PhiloEditor®: simplified HTML markup for interpretative pathways over literary collections

Claudia Bonsi
University of Milano-Bicocca
claudia.bonsi@unimib.it

Angelo Di Iorio
University of Bologna
angelo.diiorio@unibo.it

Paola Italia
University of Bologna
paola.italia@unibo.it

Francesca Tomasi
University of Bologna
francesca.tomasi@unibo.it

Fabio Vitali
University of Bologna
fabio.vitali@unibo.it

Ersilia Russo
University of Florence
ersilia.russo@unifi.it

Abstract

English. In this paper we introduce PhiloEditor®, a software environment for the representation and coloring of time-based modifications of literary texts meant for scholarly editions. The purpose of the tool has evolved from a simple philological characterization of temporal evolution of literary texts to a critical and hermeneutic approach of interpretative pathways. From a technical point of view, PhiloEditor® is based on an approach to markup that heavily diverges from the traditional XML/TEI and towards a reliance on a custom subset of HTML5, based on a mature theory of XML design patterns, well-behaved and well-formed, readily convertible into XML/TEI or whatever other XML vocabulary needed, and at the same time fully compliant with modern web frameworks and browsers.

Italiano. Nel presente contributo introduciamo PhiloEditor®, un ambiente digitale per la rappresentazione e la marcatura delle modifiche temporali di testi letterari per l'allestimento di edizioni critiche digitali. L'obiettivo della piattaforma è evoluto dalla semplice caratterizzazione filologica dell'evoluzione temporale di testi letterari ad un approccio critico ed ermeneutico di percorsi interpretativi. Da un punto di vista tecnico, PhiloEditor® si basa su un approccio di markup che diverge fortemente dal tradizionale XML/TEI affidandosi ad un sottoinsieme personalizzato di HTML5, basato su una teoria matura di modelli di progettazione XML, ben educati e ben formati, facilmente convertibile in XML/TEI o in qualunque altro vocabolario XML, e allo stesso tempo pienamente conforme ai moderni framework e browser.

1 Introduction

Descriptive markup has been introduced three decades ago as the main mechanism to provide structured annotations over arbitrarily organized text documents. The rigidity of SGML and XML-based languages and the complexity of their editing tools made HTML an attractive markup language for text representations, as well as web pages, academic articles, structured documents and literary collections. However, the syntax of HTML is too flexible and its vocabulary too rich and unspecific. So the idea of channeling and restricting this richness into smaller and simplified subsets of the language was tried: Scholarly HTML, RASH and ADF, for instance, are simple and rigid subsets of HTML5 provided with similar approaches towards a restricted vocabulary and syntax. PhiloEditor®[1] started as a web-based tool that identifies variants and versions of literary texts using such descriptive richness as a mechanism for creating and perusing complex, multiform and coexisting interpretative pathways over literary collections. It has been used for three years now at the Departments of Italian Studies of the University of Rome and at the University of Bologna, in order to study and characterize the differences between earlier and later versions of the same literary texts. PhiloEditor® was originally meant to provide a visualization of the differences between two versions of the same text, as a display of the output of a diff tool over two text documents. To ease the development of this tool, a simplified form of HTML5 immediately displayable in a browser window was used. Over time, in addition to the mere display of differences, a few additional interpretative tools were added to the interface to provide for highlighting of some semantically relevant textual features, rendered with different colors of the text. Finally, a few

[1] A demo is available at http://site1705.web.cs.unibo.it/phed6.2/#.

extractors of data were created to collect and graphically represent some statistics of features in the annotations and in the very text, exploiting the regularities in the controlled use of simplified HTML markup. The purpose of the tool, therefore, has shifted from the merely philological characterization of literary texts (i.e., to represent their temporal evolution) to a critical and hermeneutic approach to them, by providing support for custom, discipline-specific pathways over complex, multidimensional and temporally complex collections of (literary) documents. At the same time, the minimality of the markup language of the tool (relying exclusively on a well-behaved and extremely simplified subset of HTML) allows an extreme homogeneity and control over markup without the necessity of giving up in richness of features. Texts annotated through this simple format are extremely regular and can be exported to a well-known literary XML vocabulary for literary texts such as XML/TEI.

2 Markup patterns in simplified HTML

The pattern theory of documents (Di Iorio *et al.*, 2014) has been created to provide a generative approach to useful patterns in document structures. The advantages in restricting elements to patterns are to achieve *orthogonality*, as each pattern has a specific goal, and fits a specific context with no overlap with other patterns, and *assemblability*, as it is clearly associated to precise nesting rules. By limiting the possible choices, patterns prevent the spread of arbitrary structures and allow authors to create unambiguous, manageable and well-structured markup languages and, consequently, documents which drive reusability and homogeneity (Di Iorio *et al.*, 2012; Di Iorio *et al.*, 2013).

In our theory of patterns applied to PhiloEditor®, elements are divided into four categories according to two axes: text/no-text content and element/no element content. This creates a basic scaffolding of four categories: *marker* is the class of elements that can contain neither text nor other elements (i.e., empty elements), *flats* are the elements that can only contain text, *buckets* are the elements that can only contain other elements, and *mixed* is the class of elements that can contain both text and other elements.

	Cannot contain text	**Can contain text**
Cannot contain elements	Empty element (*Marker*)	Plain text element (*Flat*)
Can contain elements	Plain structural element (*Bucket*)	Both elements and text (*Mixed*)

Figure 1. Main categories of our pattern theory of documents.

We now switch to consider where these elements can be used, that is the context model that can accommodate them. Since only two categories can contain other elements, we have exactly eight patterns of elements.

	Marker	**Flat**	**Bucket**	**Mixed**
Marker	-	-	-	-
Flat	-	-	-	-
Bucket	Meta	Field	Container	Block
Mixed	Milestone	Atom	Popup	Inline

Figure 2. The basic set of patterns.

According to this model:

- A *meta* element is a marker (i.e., an empty element) placed within a bucket, i.e., never close to text fragments. Usually their real position is not relevant, and only its existence has some relevance: it is the

perfect candidate for metadata structures, hence its name. Element <meta> in HTML is clearly and rightly a meta element.

- A *milestone* is a marker (i.e., an empty element) placed within a mixed, i.e., close to the text fragments. Its location in the document is often its most important contribution, i.e. it represents a special position within of the document, hence its name. Elements
 and <img> in HTML are milestones.
- A *field* is a flat element (text allowed, elements not allowed) placed within a bucket, i.e., never close to text fragments. It is just a container of data whose position is not particularly relevant. Its proper use is as a field of a record, hence the name. Element <title> in HTML is a field.
- An *atom* is a flat element (text allowed, elements not allowed) placed within a mixed, i.e., close to text fragments. Its location in the document is often its most important contribution. Since it allows no element content, it is an atomic container of text, hence its name.
- A *container* is a bucket (text not allowed, elements allowed) placed within a bucket. It is a container of elements and it provides the fundamental scaffolding of the document structure. Containers are arguably the most important pattern of the eighth. For instance, elements <html> and <table> in HTML are containers, but there are many others.
- A *popup* is a bucket (text not allowed, elements allowed) placed within a mixed element. It provides an interruption of the usual flow of inline and text elements within a mixed element, creating a separate context for buckets such as containers. It acts as a frontier element between mixed and buckets, and lets the contained elements jump out of the constraints of mixed elements, hence its name. Element <figure> in HTML is a popup.
- A *block* is a mixed element (text and elements are both allowed) placed within a bucket. It is the earliest element in a hierarchy of containment of buckets that can contain text, and therefore acts as a frontier element between buckets and mixed. The most important type of blocks is the paragraph, i.e. a basic, independent container of text and inline elements of a document, a block of text vertically separated from the others of the same type, hence its name. Element <p> in HTML is therefore a block.
- An *inline* is a mixed (text and elements are both allowed) placed within a mixed. An inline element characterizes text fragments according to several criteria, such as style, typography, semantics, etc. Since it does not break the paragraph structure, it stays on the same line as the text, hence its name. Elements <b> or <a> in HTML are inlines, as well as many others.

These eight patterns represent by construction the complete set of possible element types, and no others can exist without loosen the rules. However we have identified some specific regularities within containers, which are very important and complex elements. Rather than creating independent patterns, we identified sub-patterns of containers, inheriting all their characteristics and adding others: *record, table,* hierarchical container (*hcontainer*).

This theory shows that although HTML does not recognize nor use patterns in a systematic way, it is possible to restrict HTML by selecting a reasonable subset of elements expressive enough to capture the typical components of domain-specific documents while being also well-designed, easy to reuse and robust.

3 Simplified HTML for scholarly, technical and literary texts

The discussion to adopt HTML as the markup language of choice even for specialized domain-dependent documents started in the academic publishing domain: through the unification of data formats we can homogenize the process of drafting, submitting and publishing documents. HTML easily supports the embedding of semantic annotations to improve the sharing of communication results, thanks to already existing W3C standards such as RDFa (Sporny, 2015) or JSON-LD (Sporny *et al.*, 2014), so that discoverability, interactivity, openness and usability of the scientific works are increased (Shotton *et al.*, 2009). The HTML language has been widely used as a full-fledged markup language to encode texts, not only plain Web pages but also academic articles, technical documentation, literary artefacts and data reports. The ability to embed semantics within HTML pages, for instance by using RDFa (Herman *et al.*, 2015), is a key factor for this success since it allows designers to express rich

information about any domain and to overcome the limitations of a language developed and used for long time for other purposes. HTML has recently gained importance as a markup language for writing technical documentation as well. ADF (Caponi *et al.*, 2018) is a pattern-based subset of HTML that allows designers to express information about the authoring process such as change-tracking data, templates and reusable fragments and authorship attribution data. The format can be manipulated by a Web editor that relies on the pattern-based structure of the language and its ability to express fine-grained semantic information on each piece of content.

4 PhiloEditor® for the scholarly markup of literary documents

PhiloEditor®, as a web-based environment for reading and annotating variants, is aimed to be a valuable support for scholars in the criticism of variants. Its main characteristics are the display of differences (*deltas*) between two versions of the same document and the classification of semantically relevant fragments of the text (*colouring*), according to a parametric set of features provided by a scholar in a specified domain. PhiloEditor®'s data model is based on a highly simplified version of HTML5 that is fully based on the pattern model described in section 2. PhiloEditor®'s features make it particularly suitable for the study of literary texts in multiple versions, in particular those resulting from the shifts of volition of the author (authorial philology). In fact, it has been tested first on the two printed editions of *I Promessi Sposi* by Alessandro Manzoni (Bonsi *et al.*, 2015), then on the internationally-known *Pinocchio* by Carlo Collodi (Gargano and Italia, 2018), comparing the version published periodically between 1881 and 1883 on "Giornale per i bambini" magazine with the printed edition published by Paggi in 1883.

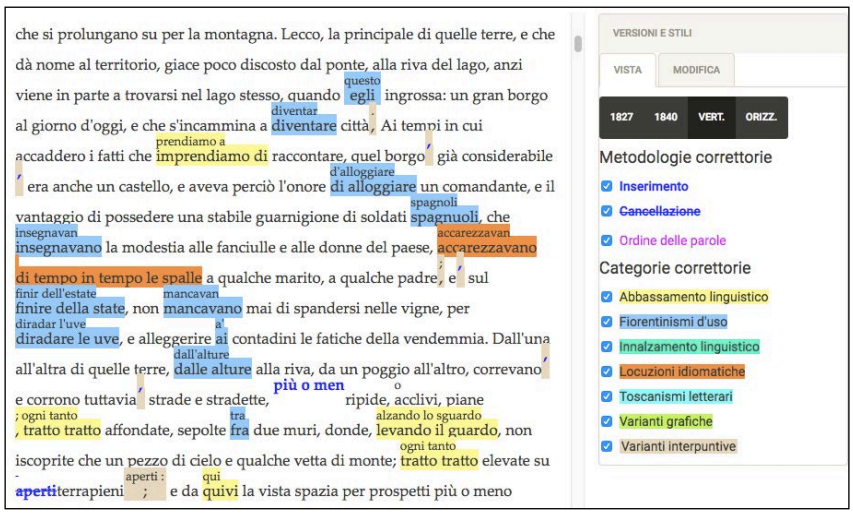

Figure 3. Color coding of the edits in *I Promessi Sposi*.

For versions and variants to be comparable, it's necessary for a delta to be created. Many algorithms exist that perform such tasks, but the kind of delta is also relevant for our purpose. Most algorithms meant for computer code assume that the relevant atomic entity to be differentiated is the individual line, since most programming languages use lines as their atomic semantic unit of production. In other types of documents, atomic entities may be single characters or structural nodes (e.g., a paragraph or a whole chapter). These are not particularly appropriate for literary texts, where long paragraphs with many differences (as in the case of our documents) would make the understanding of the variants difficult. The best choice for literary texts in our opinion is word-based diffing, which identifies whole words as the basic perceivable and understandable difference between variants. We thoroughly tested two Javascript libraries for word-based diffing, Javascript diff algorithm (Resig, 2005) and wikEd diff (Cacycle, 2017). The final choice was the latter, which is more precise and better parameterized. This algorithm reads two text files and outputs a document comparing them by placing the differences as spans of an HTML document. The output needs to be further synthetized and qualified: in particular,

wikEd diff generates a simple list of insertions and deletions. Literary variants do in fact contain relatively few simple deletions and insertions, and mostly replacements (where an old fragment is substituted by a new fragment). A replacement is seen by the diff algorithm as a deletion of the old text followed, in the same position, by an insertion of the new text. Correctly identifying and characterizing replacements is important for a better description of the actual editing operation occurred and this is generated by appropriately grouping the list of insertions and deletions provided by the diff algorithm before the result is displayed to the user. PhiloEditor® provides different ways to display variants: the user can choose an *exclusive*, a *synoptic* or a *layered* representation of the text. In these views, all modifications are characterized as replacements. Complex diff issues can be handled manually: the segment of the variants automatically created can be modified thanks to a simple interface operation that makes possible to divide or combine groups of elements according to the user's needs. Word-based diffing doesn't work with macro-variants though, so for now PhiloEditor® can only manage micro-variants.

The increased familiarity with the temporal evolution of the text brought a growth in curiosity towards the types of modifications. For this reason, we devised a two-layer model of modification types (Italia, 2018): "categories of corrections" to identify the actual methodology of correction (i.e. shifts of textual passages, additions, deletions, inversions of words, etc.) and "linguistic categories" to identify the linguistic categories that drove the correction. In order to describe the text in these terms, a color-based coding schema of the fragments of text affected by each phenomenon was created. Since the two classes could overlap, two separate types of styles were created using the color of the text (categories of correction) and the color of the background (linguistic categories). Displaying these changes visually enables us to reach a more complex perception of linguistic features of literary texts and facilitates not only the representation of important syntactic structures but also their hermeneutic implications.

The variety of textual features that need to be expressed in the markup of documents edited inside PhiloEditor® is limited: one needs to support the hierarchical structure of the document (chapters, paragraphs, text), very little typography (just a few words in italic here and there), plus the application-specific requirements of describing replacements (e.g., the edits generated by the diff engine as insertions and deletions and converted by an internal algorithm) and the colors, i.e., the linguistic categories described by scholars.

```
che si prolungano su per la montagna. Lecco, la principale di quelle terre, e che
dà nome al territorio, giace poco discosto dal ponte, alla riva del lago, anzi viene
in parte a trovarsi nel lago stesso, quando
    <span class="replace florentinism" data-responsible="Teresa">
        <span class="new">questo</span>
        <span class="old">egli</span>
    </span>
 ingrossa: un gran borgo al giorno d'oggi, e che s'incammina a
    <span class="replace florentinism" data-responsible="Teresa">
        <span class="new">diventar</span>
        <span class="old">diventare</span>
    </span>
 città
    <span class="replace punctuation" data-responsible="Teresa">
        <span class="new">.</span>
        <span class="old">,</span>
    </span>
 Ai tempi in cui accaddero i fatti che
    <span class="replace lowering" data-responsible="Teresa">
        <span class="new">prendiamo a</span>
        <span class="old">imprendiamo di</span>
    </span>
 raccontare, quel borgo
    <span class="replace insert punctuation" data-responsible="Teresa">
        <span class="new">,</span>
        <span class="old"> </span>
    </span>
 già considerabile
    <span class="replace insert punctuation" data-responsible="Teresa">
        <span class="new">,</span>
        <span class="old"> </span>
    </span>
 era anche un castello, e aveva perciò l'onore
    <span class="replace florentinism" data-responsible="Teresa">
        <span class="new">d'alloggiare</span>
        <span class="old">di alloggiare</span>
    </span>
 un comandante, e il vantaggio di possedere una stabile guarnigione di soldati
```

Figure 4. The HTML source of *I Promessi Sposi*.

Correspondingly, we have created an extreme simplification of the HTML5 used by the application: <section>, <p> and <i> are used for the document, and is used for both modifications and color coding (since most of the colors are applied to modifications anyway). HTML classes are created and

used both to provide semantics and rendering characteristics, and an additional data-* attribute is used to provide basic provenance attribution. Given that the same span can be associated to multiple classes, clashes in categorization are impossible. On the other hand, the HTML that is being created is rigorously well-formed and homogeneous, and generating a correct and valid XML/TEI source is immediate and straightforward.

Using HTML class names for semantic characterization has both advantages and issues. Of course, classes make the association of special rendering to fragments easy. On the other hand, it is extremely difficult to impose constraints on the list of classes that can be used, e.g., to specify within a schema that elements of the class "replace" must first contain an element with the class attribute containing "new" and then an element with the class attribute containing "old".

5 PhiloEditor® and Scholarly Editions: markup tools for literary texts

Most of the available online editions of literary texts are based on full-text transcriptions of original texts into electronic form (Franzini, 2012), typically using the XML/TEI model, where the sources (*witnesses* in philology jargon) are traditional primary sources. The infrastructure is generally based on standard web programming languages, both client- and server-side. Users have limited access to the digital sources without any awareness of the backend software employed. Most of the editions, as said, are XML/TEI documents, written by hand with a stand-alone application such as Oxygen, and converted as needed into HTML/CSS documents using XSLT stylesheets. All projects in this domain are built on the same basic structural components: they consist of a set of files (*assets*) stored inside an information architecture such as a database or file system (*structure*) where they can be accessed (*services*) and shown on a browser (*use/display*) (Druker *at al.*, 2014). Different phases (assets, structure, services, use/display) mean different tools. Thus, the DiRT Directory is a registry of digital research tools for scholarly use, while the Taxonomy of Digital Research Activities in the Humanities (TaDiRAH) "breaks down the research lifecycle into high-level 'goals', each with a set of 'methods' ".

In the field of literary texts, editing is the first important step of the process: editors need to use simple applications in order to describe their documents and their features. TextGrid, for instance, is a real research environment with all the necessary tools and services to support the entire research process, especially in digital scholarly editing. The downloadable "TextGridLaboratory" is an editor for XML/TEI markup with a view on the source code and a traditional visualization mechanism. Catma (Computer Assisted Text Markup and Analysis) is an online environment that manages analysis, manual and automatic annotations, and visualizations of documents. Juxta is an open-source tool for comparing and collating multiple witnesses of a text work, useful for an analytic visualization of textual variants. Tapas is meant for visualizing, storing and sharing XML/TEI documents, while EVT (Edition Visualization Technology) is a downloadable open source tool that creates digital editions from XML-encoded texts. The final styling of documents is entrusted to CSS style-sheets and is easily customizable.

As shown, XML/TEI is the fundamental data model for all documents, and it is used as a work format rather than as an output format, requiring scholars to learn it in depth. Simplified HTML could be an alternative that, while maintaining complete exportability to XML/TEI, allows application designers to rely on web technologies much easier to work with, and avoiding exposing literary scholars to angle brackets altogether.

6 Conclusions

PhiloEditor® is but one of the activities in which we are pushing for the use of a simple HTML instead of a full-fledged custom XML vocabulary, although striving to identify a well-formed, well-behaved, totally descriptive and specialized subset of HTML. In PhiloEditor® two specific markup needs, the description of modifications and the coloring of semantic characterization of the texts, have been expressed within a standard and very simple subset of HTML. Similar activities, such as RASH or ADF, are used in other completely different, but still very specialized, domains. Overall the response to the features and the flexibility of PhiloEditor® has been overwhelmingly positive. The objective of the next

future is to slowly convert PhiloEditor® into a full-scale environment for all the needs of the collection, digitization and publication of scholarly editions of literary texts.

References

Bonsi C, Di Iorio A, Italia P, Vitali F. 2015. *Manzoni's electronic interpretations*, in The Mechanic Reader. Digital methods for literary criticism, special issue of "Semicerchio", LIII (2015/2), pp. 91-99.

Cacycle. 2017. wikEdDiff, https://en.wikipedia.org/wiki/User:Cacycle/wikEdDiff

Caponi A, Di Iorio A, Vitali F, Alberti P and Scatá M. 2018. *Exploiting patterns and templates for technical documentation.* In DocEng '18: ACM Symposium on Document Engineering 2018, August 28–31, 2018, Halifax, NS, Canada. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3209280.3209537

Di Iorio A, Italia P, Vitali F. 2015. *Variants and Versioning between Textual Bibliography and Computer Science*, in F. Tomasi, R. Rosselli Del Turco, and A.M. Tammaro (Eds.), Humanities and Their Methods in the Digital Ecosystem. Proceedings of Third AIUCD Annual Conference (AIUCD2014). Selected papers. ACM, New York.

Di Iorio A, Peroni S, Poggi F, Vitali F, Shotton D. 2013. *Recognising document components in XML-based academic articles*. In: Proceedings of the 2013 ACM symposium on document engineering. New York. ACM. 181-184.

Di Iorio A, Peroni S, Poggi F, Vitali F. 2012. *A first approach to the automatic recognition of structural patterns in XML documents*. In: Proceedings of the 2012 ACM symposium on document engineering. New York. ACM. 85-94.

Di Iorio A, Peroni S, Poggi F, Vitali F. 2014. *Dealing with structural patterns of XML documents*. Journal of the American Society for Information Science and Technology 65(9):1884-1900.

Dirt Directory, https://dirtdirectory.org/

Drucker J., Kim D., Salehian I., Bushong A. 2014. *Introduction to Digital Humanities. Concepts, Methods, and Tutorials for Students and Instructors*. http://dh101.humanities.ucla.edu/?page_id=15

Edition Visualization Technology (EVT), http://evt.labcd.unipi.it/

Franzini, G. 2012-, *Catalogue of Digital Editions*. DOI: 10.5281/zenodo.1161425, https://dig-ed-cat.acdh.oeaw.ac.at/

Gargano T, Italia P. 2018. *Philoeditor 3.0: Pinocchio*, in P. Ponti e M. Marazzi (a cura di), *«Senza giudizio... e senza cuore»,* Atti del convegno di studi su Pinocchio, (18-19 maggio 2017 – Università Cattolica del Sacro Cuore e Università degli Studi di Milano), numero monografico della "Rivista di letteratura italiana", a. XXXVI, n. 2 (2018), pp. 133-144.

Herman I, Adida B, Sporny M, Birbeck M. 2015. RDFa 1.1 Primer - Third Edition Rich Structured Data Markup for Web Documents, W3C Working Group Note 17 March 2015.

Italia P. 2018. *Filologia d'autore digitale e multidiscipinare. Dall'Authorship alla Fotonica*, in G. Sampino, F. Scaglione (a cura di), Saperi Umanistici nella Contemporaneità, Atti del Convegno Internazionale dei dottorandi (17-18 settembre 2015 – Università degli Studi di Palermo), La Biblioteca di Classico Contemporaneo 6, Palumbo Editore, Palermo 2018, pp. 322-334.

Juxta, http://www.juxtasoftware.org/

Meister J.C., Petris M., Gius E., Jacke J. CATMA 5.0 (2016) [software for text annotation and analysis]: http://www.catma.de

Resig J. 2005. Javascript Diff Algorithm, https://johnresig.com/projects/javascript-diff-algorithm/

Shotton D, Portwin K, Klyne G, Miles A. 2009. Adventures in semantic publishing: exemplar semantic enhancements of a research article. PLOS Computational Biology 5(4):e1000361

Sporny M, Kellogg G, Lanthaler M. 2014. JSON-LD 1.0—a JSON-based serialization for linked data. W3C Recommendation 16 January 2014. World Wide Web Consortium. https://www.w3.org/TR/json-ld/

Sporny M. 2015. HTML+RDFa 1.1: support for RDFa in HTML4 and HTML5. W3C recommendation 17 March 2015. World Wide Web Consortium. http://www.w3.org/TR/rdfa-in-html/

TaDiRAH, https://dirtdirectory.org/tadirah

Tapas project, http://tapasproject.org/

TextGrid Consortium. 2006–2014. TextGrid: A Virtual Research Environment for the Humanities. Göttingen: TextGrid Consortium. textgrid.de