HypereiDoc – An XML Based Framework Supporting Cooperative Text Editions*,**

Péter Bauer[1], Zsolt Hernáth[2], Zoltán Horváth[1], Gyula Mayer[3], Zsolt Parragi[1], Zoltán Porkoláb[1], and Zsolt Sztupák[1]

[1] Dept. of Programming Languages and Compilers
[2] Dept. of Information Systems
Faculty of Informatics, Eötvös Loránd University
Pázmány Péter sétány 1/C H-1117 Budapest, Hungary
bauer_p@inf.elte.hu, hernath@ullman.inf.elte.hu, hz@inf.elte.hu
gsd@inf.elte.hu, zsolt.parragi@eotvos.elte.hu, sztupy@eotvos.elte.hu
[3] Hungarian Academy of Sciences, Research Center for Ancient Studies
Múzeum krt. 4/F. H-1088 Budapest, Hungary
gam@cs.elte.hu

Abstract. HypereiDoc is an XML based framework supporting 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. 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 on-line visualizing the base text and the annotations. The framework makes scholars able to work on the same text in a cooperative and distributed way. Our framework has been successfully tested by philologists working on the Hypereides palimpsest.[1]

1 Introduction

The XML document format is a well-respected solution for the document processing domain. A wide range of applications are based on XML from *DocBook*

* Supported by ELTE Informatikai Kooperációs Kutatási és Oktatási Központ.
** The initiative and the frames of the interdisciplinary co-operations have been established and are maintained by the Classical Philology Workshop – Eötvös József Collegium. (László Horváth; OTKA inv. no. T 47136 and IN 71311; horvathl@eotvos.elte.hu)

[1] The text edition of Hypereides' speech against Diondas is based on the above described editor. The publication is forthcoming in the Zeitschrift für Papyrologie und Epigraphik vol. 2008 (October). Similarly, this editor will be applied in the revised edition of Hypereides' Against Timandros (cf. [8,9]) forthcoming in AAHung vol. 2008. After the above mentioned publications the entire Greek texts together with the editor will be made accessible on the URL: http://hypereidoc.elte.hu/

P. Atzeni, A. Caplinskas, and H. Jaakkola (Eds.): ADBIS 2008 , LNCS 5207, pp. 14–29, 2008.

[11] to *Office Open XML*, the new document format of Microsoft Word [22]. 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 is the study of inscriptions or epigraphs engraved into durable materials (e.g. stone). Papyrology focuses on the study of ancient literature, correspondence, legal archives, etc. as preserved in papyri. Epigraphy and papyrology include both the interpretation and translation of ancient documents. Such historical relicts are often damaged and their study produces controversial results by nature. Scholars have solutions of long standing 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 quote 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. It is of a primary importance that no information should be lost during the transcription process, 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 [4].

The aim of the project is to equip the scholarly teams with general and flexible tools, which enable them to create both consistently tagged source files and pretty printed output.

For the realization of the project goals the XML document format has been chosen for storing the documents and the associated pieces of information. This format primarily supports the Unicode character encoding. The Text Encoding Initiative (TEI) Guidelines [25,27] provide detailed recommendations for storing documents in XML format. Its epigraphical customization is called EpiDoc [18]. During the course of the project we take these standards and recommendations as starting points. The TEI Guidelines' version at the time the project started was P4, now it is P5. The current EpiDoc (version 5) is based on TEI P4 and is not yet updated to TEI P5 but we follow the Guidelines of it. However, our goals exceed the possibilities of these recommendations, thus their customization and completion is required.

The rest of the paper is organized as follows: We formalize the problem in section 2. The design and implemenation of the HypereiDoc framework are discussed in details in section 3. The successfull application of the HypereiDoc framework in the Hypereides palimpsest project is described in section 4. In section 5 we give an overview of XML based projects from the epigraphical and papyrological domain. We summarize our results in section 6.

2 Formalizing the Problem

In order to formalize the problem and the needs of computer support discussed informally above, consider the following computer produced linguistic puzzle:

– get a human-readable text-file and split it into portions, called pages;
– damage strings by inserting, deleting, or replacing few characters;
– concatenate the pagess in a random order.

To solve such text puzzles, i.e. to reconstruct the hypothetically original text

– the first task is to find the joints where the text can be splitted into pages
– the second is to restore damaged piece of text or characters,
– last concatenate pages in an adequate order found.

Following the above instructions the hypothetically original text can only be more or less exactly reconstructed after several tries. Each try may modify some words, may insert, delete, or replace characters, some of them use text versions produced by a sequence of earlier tries, etc. To make, catalogue, reference them, computer support is needed, which itself needs a problem-oriented data model that provides occasionally nested text operations cited informally above. To establish the adequate data model we need some base notions and terminology introduced next.

2.1 Basics

Definition 1 (Base text-document, Raw text-document, Raw text). Given R – a text, i.e a particular sequence of UNICODEs (UTF8), let X_R denote the TEI P5 conformed XML document being valid against the document grammar [21]. Texts R, and X_R are referred to as *raw*, and (R-based) *base text-documents*, respectively. Any portion of text R and of #PCDATA typed element content in X_R is called *raw text*.

Remark 1. Notice that X_R defines a *frame of reference* to locate and reference any piece of raw text of R.

Going on with introducing our base terminology, we introduce three primitive binary operatons used to locate, and issue semantics to, or modify raw texts inside base text-documents. Locating a raw text takes place by specifying the positions of its first and last character. In case of semantics issuer and corrective operations the first operand is a reference to a located raw text inside a base text-document, the seccond, in turn, always a raw text literal.

Definition 2 (Primitive Operations). Let O be the set of primitive operations $O = \{\mathbf{LO}, \mathbf{IN}, \mathbf{RE}\}$. They are semantic operations in the sense that each of them issues some semantics to raw texts inside a base text-document as seen below:

– **LO**cate: locate raw texts inside a base text-document.
Given X_R base text-document, and Xpointers x_s, x_e being valid according to the tagging of document X_R, $\mathbf{LO}(x_s, x_e)$ is the raw text betwen characters pointed to by x_s, and x_e, inclusive. An LO is called performable, if their operands are valid.

– **IN**terpret: interpret raw texts inside a base text-document.
Given raw text locator l_t for X_R base text-document as $\mathbf{LO}(x_{st}, x_{et})$, and t raw text literal, $\mathbf{IN}(l_t, t)$ issue semantics given by t to raw text located by l_t.
– **RE**vise: revise raw texts inside a base text-document.
Keeping notations l_t and t used above, $\mathbf{RE}(l_t, t)$ replaces raw text located by l_t with raw text t that may supply additional semantics as well.

2.2 Virtual Operation Performance

It is very important to see that base text-documents have to be kept untouched, even if semantics issuer or corrective operations are applied on it. One way to perform such operations would be to follow what database journal mechanisms do: executing operations results in new versioned complete (base) text-document, occasionally inheriting issued semantics, and corrections from other versions. This way would, however, lead to an unnecessary growth of base text-document versions, and instead, we develop a kind of *virtual* execution method whenever operations are to be performed. Informally, we say that applying a primitive operation o_1 to a base text-document X_R can be considered as an expression of form

$$(X_R, o_1),$$

and called a *virtual text-document*. If one wants to perform an operation o_2 on virtual text-document (X_R, o_1), simple create an expression of form

$$((X_R, o_1), o_2),$$

and so on. Following this philosophy, and considering the expression

$$(((\dots(X_R, o_0)\dots), o_{r-1}), o_r),$$

$\forall 0 < i \leq r$, operation o_i refers to a raw text inside the virtual text-document

$$((\dots(X_R, o_0)\dots), o_{i-1}).$$

The latter, however, means that operation **LO** sould be able to locate raw texts that are completely outside or partially inside X_R. That, in turn, in harmony with the definition of **LO** is possible, if operations' raw text literal can also be marked off by Xpointers, and all operations above implicitely refer to the same base text-document. We now formalize our conclusion as follows.

Definition 3 (Homogeneous Operation Sequence). A (possible empty) sequence $\{o_0, \dots, o_n\}$ of primitive operations is called homogeneous, if operands of each **LO** occurence inside the sequence – being present either as the first operand of some **IN** or **RE**, or as an operation of its own – refers implicitely either to the same base text-document, or to a raw text literal operand of some preceding operation.

Definition 4 (Annotation). A possibly empty sequence of homogeneous operations that refers implicitely to a base text-document X_R, and established as a TEI P5 conformed XML document being valid against document grammar [21] is called an annotation. The annotation that implements an empty sequence is called the empty annotation, and denoted by $A_\emptyset$.

Definition 5 (Virtual text-document). Given R-based base text-document X_R, and a non-empty annotation sequence $\{A_{t_0}, \ldots, A_{t_r}\}$, where indexes of annotations are some kind of time stamps indicating their creation time. $(X_R, A_\emptyset)$ 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.

Definition 6 (Merging Virtual text-documents having common roots). 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 $V_R^{'}$.

2.3 A Data Model for Text Annotations

The data model developed here is referred to as VITAM[2]. Informally, it contains *virtual text-documents* as data items and *annotation sequences* and virtual text-documents' *merging* as operations. Since data are virtual i.e. their raw text content can only be achieved by processing annotation sequences over virtual text-documents, an important issue is to define the *well-formedness*, and the *validity* of virtual text-documents.

Definition 7 (Well-formedness). Given X_R base text-documents, $(X_R, A_\emptyset)$ is a well-formed virtual text-document. Given V_R well-formed virtual text-document, and $\{A_0, \ldots, A_r\}$ annotation sequence, $(V_R, \{A_0, \ldots, A_r\})$ is a well-formed virtual text-document.

Definition 8 (Annotation Validity). Annotation $A_\emptyset$ is valid with respect to any virtual text-document. Given V_R virtual text-document, an annotation A is valid with respect to V_R, iff all occurences 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. $(V_R, \{A_0, \ldots, A_{i-1}\})$.

Definition 9 (Virtual text-document Validity). Given, X_R base text - document, $(X_R, A_\emptyset)$ is a valid virtual text-document. Given V_R valid virtual

[2] VIrtual Text-document Annotation Model.

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.

Remark 2. Notice, well-formedness does only declare the validity of XML documents virtual text-documents consist of against the document grammar [21]. It is also important to see that while valid annotation sequences w.r.t. valid virtual text-documents produce valid virtual text-documents from those, merging commonly rooted virtual text-documents does not, however, warrant valid merge, unless the merging procedure involves forced validity check.

3 Implementation

The XML model is based upon the TEI Guidelines (version P4 and P5) and its epigraphical customization, Epidoc Guidelines (version 5). We extended these standards to meet the HypereiDoc project requirements, thus we can use embedded and overlapped annotations and we also support a more free way to use the Critical Apparatus.

3.1 Layered Structure

Our schema is based on a multi-layer approach. We defined a *Base Text Layer* (see definition 1) 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* (see definition 4) with the attached philological metadata. This 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. Later editions may be based upon one or more previously published layers, thus creating critical editions is also supported.

Philologists can define their own Annotation Layers which may refer to only the Base Text Layer or one or more Annotation Layers. They can add notes and annotations to the original text and to previous annotations, they can make reflections on earlier work or create a new interpretation. We have designed a schema to handle these references and to support the distributed and collaborative work with using more Annotation Layers in one edition.

To make exact references to any point of the text, we need to discuss the structure of the text. The primary structure of the text is its logical structure according to the TEI Guidelines. The TEI suggests the header part for storing the associated information, while the text is structured by `div` tags. Also, for the physical structuring of the text representation empty tags are suggested according to XML's milestone technique. We store the transcription text in the Base Text Layer this way.

The palimpsest provides an existing physical structure of the text. This presents a well-identifiable base for processing the document as it can be clearly sectioned into codices, quires, leaves, sides, columns, and lines. Therefore we intend to regard these as the primary structure for our Reference System, making it possible to

define exact references to the document's specific parts. The references are needed for philological processing, for annotating the text and for mapping between the images and the transcription.

For philologists processing the document the most important aspect is the annotation facilities, as they can use it with the Leiden Conventions, and the application of the critical apparatus. The TEI sets up the structured recording of this information in the text, while the EpiDoc describes guidelines for the application of these techniques. However, a weakness of TEI P4 and EpiDoc is that these pieces of information are stored in the form of XML tags inserted into the document [2,3]. Therefore due to the requirements for well formed XML documents, annotations defined by the philologists can be embedded only if the tags are balanced.[3]

Let us consider the following example. The string *omen* is readable, however, *aut* beside it is missing due to a flaw in the material of the codex. At the same time, 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 XML tags suggested by TEI P4 and Epidoc. This is a quite possible situation in the palimpsest. Besides the philological annotations the text parts marked by the apparatus of similar passages may also overlap.

Consequently, we have developed a Reference System built on the physical structure of the document. This enables the handling of any overlapping annotation. With this reference system missing word and sentence boundaries can easily be described, even if interpreted differently by various philologists. Punctuations missing from the document can also easily be coded. As a result, the XML transcription may consist of several layers.

We face a special situation in the case of our sample project: 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*).

Since the undertext has not yet been exactly identified on all leaves and it is also possible that by finding new leaves we need to reorder the whole codex, we intend to regard the page numbering of the overtext as the base for the Reference System. This can be extended or changed while it does not affect the interpretation of the undertext. Since the undertext can only be interpreted or even displayed in its original page order if the exact structure of the undertext

[3] TEI P4 has draft recommendations on solving this problem [26], but these are not elaborated and less powerful than our Reference System. TEI P5 supports multiple ways to handle overlapping tags, and we use one of these techniques to implement the Reference System.

is known, we defined the Ordering and Indexing Layer independently from the Base Text Layer. We store this data in an external XML file because philologists may not agree on the page order and they may want to use their own Ordering and Indexing XML file. Ordering and Indexing mean that we assign the overtext leaves and sides to undertext leaves and sides and this assignment can be changed without modifying the Base Text Layer. Therefore we can change the page order in the restructured codices if needed without the need of changing the references of the annotations.

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. The Ordering and Indexing Layer assigns the overtext leaves and sides to undertext quires, leaves and sides.

3.2 Reference System

Due to the embedded and overlapped annotations and the multi-layer approach we define three types of references. The *Absolute References* point at a character position in the Base Text (cf. remark 1). Their structure is overleaf, overside, column, line, (optional) character, and (optional) position. The *Internal Relative References* point at a character position in text inserted by a previous annotation in the same Annotation Layer. Their structure is annotation identifier, (optional) character, and (optional) position. The *External Relative References* point at a character position in text inserted by an annotation in a previous Annotation Layer. They identify the previous layer, the annotation, (optionally) the character, and (optionally) the position. Notice, the Relative References above are particular cases of virtual operation performance (cf. section 2.2).

Only alphanumeric characters are numbered, whitespaces and the philologists' various brackets are disregarded because, in harmony with definition 1, they are annotations in the text. Character means only alphanumeric characters in the paper. Please note that the "character" field does not refer to a character but the position between two characters. The zeroth referred position is before the first character in the line while the first referred position is after the first character of the line and before the second. In a line containing n characters the n^{th} position is after the last character of the line.

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* (see operation **IN** in definition 2). The marked text may be later referred absolutely or relatively to this annotation. It is also possible that the annotation inserts new characters in the text. This type of annotation is called *Inserting Annotation* (see operation **RE** in definition 2). The inserted text may only be referred relatively to this annotation.

Relative References are used if we want to refer to characters inserted by a previous annotation. To make this possible all annotations regardless their type have an identifier which is unique in the given layer. Annotations must be

processed by the course of their identifier's lexicographic order which is identical to the order of the tags in the XML document.

Please note that if we want to refer to a character position in a text portion which is inserted by multiple embedded annotations, the exact annotation which has actually inserted the character is known, therefore we do not have to deal in the references with the annotation hierarchy.

In cases of embedded and overlapping annotations it is possible that the reference is ambiguous. For instance the base text is *abc* and the annotation claims that *def* is missing after *a* which is marked as $ab[def]c$ according to the Leiden Conventions. After that the absolute character position 1 is ambiguous: it can either point to $ab|[def]c$ or $ab[def]|c$. (The point where the examples refer to is marked with a | character.) In this case we use the Position attribute which has four values: "l" for left side, "r" for right, "b" for before, and "a" for after.

The "l" value is applicable when the Character attribute is present and it points the position before the annotation that was inserted to the Character position given. The "r" value is applicable when the Character attribute is present and it points the position after the annotation that was inserted to the Character position given. The "b" value is applicable in Relative References when the Character attribute is not present and it points the position before the referred annotation, while the "a" is applicable in Relative References when the Character attribute is not present and it points the position after the referred annotation.

Please note that the Position attribute is omittable but Character attribute can't be omitted if Position is not present or has the value of "l" or "r". If the Position attribute takes value "b" or "a" then Character attribute must be omitted.

We can also use Relative References to Marking Annotations. This makes unambiguous the character positions at the end of embedded and overlapped annotations. In the previous example the Relative Reference for character position 0 in the annotation refers to $ab[|def]c$ and the relative reference for character position 3 refers to $ab[def|]c$. This is useful when marked text is inserted before or after an already marked text part.

3.3 XML Pointer-Based Implementation of the Reference System

The XML Pointer Language (XPointer) Framework is a W3C standard that allows one to point to an arbitrary position in an XML document. An extension of Xpoinetrs introduced by TEI P5 Guidelines offers some supplements [28], such as the `left` and `right` pointer schemes, to the main standard.

To implement our reference system with XPointer we use the following mechanism: An *Absolute Reference* to leaf 27, side recto, column a, line 1 and character 3 looks like the following figure:

```
archimedes/P2/#xpointer(//pb[n='27:r']/following::cb[n='a']/
    following::lb[n='1']/following::text()[1]/point()[3])
```

In the example above `archimedes/P2` is the name of the base text file. In this encoding not only the file name but also the version number of the base text

is included, therefore possible later changes of the base text will not interfere with the reference system. After the file name the location part of the reference is converted into the exact position within the XML document. Because our XML structure has empty tags to mark the physical structure we had to use the following axis, which unfortunately makes the references more complex. After linking to the correct text node we use the point function of the XPointer scheme to point to the exact position.

In internal relative references we use

```
#xpointer(//[xml:id='b13']//text()/point()[2])
```

which means the second character position in the raw text in the thirteenth annotation. Our reference system allows us to include positional information like *left*, *right*, *before* or *after*. To use positions like *left*, or *right* we use TEI P5's supplement functions: `left` and `right`. We think of *before* and *after* as the position before or after an annotation, thus they are used like this:

```
#left(xpointer(//[xml:id='b13']))
```

This refers to the leftmost point of an annotation tag, which in turn is "before" the annotation. The after tag is similar but uses the `right` function instead of `left`.

External relative references are composed of the file information from *absolute references* and the relative positional information from *internal relative references*. We need to include a file name which includes the name of the annotator, a version number, and the location of the annotation like this:

```
annotations/mgy/a001#xpointer(//[xml:id='b2']//text()/point()[1])
```

Unfortunately, when the TEI Consortium designed the TEI P5 Guidelines they did not think about XPointer as a pointer to an arbitrary position, but as a pointer to an arbitrary tag. Because of this the guidelines lack support of the type of overlapping we need.

The guidelines allow us to implement new features by creating a new XML namespace but we wanted to stick with the P5 guidelines to maintain maximum compatibility. Therefore we had to use two of the tags that allow us to link to an interval in the text. These two tags are the `app` and the `note` tags. The `app` tags contains a critical notes to spans of texts. From a philological point of view this can also be used to describe the annotations we need. We use the `from` and `to` attributes to denote the start and end position of the annotation. The `note` tags have the `target` and `targetEnd` attributes to express the start and end position of annotations.

To ease the publication there is also a "flat file format" that is more close to the basic TEI P5 Guidelines. In this file format we do not use the XPointer scheme, because there are only a few tools that can handle it. Instead we use that feature of the `app` and the `note` tags, that allows them to be inlined into the text. In this mode at the beginning of the location the annotation refers to, we add

an `anchor` tag, and we add the `app` or the `note` tag at the end of the referenced interval with a link to the anchor. Of course this breaks the collaborative nature of our system, so this file format should only be used for a frozen digital dataset of a publication.

3.4 Version Control

Though this type of collaborative work is agile by nature, we do not establish a real version control system like Subversion or CVS does, where you can acquire the whole file, regardless whether it is the head revision or an earlier version. In our system what you have is a directory tree containing the base text-document and it's annotation sequences. Since the base text-document is read-only, and therefore, not under version control, the annotation sequence may be, however, extended. This extension means that new annotations may be added to such a sequence (e.g. annotations that occasionally have impacts on some previous annotations, but can never change them). Practically this is the same as other version controlling systems store their files internally, because they usually does only store the differences between the different versions.

To accomplish this, we use a web-server called WEBRick [10], that handles the requests in a RESTful [7] way. In this system a virtual-text document can be considered as a resource on which following version control operations are defined. The `list` operation shows the annotations of this resource (the base virtual-text document), the `create` operation adds a new annotation to the sequence (automatically time-stamped), and the `show` operation gets the appropriate version of the file from the server. Because we deny the modification and deletion of resources, operations `modify` and `delete` are not supported by our system.

3.5 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. In this mode the philologist may edit the base text and its annotations at the same time. Its primary output is the flat XML format which is after the base text is finished can be converted to the layered structure.

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. One can select already existing and published annotation XML files which will be the base of the work. The editor helps with the conflict resolution between the dependencies – when two or more XML documents conflicts each other – and it can also flatten the finished work to help the publication, because other TEI P5 compatible tools might use the flat file format more easily (see definition 6).

When one loads a layered XML document, the editor first checks wether or not the XPointers point to existing locations. If the input is valid it also validates the dependencies and tries to resolve the conflicts and unifies them controlled by the philologist whenever neccessary (cf. remark 2). If the previous process succeeds the user is greeted with a window like the screenshot in Figure 1. For

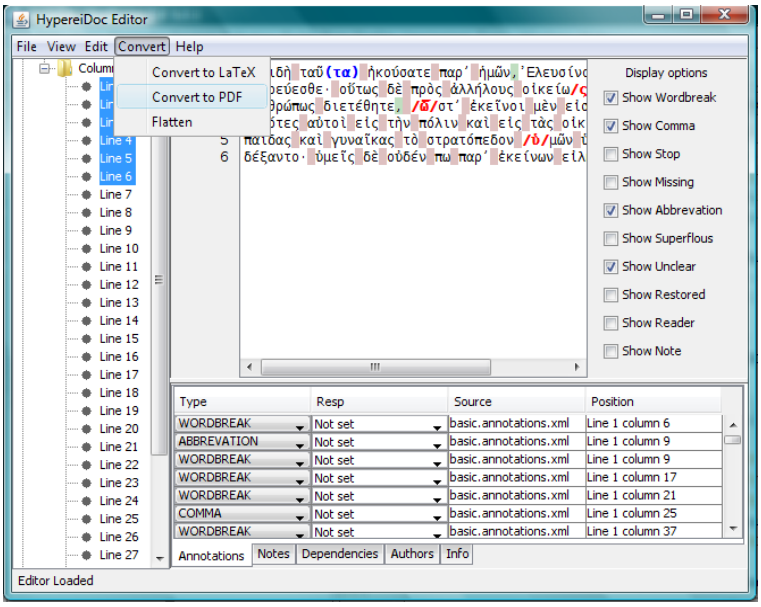

Fig. 1. Editor in Annotation mode

publications we provide some additional tools, such as the LaTeX converter, that converts the base text, and the selected annotations into a LaTeX file, that can be converted to PDF. Our integrated toolset consists of:

- Java based, platform-independent editor with graphical interface producing valid XML output.
- Java based tool for displaying the XML encoded transcription data in a form which is traditionally used by scholars, based on a compiler producing PDF.
- Java based validator tool which checks basic semantic relations.
- Java based converter tool exporting and importing flattened (one-layer) TEI P5 transcriptions.
- GTK based, platform-independent GIMP plug-in for linking image positions with lines of transcription producing XML output.

4 The Hypereides Palimpsest – A Sample Project

The HypereiDoc system has been created with the application to the decipherement of the now famous Archimedes Palimpsest [13] in view.[4] It consists of

[4] The story of the Palimpsest is described in detail in [6].

remains of at least five former codices. One of those discarded and reused books contained speeches of Hyperides [8], [9]. The transcription process involves many scholars working in different groups, making new suggestions and referring to each other's work.

It is the intention of the owner of the manuscript, that the Archimedes Palimpsest should be presented to the public in the most adequate way. Prime targets are the lower texts of the codex, together with the complex history of their decipherement. Presentation includes tagging and formatting. To format the sort of complex scholarly texts to be created, there exists an excellent platform, i.e. Edmac [5] resp. its variant for LaTeX Ledmac [30].

However, as excellent as is LaTeX at formatting, and as good as it is at tagging, the level of sophistication it provides at the latter is way behind what we need. Our goal is to document not only the final result, but also important steps in the scholarly process of creating the transcription. Of course, consistent tagging must not be so permissible as it is the case with a TeX based system. Thus we could retain LaTeX as our frontend for paper publication and paper-like visualization, but had to find an adequate system for tagging. Therefore a system meeting the complex scholarly requirements has been devised – the HypereiDoc framework.

Figure 2 shows the first five lines of a page. In the formatted output margins are reserved to refer to the present physical structure of the codex: in the left margin leaf 138, side recto is noted, in the right margin the line numbers. In line 3 parentheses indicate complementation of an abbreviation, and in line 5 curly brackets enclose a letter visible and readable in the codex, but superfluous according to grammatical rules. A dot beneath a letter shows that the letter is incomplete, but the traces are compatible with the interpretation presented. The

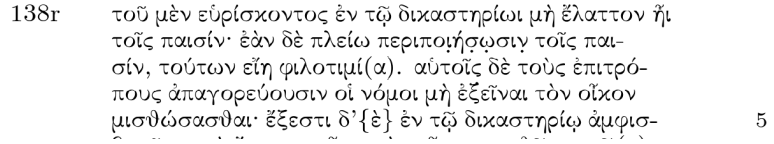

Fig. 2. Formatted main text

main text is accompanied by two series of annotations (a so called apparatus of similar passages and a critical apparatus). Figure 3 is a snippet of the second apparatus. Traditionally, the language of the apparatuses is latin with commonly used abbreviations. Since the existing pages do not contain the beginning and the title of the speech, the title is reconstructed from other sources, and does not have a corresponding line number. A significant difference between the XML source files and the output formatted for print is, that the latter contains only a selection of the information available. E.g. in line 5 the emendation of the text is so obvious, that no reason needs to be given. In line 1 reading has been substantially improved against the first publication in [8], and account should be given of it. In line 3 the cited scholar suggests to insert a two letter word, but this emendation is not regarded necessary by the editor of the apparatus. In line 10

Tit. vel Ὑπὲρ Ἀκαδήμου (or. [II] Jn.) coni. Todd || Fol. 138r **1** μὴ ἔλαττον ᾖ Wilson || **3** τούτων ⟨ἂν⟩ εἴη Handley || **10** scriba spatium quinque litt. in fin. v vacuum reliquit propter defectum membr., ubi antea cum stylo materiam perforavit lacunamque creavit || **14**

Fig. 3. Formatted annotations

we face a strange and rare situation, which can not be formalized and therefore is described in a 'human readable' sentence ("free text" within the annotations). The formatted version illustrated above is tailored to the conditions of a traditional paper publication. The XML source files contain much more information in an easily parseable form and will be made public simultaneously with the printed version to appear autumn 2008 in the same journal as [8].

Scholars throughout the world will be able to contribute, enhance or even fork new versions of the fileset if they deem so.

5 Related Work

During the HypereiDoc project, a number of existing related projects have been carefully revised.

Gothic Bible and minor fragments [20] are using TEI P4 without any reference to Epidoc. No overlapping annotation occurs in this project. Perseus Digital Library [24] is a huge library of TEI documents without any annotations. Aphrodisias Project at UNC and Kings College [12] is using Epidoc XML, but no overlapping annotation occurs. Digital Library Production Services at University of Virginia Library (DLPS) [16] uses TEI P4 with local modifications. The project adapts the standard to their needs. Center for Hellenic Studies - The Homer Multitext Project [15] is a TEI Core structure with embedded pictures instead of textual transcription. Most of Oxford Text Archive's [23] projects use the SGML (not the XML) version of TEI for the header only with simple ASCII text representation. The Newton Project uses TEI XML with nesting but without overlapping, and is not related to epigraphy. Cambridge University Press [14] is publishing CD-ROM versions of English literary classics, including the works of Chaucer, Shakespeare, Samuel Johnson, and John Ruskin. These projects are not related to epigraphy. UVA Library, University of Virginia Text Center [29] is using TEI without overlaps, and is not related to epigraphy. Duke University Digitized Collections [17] has mostly 20th century texts, and is not related to epigraphy.

6 Conclusions

The HypereiDoc framework has been created to provide informatics background for transcribing literary, papyrological or epigraphical documents. An XML-based multi-layered structure is introduced to allow distributed, version-controlled work of scholars in a cooperative way. Handling of philological notations, associated interpretations, and commentaries are supported by an extensive reference system

based on XML pointers. The expressive power of the defined document model exceeds the capability of the other proposals.

A software package of supporting tools was created to provide a convenient interface for recording information in an error-free way, validating and building, as well as for the creation of critical editions. The solutions provide the greatest portability possible between operating systems with respect to both the tools and the finished documents.

References

1. Abiteboul, S., Buneman, P., Suciu, D.: Data on the WEB – From Relations to Semistructured Data and XML, W3C Proposed Edited Recommendation, October 30, 2003, San Francisco (2000) ISBN 1-55860-622-X
2. Bauman, S.: TEI HORSEing Around. In: Proceedings of Extreme Markup Languages (2005)
3. De Rose, S.: Markup Overlap: A Review and a Horse. In Proceedings of Extreme Markup Languages (2004)
4. van Groningen, B.A.: De signis criticis in edendo adhibendis. Menemosyne 59, 362–365 (1932)
5. Lavagnino, J., Wujastyk, D.: Critical Edition Typesetting: The EDMAC format for plain TEX. San Francisco and Birmingham, TEXUsers Group and UK TEXUsers Group (1996)
6. Netz, R., Noel, W.: The Archimedes Codex. Revealing The Secrets Of The World's Greatest Palimpsest, London (2007) ISBN-13: 9780297645474
7. Richardson, L., Ruby, S.: RESTful Web Services. O'Reilly, Sebastopol (2007)
8. Tchernetska, N.: New Fragments of Hyperides from the Archimedes Palimpsest. Zeitschrift für Papyrologie und Epigraphik 154, 1–6 (2005)
9. 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, 1–4 (2007)
10. Thomas, D., Fowler, C., Hunt, A.: Programming Ruby: The Pragmatic Programmers' Guide, p. 733. O'Reilly, Sebastopol (2004)
11. Walsh, N., Muellner, L.: DocBook: The Definitive Guide. O'Reilly, Sebastopol (1999)
12. The Aphrodisias Project, UNC and Kings College, http://insaph.kcl.ac.uk/ala2004/
13. Archimedes Palimpsest, http://www.archimedespalimpsest.org/
14. Cambridge University Press, http://www.cup.cam.ac.uk/
15. Center for Hellenic Studies – The Homer Multitext Project, http://www.chs.harvard.edu/publications.sec/homer_multitext.ssp
16. The Digital Library Production Services, http://www.lib.virginia.edu/digital/reports/teiPractices/dlpsPractices_postkb.html
17. Duke University Digitized Collections, http://library.duke.edu/specialcollections/collections/digitized/
18. Epidoc Guidelines, http://www.stoa.org/epidoc/gl/5/toc.html
19. Extensible Markup Language (XML) 1.0 (Third Edition), http://www.w3.org/TR/2003/PER-xml-20031030
20. The Gothic Bible, http://www.wulfila.be/gothic/

21. HypereiDoc Project Homepage, http://hypereidoc.elte.hu/
22. Microsoft Office Word 97-2007 Binary File Format, http://www.ecma-international.org/news/PressReleases/PR_TC45_Dec2006.htm
23. The Oxford Text Archive, http://ota.ahds.ac.uk/
24. The Perseus Digital Library, http://www.perseus.tufts.edu/hopper/
25. TEI P4 Guidelines, http://www.tei-c.org/Guidelines/P4/index.xml
26. TEI P4 Multiple Hierarchies, http://www.tei-c.org/release/doc/tei-p4-doc/html/NH.html
27. TEI P5 Guidelines, http://www.tei-c.org/Guidelines/P5/index.xml
28. TEI XPointer Supplements, http://www.tei-c.org/release/doc/tei-p5-doc/en/html/SA.html
29. University of Virginia Text Center - UVA Library, http://etext.lib.virginia.edu/standards/tei/uvatei.html
30. Wilson, P.: Ledmac, ftp://dante.ctan.org/tex-archive/macros/latex/contrib/ledmac/