A Vision for User-Defined Semantic Markup

Michael Piotrowski
michael.piotrowski@unil.ch
University of Lausanne
Department for Language and Information Sciences
Lausanne, Switzerland

ABSTRACT

Typesetting systems, such as LaTeX, permit users to define custom markup and corresponding formatting to simplify authoring, ensure the consistent presentation of domain-specific recurring elements and, potentially, enable further processing, such as the generation of an index of such elements. In XML-based and similar systems, the separation of content and form is also reflected in the processing pipeline: while document authors can define custom markup, they cannot define its semantics. This could be said to be intentional to ensure structural integrity of documents, but at the same time it limits the expressivity of markup. The latter is particularly true for so-called lightweight markup languages like Markdown, which only define very limited sets of generic elements. This vision paper sketches an approach for user-defined semantic markup that could permit authors to define the semantics of elements by formally describing the relations between its constituent parts and to other elements, and to define a formatting intent that would ensure that a default presentation is always available.

CCS CONCEPTS

• **Applied computing** → **Markup languages**; *Format and notation*; *Document scripting languages*.

KEYWORDS

markup semantics, document models and structures, document authoring, scholarly publishing

ACM Reference Format:
Michael Piotrowski. 2019. A Vision for User-Defined Semantic Markup. In *ACM Symposium on Document Engineering 2019 (DocEng '19), September 23–26, 2019, Berlin, Germany*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3342558.3345414

1 INTRODUCTION

The first occurrence of a technical term is often highlighted, typically by setting it in italics. While visually indistinguishable from other uses of italics (e.g., book titles, foreign words, or general emphasis), it clearly has a different meaning. In traditional typesetting systems, such as LaTeX, it is easy for authors to make a distinction between emphasis and the first mention of a term in the document source, even if they are rendered in the same way:

```
\newcommand{\term}[1]{\emph{#1}}
```

Making this seemingly trivial distinction allows authors to change the presention of one of them without impacting the other, or to apply further processing, e.g., generate an index of terms. In typical XML-based workflows authors can theoretically introduce new markup elements, but XML does not provide the means to specify their semantics or, as authors generally do not have access to later stages of processing, their rendering.

This problem is even more acute in so-called *lightweight markup languages*, such as Markdown[1] and Org-mode [11]. Originally intended as just an easier way to write HTML, the capabilities of lightweight markup languages and the corresponding processors have grown significantly. Multi-format converters, such as Pandoc,[2] make lightweight markup languages also attractive for scholars: they allow them to write in a concise format and then easily produce the format required for submission, be it PDF, LaTeX, Microsoft Word, or JATS. This also makes it easier to resubmit papers to a different publication venue or to reuse content [5]. One significant drawback, however, is that authors are effectively limited to a small, fixed set of generic markup elements, and thus to using italics indifferently for emphasis, terms, book titles, etc.

The structural aspects of markup—sections, subsections, paragraphs, etc.—are more or less identical across markup vocabularies and can be mapped well by tools like Pandoc. What is lost, however, are the elements *specific* to particular domains, such as those specific to philology in TEI or to software documentation in DocBook. The standardization on the lowest common denominator helps interchange and conversion of documents, but prevents the use of "semantic" markup, which ensures consistent formatting and may express a meaning beyond formatting.

Whatever the discipline, research is characterized by analyzing specific topics in depth, combining existing knowledge with new ideas, defining new concepts and methods, and requiring new terminology and new notations. At the same time, different disciplines and areas of research also have very different requirements. It is thus illusionary to ever create a comprehensive markup vocabulary for scholarly publications; authors would thus ideally be able to define custom semantic markup in a *declarative* fashion; better yet, the definition would be understandable to different processors and could be automatically mapped to different target formats.

In the next section we introduce our conceptual approach to user-defined semantic markup. We will then briefly review related work, before we discuss the proposal and the required future work.

DocEng '19, September 23–26, 2019, Berlin, Germany

ACM ISBN 978-1-4503-6887-2/19/09…$15.00
https://doi.org/10.1145/3342558.3345414

[1] https://daringfireball.net/projects/markdown/
[2] https://pandoc.org

2 USER-DEFINED SEMANTIC MARKUP

2.1 Goals and Desiderata

As stated above, our primary goal is to give document authors a way to declaratively define custom markup so that it can both be sensibly formatted and semantically processed. The most important aspects are thus:

User-definable. Authors should be able to define custom markup on a per-document basis.

Declarative. The specification should be declarative rather than procedural, unlike macro definitions in traditional typesetting systems. With respect to formatting, authors should thus be able to specify a "formatting intent" that is sufficiently precise to permit a useful default rendering and sufficiently abstract to be adapted to different concrete layouts.

System-independent. Ideally, the same definition should be usable with XML, LaTeX, Pandoc, Org, etc., and only the concrete syntax would depend on the system used.

2.2 Conceptual Approach

As an example, consider glosses of foreign words, such as (underlined):

> … the French word *chaise* 'chair' is a feminine noun.

Such constructs occur frequently in linguistic discussions, but are not limited to linguistic texts; they can occur in all kinds of texts, from discussions of international law, philosophy, history, or engineering. However markup vocabularies rarely provide authors with dedicated elements; it is thus a good candidate for a user-defined element. If we start from the typography, a basic definition in LaTeX could be as simple as:

```
\newcommand{\gloss}[2]{\emph{#1} `#2'}
```

Since the term to be glossed is likely to be in a different language than the rest of the text, a better definition could be:

```
\newcommand{\gloss}[3]{
  \emph{\selectlanguage{#1}#2} `#3'}
```

This defines a macro that accepts three arguments:

(1) the language of the term to be glossed,
(2) the term itself, and
(3) the gloss.

The markup for the above example *chaise* 'chair' would thus look like:

```
\gloss{french}{chaise}{chair}
```

As already noted by Renear et al. [9], the meaning of document elements is to a large extent defined through their relations between their constituent parts, to other document elements, and to document-external entities. In this simple example we can observe the following relations between the three arguments (using the same numbering of arguments):

- arg_2 has-language arg_1
- arg_2 has-translation arg_3

In addition, one could say that the name of the language specified in arg_1 refers to some entitiy external to the document (a language), so one could add the relation

- arg_1 refers-to ⟨*language*⟩

If these relations were defined in a machine-readable way, and if the possible types of relations were known (i.e., there existed a controlled vocabulary of relations), a system could already recognize two different definitions of such an element as identical, regardless of names and order of arguments.

2.3 Describing the Formatting Intent

When we consider authoring, there is a an important type of processing that is often neglected: formatting. If users are permitted to define new elements, it is also necessary to have at least a sensible basic rendering. Again, this is easy to achieve in a typesetting system (such as LaTeX), but very difficult in an XSLT-based XML processing pipeline. Even when users are permitted to specify a "user stylesheet," it requires potentially very detailed knowledge of the standard processing. It would also be unlikely to be portable to other processing pipelines for the same document type. Processors for Markdown and other lightweight markup languages typically do not even have stylesheets but the conversion to the target format is embedded in the processor's code and neither visible nor alterable by authors.

What is needed, therefore, is also a way to describe a *formatting intent* that abstracts from the implementation details of a particular processor, but is detailed enough to permit a processor to create a useful rendering matching the rest of the document design. Ideally, such a description would be portable between different processors: one might map it to XSLT, another to CSS, yet another to LaTeX, etc. This task may at first be daunting, but the two-dimensional visual language of print-like documents is based on a very small number of basic elements, many of which already appear on this page: in the end, almost all semantically marked elements are rendered using some combination of type size and type style changes with respect to surrounding material and horizontal and vertical placement relative to surrounding material. The formatting of an element is thus similarly defined through relations to other elements.

In the example above, the formatting intent could specify that the element is to be rendered as an inline element consisting of the sequence of arg_2 and arg_3 (both inline elements as well), with arg_3 enclosed in inner quotes; the italicization of arg_2 could be derived from the fact that it is identified as being in a different language. While other ways of formatting are possible, such as

> … the French word "chaise" (chair) is a feminine noun.

in the absence of a more specific formatting (which could be identified as applicable by the semantics of the element) the suggested formatting would be perfectly adequate.

2.4 Putting It All Together

The above discussion suggests an interpretation of markup element as functions with a certain number of arguments whose semantics (or "type") is specified by relations between them (e.g., a translation of a string must also be a string) or to external entities (e.g., the value of an argument declared as a reference to a language must be one of the set of legal languages), yielding a formatting intent, i.e., a sequence of formatting elements. In a Lisp-like notation, this could be expressed roughly as follows:

```
(defelem gloss (lang foreign gloss)
  "Documentation of custom `gloss' element"
  ((has-language foreign lang)
   (has-translation foreign gloss)
   (refers-to lang iso639-language))
  ((inline-seq foreign (inner-q gloss))))
```

This example is merely for illustration and not meant to imply a specific implementation. Given the declarative approach and the central role of controlled vocabularies, we believe RDF could be a suitable framework for a concrete implementation.

Document processing systems would provide authors with a way to associate such definitions with a syntax specific to their markup languages, such as `<gloss>` in an XML-based system. They would parse this definition and map it to their internal representations and, for example, generate a corresponding stylesheet fragment, which is then included by the master stylesheet.

3 RELATED WORK

Related work is mainly found in two areas: (1) *semantic markup*, i.e., markup that aims to express a meaning beyond a particular presentation, and (2) *markup semantics*, i.e., the study of the meaning of markup and how this meaning can be formalized. The concept of semantic markup started to emerge in the late 1960s with the ideas of *generic coding* by William W. Tunnicliffe and Stanley Rice's *text format models* [10], which ultimately lead to the development of SGML [4] and thus XML. SGML and XML applications, however, only define the syntax of a markup language in a formal way; as Sperberg-McQueen et al. point out, "[i]n practice, the semantics of markup is most of the time specified only through human-readable documentation. Most existing colloquial markup languages are documented in prose, sometimes systematically and in detail, sometimes very sketchily." [14] This also applies to LaTeX markup, which is based on the same ideas; however, as its markup language is defined as macros in the TeX typesetting language, it does have executional semantics. Lightweight markup languages are typically defined by a mapping to some other markup language (most often HTML), so they inherit their semantics and only define a different syntax.

The vision of the Semantic Web and the needs of search engines have motivated research on the "semantic enhancement" of documents using RDF and OWL to provide formally defined semantics for selected content. Particularly noteworthy is the work based on the EARMARK markup language Peroni et al. [7, 8] and the Semantic Publishing and Referencing (SPAR) Ontologies [6], a modular suite of OWL ontologies for describing many aspects of semantic publishing and referencing in a machine-readable way using RDF. In our context, the Document Components Ontology (DoCO) [1] is of particular interest, as it defines a general-purpose structured vocabulary of document elements to describe both structural (e.g., block, inline, paragraph, section, chapter) and rhetorical (e.g., introduction, discussion, acknowledgements, reference list, figure, appendix) document components in RDF. This line of research has notably be applied in the context of the RASH framework [2] for semantic scholarly publishing.

The first and probably the most comprehensive theoretical investigation into markup semantics for XML was the BECHAMEL Markup Semantics Project [3, 9, 12, 13]. Noting that "there is no established machine-processable formalism that would support, for example, automatic inferences or checks on the satisfaction of semantic constraints," [3, p. 40] the project created a series of Prolog-based experimental systems for defining and reasoning about the meaning of markup. Dubin et al. [3, p. 41–42] give five examples of problems caused by the lack of formally described semantics for markup:

(1) Class relationships: there is no way to subtype elements, for example, to indicate that `<bold>` *is-a* emphasis.
(2) Propagation: there is no way to formally specify which properties of elements are propagated to child elements. For example, the XML Specification defines that the value of the `xml:lang` attribute is propagated, but there is no way for schema designers to define this behavior for other attributes.
(3) Context and reference: there is no way to specify semantic relations between elements; for example, that a `<title>` child element of a `<chapter>` element is meant to give the title of the chapter (and may give the title of a book or of a person in a different context).
(4) Ontological variation in reference: This refers to the fact that, if `<a>`, `<b>`, and `<c>` are meant to convey a certain property of their content, what is the meaning of something like `<a><b><c>x</c></b></a>`? There is no way to tell whether *x* has the properties *a*, *b*, and *c*, or whether *x* has only *c*, and *b* qualifies *c*.
(5) Full and partial synonymy: There is no way to formally indicate or determine whether two elements are full or partial synonyms, i.e., whether they have the same or almost the same meaning.

The Darwin Information Typing Architecture[3] (DITA) is an XML-based approach for large-scale technical documentation authoring, management, and delivery. The most notable feature of DITA in our context is *specialization*, which allows for the derivation of new element types and attributes from existing ones. This feature addresses point (1) above with the goal of facilitating interchange of content and ensuring a minimum level of common processing for all DITA documents, while permitting specialization-aware processors to add specialization-specific processing to the generic processing. However, specialization is not designed for use by authors, but to permit "information architects" to create company- or industry-specific versions of the DITA schemas. The specialized processing also has to be defined and implemented separately in the XML pipeline.

The work summarized above provides us with important foundations for our vision but addresses different use cases; in particular, we are not aware of any approaches that (1) permit authors to define custom elements in a semantic fashion and (2) to also specify the intended formatting in a declarative way.

4 DISCUSSION

While motivated by some of the same considerations as the work discussed in Section 3, our approach differs in that it considers syntax, semantics, and presentation as closely related rather than strictly separated. Rather than annotating a pre-existing syntactic

[3] https://www.oasis-open.org/committees/dita/

structure with semantics, we seek to come up with a way to allow document authors to formally define the semantics and the intended rendering of document elements, which can then be associated with an arbitrary syntax.

Our approach consequently places less importance on syntactic structures and validation. Syntactic validity is important for some types of applications, but for most narrative document types, notably scholarly publications, the structures tend to be trivial: sections and subsections with titles, paragraphs, lists, figures, and a couple of other elements, which carry little meaning on their own.

Instead, our proposed approach relies on three controlled vocabularies to formalize the relations that describe the semantics and the formatting intent of elements:

(1) A vocabulary of relations between the arguments of markup elements;
(2) a vocabulary of relations of arguments to external entities such as languages, scripts, notations, URIs, person, or places.
(3) an abstract layout vocabulary to describe the relations between layout elements.

For all three vocabularies, the Document Components Ontology (DoCO, see Section 3) and the Pattern Ontology,[4] which it imports, could serve as a basis and starting point.

As outlined above, our approach currently focuses on *individual* custom elements. Conceptually, it is straightforward to integrate custom elements into the default processing; for example, if an element has an argument identified as *label*, cross-referencing should work in the same way as for standard elements. However, we have intentionally excluded further processing, as this would raise questions that cannot be reasonably discussed in a short paper.

For example, if one can define a `<term>` element to mark technical terms, it seems reasonable to expect that a glossary (i.e., a list of terms) can be generated. This may be done by defining a `<glossary>` element whose semantics are defined in relation to individual `<term>` elements (e.g., is-list-of), with a corresponding formatting intent. However, even the creation of basic indexes requires various types of sorting and possibly text and structure transformations. While certainly useful, a general implementation might quickly approach the capabilities of languages like XSLT.

On the other hand, formatted documents are built from a relative small set of basic structures; index-like lists of items could be considered such a structure. This would permit a user to simply define a `<glossary>` as a kind of index, which collects `<term>` elements according to one of a number of predefined schemes. While somewhat less general, this approach might offer a better balance between expressiveness and complexity.

A key question is thus how to identify the optimal balance, the "sweet spot." Among the potential factors to consider we want to hightlight just three. (1) Security: if documents containing custom elements are exchanged—and this is part of our motivation—it must be guaranteed that they are computationally well-behaved and that custom elements do not open new attack vectors. (2) Application domain: our approach is intentionally "document centric" and based on knowledge about documents. (3) Target users: we aim to enable *authors* to define custom markup. This is obviously not a complete assessment, but in our opinion, these factors at least strongly suggest a strictly declarative approach.

[4] https://essepuntato.github.io/po/current/pattern.html

5 CONCLUSION

We have outlined a vision for user-defined semantic markup as a basis for discussion in the document engineering community in order to gather feedback and further potential use cases. In particular, the definition of common vocabularies for both semantic and layout relations can only be successful if it is based on a wide consensus.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable feedback.

REFERENCES

[1] Alexandru Constantin, Silvio Peroni, Steve Pettifer, David Shotton, and Fabio Vitali. 2016. The Document Components Ontology (DoCO). *Semantic Web* 7, 2 (2016), 167–181. https://doi.org/10.3233/sw-150177

[2] Angelo Di Iorio, Andrea G. Nuzzolese, Francesco Osborne, Silvio Peroni, Francesco Poggi, Michael Smith, Fabio Vitali, and Jun Zhao. 2015. The RASH Framework: Enabling HTML+RDF Submissions in Scholarly Venues. In *Proceedings of the ISWC 2015 Posters & Demonstrations Track*, Serena Villata, Jeff Z. Pan, and Mauro Dragoni (Eds.). CEUR, Aachen, 4. http://ceur-ws.org/Vol-1486/paper_72.pdf

[3] David Dubin, Allen Renear, C. M. Sperberg-McQueen, and Claus Huitfeldt. 2003. A Logic Programming Environment for Document Semantics and Inference. *Literary and Linguistic Computing* 18, 1 (2003), 39–47. https://doi.org/10.1093/llc/18.1.39

[4] Charles F. Goldfarb. 1997. SGML: The reason why and the first published hint. *Journal of the American Society for Information Science* 48, 7 (1997), 656–661. https://doi.org/10.1002/(sici)1097-4571(199707)48:7%3C656::aid-asi13%3E3.0.co;2-t

[5] Albert Krewinkel and Robert Winkler. 2017. Formatting Open Science: agilely creating multiple document formats for academic manuscripts with Pandoc Scholar. *PeerJ Computer Science* 3 (May 2017), e112. https://doi.org/10.7717/peerj-cs.112

[6] Silvio Peroni. 2014. The Semantic Publishing and Referencing Ontologies. In *Semantic Web Technologies and Legal Scholarly Publishing*. Law, Governance and Technology Series, Vol. 15. Springer, Cham, Switzerland, 121–193. https://doi.org/10.1007/978-3-319-04777-5_5

[7] Silvio Peroni, Aldo Gangemi, and Fabio Vitali. 2011. Dealing with Markup Semantics. In *Proceedings of the 7th International Conference on Semantic Systems (I-Semantics '11)*. ACM, New York, NY, USA, 111–118. https://doi.org/10.1145/2063518.2063533

[8] Silvio Peroni, David Shotton, and Fabio Vitali. 2012. Faceted documents: describing document characteristics using semantic lenses. In *Proceedings of the 2012 ACM symposium on Document engineering (DocEng '12)*. ACM, New York, NY, USA, 191–194. https://doi.org/10.1145/2361354.2361396

[9] Allen H. Renear, David Dubin, and C. M. Sperberg-McQueen. 2002. Towards a semantics for XML markup. In *Proceedings of the 2002 ACM symposium on Document engineering (DocEng '02)*. ACM, New York, NY, USA, 119–126. https://doi.org/10.1145/585058.585081

[10] Stanley Rice. 1978. *Book Design: Text Format Models*. Bowker, New York, NY, USA.

[11] Eric Schulte, Dan Davison, Thomas Dye, and Carsten Dominik. 2012. A Multi-Language Computing Environment for Literate Programming and Reproducible Research. *Journal of Statistical Software* 46, 3 (25 1 2012), 1–24. https://doi.org/10.18637/jss.v046.i03

[12] C. M. Sperberg-McQueen, David Dubin, Claus Huitfeldt, and Allen H. Renear. 2002. Drawing inferences on the basis of markup. In *Proceedings of Extreme Markup Languages 2002*, B. Tommie Usdin and Steven R. Newcomb (Eds.). 20. http://conferences.idealliance.org/extreme/html/2002/CMSMcQ01/EML2002CMSMcQ01.html

[13] C. M. Sperberg-McQueen, Claus Huitfeldt, and Allen H. Renear. 2000. Meaning and interpretation of markup. *Markup Languages: Theory & Practice* 2, 3 (2000), 215–234. http://cmsmcq.com/2000/mim.html

[14] C. M. Sperberg-McQueen, Yves Marcoux, and Claus Huitfeldt. 2010. Two representations of the semantics of TEI Lite. In *Digital Humanities 2010 Conference Abstracts*. ADHO, 244–245. http://dh2010.cch.kcl.ac.uk/academic-programme/abstracts/papers/html/ab-663.html