that locates the selection $\{A^0_{LO_{\{S_0,\ldots,S_r\}}}, S_0,\ldots,S_r\}^S$.

Remark 4
Keeping the notations of the two above definitions one may realize, that particular applications of operations **PSA** and **CFU** give the means for undoing arbitrary annotations. Given arbitrary virtual text-document V_R, and annotation A being valid w.r.t. V_R, the means itself is the selection $\{A^0_{LO_A}, A\}^S$, and can be obtained by performing either $\mathbf{PSA}(\mathbf{LO}_A, \{A^0_{LO_A}, A\})$, or $\mathbf{CFU}(\{A\})$.

Definition 9 (SElect Annotation from selection)
Given selection $S^S = \{S_0,\ldots,S_v\}^S$, let A^* denote one of the annotations $S_0,\ldots,S_v$. Operation $\mathbf{SEA}(\mathbf{LOA}_{A^*}, S^S)$ results in $\mathbf{LOA}_{A^*}$.

Notice the operation introduced above helps both detecting and resolving conflicts caused by annotations being common-domained w.r.t. some virtual text-document. Conflict resolutions achieved by them are however rather coarse in the sense that different annotations of their common source domain may accidentally contain the same character at the same position. In order to carry out conflict resolutions as fine as possible, contradicting annotations need to be splitted to annotation sequences as formalized below.

Definition 10 (Split Annotation)
Given annotation A, and an **LO** that locates a single position within text literal operand of A, operation $\mathbf{SPA}(\mathbf{LO}, A)$ results in a **LOA** to sequence $\{S_{1_A}, S_{2_A}\}$ such that $\mathbf{LO}_{S_{1_A}}$ and $\mathbf{LO}_{S_{2_A}}$ in this order are consecutive source domains.[5]

Definition 11 (Join Annotations)
Given annotation sequence $\{A_0, A_1\}$, where $\mathbf{LO}_{A_0}$ and $\mathbf{LO}_{A_1}$ in this order are consecutive source domains, suppose, annotations A_0 and A_1 do not perform either **IN** operation, or both perform the same. Operation $\mathbf{JOA}(\mathbf{LOA}_{\{A_0,A_1\}})$ results in a **LOA** to the annotation

– the source domain of which are the catenation of consecutive source domains $\mathbf{LO}_{A_0}$, and $\mathbf{LO}_{A_1}$,
– the annotated source domain of whicih in turn is the catenation of annotated source domains $\mathbf{LO}^{A_0}$ and $\mathbf{LO}^{A_1}$ in this order.

4 Implementation

The new operations are needed because of creating editions, which is primarily a document-level activity, more exactly a layer-level activity. However implementing the new operations has effect both on annotations and on layers.

[5] Consecutive source domains are those where the end-position of the first domain, and the initial position of the second domain are the same.

4.1 Annotation-level operations

Operations supporting editions are annotation-level operations, since their Location points to an annotation in a previously defined layer. This type of Location is implemented by an *External Annotation Reference*. This type of reference is expressed by an XPointer pointing to the XML tag representing the annotation to be selected. Please note, that 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.

4.2 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 referring 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 resolve 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 collected annotations are referring the one to be hidden with an External or Internal Relative Reference or an External Annotation Reference. These types of references are similar to foreign keys in relational database systems. Hiding the referred annotation may cause two kinds of effect: cascade-hiding all the referring annotations or changing them to point to NULL. 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 2.2, 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.

4.3 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 analysis 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.

From the practical point of view, there are two types of annotations, the *Inserting Annotations* and the *Marking Annotations*[6]. Inserting Annotations are changing the characters of the Virtual Text, while Marking Annotations marks the text to add interpretations. Inserting Annotations always refer to an empty text, therefore 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.

[6] For detailed definition, see section 3.2 in [1]

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 2. 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 2. Automating operations

4.4 Layer- and document-level operations

Creating an edition means summarizing philologists' work on a given Base Text. Creating editions from published Annotation Layers consists of three main steps:

- Selecting the annotation layers containing the annotations we want to publish.
- Analyzing the conflicts between the annotations in these layers and resolving them.
- Typesetting the text with the selected annotations.

The first step is already supported by our framework, since selecting the Annotation Layers to be published is done via the same mechanism as selecting Annotation Layers to base a new annotation layer on, when creating a critical Annotation Layer for a critical edition. The last step is also supported by our editor with LaTeX and ledmac [8].

While the editor is able to detect conflicting annotations, we have to extend our data model to support storing the decisions made to resolve the conflicts. We also introduce some new functionality to support creating editions from different types of Annotation Layers and previous editions.

Please note that an Annotation Layer itself with its Base Text is also an edition, since it contains non-conflicting annotations, and the editor has the ability to export it in a format to be printed or published on the web. Therefore we will refer critical editions and editions consisting of only one Base Text Layer and Annotation Layer as *edition*.

Join Joining editions is the simplest way to create a new one. We can join editions if the domain of these editions are disjoint. In this case no conflicts can occur.

Split Splitting an edition is also a simple way to create new editions. This feature is helpful when a philologist want to publish only a portion of the text, for instance a speech from the Hypereides Palimpsest.

Merge Merging editions is similar to join except that the original editions' domain is not pairwise disjoint, therefore there may be and in general are conflicting annotations. Please note, that such conflicts can always be solved by splitting the conflicting layers and joining them again leaving out the parts which caused the conflict. However, using the new operations gives more freedom to the philologist to chose which annotations to hide, and makes it possible to leave annotations referring to a hidden one shown.

4.5 Meta-layer

When a philologist creates an edition, document-, layer- and annotation-level operations are performed. 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.

5 Related work

During the HypereiDoc project a number of existing XML merging systems and approaches were carefully revised.

Robin La Fontaine has established the Delta Format for XML [9], 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 [13]

is a well-suitable system when there's no semantical 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 [10]. 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 semantical conflicts, therefore we cannot use change detection.

Tancred Lindholm introduces the three-way merging technique for XML files in [11]. This technique cannot deal with semantical conflict resolution.

Gerald W. Manger describes a tree-based algorithm for merging SGML and XML files with respect to document validity in [12]. 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 [14]. The papers 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 [15]. This is very close to our work, since Relative References and Annotation References are foreing keys in our XML layers. However, the fallback mechanism used in HypereiDoc is not described here.

6 Conclusion

XML-based frameworks are widely used in editing epigraphical, papyrological texts. Most common systems, like TEI [6] and Epidoc [3] are able to describe the larger part of the annotations required by the scholars, but lacks to support 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. B. A. van Groningen: De signis criticis in edendo adhibendis. Menemosyne 59 (1932), pp. 362-365.
3. Epidoc Guidelines. http://www.stoa.org/epidoc/gl/5/toc.html
4. Extensible Markup Language (XML) 1.0 (Third Edition) http://www.w3.org/TR/2003/PER-xml-20031030
5. HypereiDoc Project Homepage. http://hypereidoc.elte.hu/
6. TEI P5 Guidelines. http://www.tei-c.org/Guidelines/P5/index.xml
7. TEI XPointer Supplements. http://www.tei-c.org/release/doc/tei-p5-doc/en/html/SA.html
8. Peter Wilson: Ledmac. ftp://dante.ctan.org/tex-archive/macros/latex/contrib/ledmac/
9. Robin La Fontaine: A Delta Format for XML, XML Europe 2001, Berlin, 2001 http://www.gca.org/papers/xmleurope2001/papers/html/s29-2.html
10. 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.
11. 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
12. 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
13. 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
14. Peter Buneman, Susan Davidson, Wenfei Fan, Carmem Hara, WangChiew Tan: Keys for XML, Computer Networks, Volume 39, Issue 5, August 2002, pp 473 - 487.
15. 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