Semantic Web 0 (2014) 1–17
IOS Press

WYSIWYM – Integrated Visualization, Exploration and Authoring of Semantically Enriched Un-structured Content

Editor(s): Roberto Garcia, Universitat de Lleida, Spain; Heiko Paulheim, University of Mannheim, Germany; Paola Di Maio, Universal Interfaces Research Lab, ISTCS, Edinburgh, UK
Solicited review(s): Heiko Hornung, University of Campinas, Brazil; Haofen Wang, Shanghai Jiao Tong University, China; one anonymous reviewer

Ali Khalili [a], Sören Auer [b]
[a] *AKSW, Universität Leipzig, Augustusplatz 10, 04109 Leipzig, Germany*
`khalili@informatik.uni-leipzig.de`
[b] *CS/EIS, Universität Bonn, Römerstraße 164, 53117 Bonn, Germany*
`auer@cs.uni-bonn.de`

Abstract. The Semantic Web and Linked Data gained traction in the last years. However, the majority of information still is contained in unstructured documents. This can also not be expected to change, since text, images and videos are the natural way how humans interact with information. Semantic structuring on the other hand enables the (semi-)automatic integration, repurposing, rearrangement of information. NLP technologies and formalisms for the integrated representation of unstructured and semantic content (such as RDFa and Microdata) aim at bridging this semantic gap. However, in order for humans to truly benefit from this integration, we need ways to author, visualize and explore unstructured *and* semantically enriched content in an integrated manner. In this paper, we present the WYSIWYM (What You See is What You Mean) concept, which addresses this issue and formalizes the binding between semantic representation models and UI elements for authoring, visualizing and exploration. With RDFaCE, Pharmer and conTEXT we present and evaluate three complementary showcases implementing the WYSIWYM concept for different application domains.

Keywords: Visualization, Authoring, Exploration, Semantic Web, WYSIWYM, WYSIWYG, Visual Mapping

1. Introduction

The Semantic Web and Linked Data movements with the aim of creating, publishing and interconnecting machine readable information have gained traction in the last years. However, the majority of information still is contained in and exchanged using unstructured documents, such as Web pages, text documents, images and videos. This can also not be expected to change, since text, images and videos are the natural way how humans interact with information. Semantic structuring on the other hand provides a wide range of advantages compared to unstructured information. It facilitates a number of important aspects of information management:

- For *search and retrieval* enriching documents with semantic representations helps to create more efficient and effective search interfaces, such as faceted search [33] or question answering [20].
- In *information presentation* semantically enriched documents can be used to create more sophisticated ways of flexibly visualizing information, such as by means of semantic overlays as described in [2].

– For *information integration* semantically enriched documents can be used to provide unified views on heterogeneous data stored in different applications by creating composite applications such as semantic mashups [1].
– To realize *personalization*, semantically enriched documents provide customized and context-specific information which better fits user needs and will result in delivering customized applications such as personalized semantic portals [29].
– For *reusability* and *interoperability* enriching documents with semantic representations facilitates exchanging content between disparate systems and enables building applications such as executable papers [24].

Natural Language Processing (NLP) technologies (e.g. named entity recognition and relationship extraction) as well as formalisms for the integrated representation of unstructured and semantic content (such as *RDFa* and *Microdata*) aim at bridging the semantic gap between unstructured and semantic representation formalisms. However, in order for humans to truly benefit from this integration, we need ways to author, visualize and explore unstructured *and* semantically enriched content in an integrated manner.

In this paper, we present an approach inspired by the WYSIWYM metaphor (What You See Is What You Mean), which addresses the issue of an integrated visualization, exploration and authoring of semantically enriched un-structured content. Our WYSIWYM concept formalizes the binding between semantic representation models and UI elements for authoring, visualizing and exploration. We analyse popular tree, graph and hyper-graph based semantic representation models and elicit a list of semantic representation elements, such as entities, various relationships and attributes. We provide a comprehensive survey of common UI elements for authoring, visualizing and exploration, which can be configured and bound to individual semantic representation elements. Our WYSIWYM concept also comprises cross-cutting helper components, which can be employed within a concrete WYSIWYM interface for the purpose of automation, annotation, recommendation, personalization etc.

With *RDFaCE*, *Pharmer* and *conTEXT* we present and evaluate three complementary showcases implementing the WYSIWYM concept for different domains. RDFaCE is a domain agnostic editor for text content with embedded semantic in the form of RDFa or Microdata. Pharmer is a WYSIWYM interface for the authoring of semantic prescriptions and thus targeting the medical domain. conTEXT is a Linked-Data based lightweight text anlaytics platform supporting different views for semantic analytics. Our evaluation of these tools with end-users (in case of RDFaCE and conTEXT) and domain experts (in case of Pharmer) shows, that WYSIWYM interfaces provide good usability, while retaining benefits of a truly semantic representation.

The contributions of this work are in particular:

1. A formalization of the WYSIWYM concept based on definitions for the WYSIWYM model, binding and concrete interfaces.
2. A survey of semantic representation elements of tree, graph and hyper-graph knowledge representation formalisms as well as UI elements for authoring, visualization and exploration of such elements.
3. Three complementary use cases, which evaluate different, concrete WYSIWYM interfaces in a generic as well as domain specific context.

The WYSIWYM formalization can be used as a basis for implementations; allows to evaluate and classify existing user interfaces in a defined way; provides a terminology for software engineers, user interface and domain experts to communicate efficiently and effectively. We aim to contribute with this work to making Semantic Web applications more user friendly and ultimately to create an ecosystem of flexible UI components, which can be reused, repurposed and choreographed to accommodate the UI needs of dynamically evolving information structures.

The remainder of this article is structured as follows: In Section 2, we describe the background of our work and discuss the related work. Section 3 describes the fundamental WYSIWYM concept proposed in the paper. Subsections of Section 3 present the different components of the WYSIWYM model. In Section 4, we introduce three implemented WYSIWYM interfaces together with their evaluation results. Finally, Section 5 concludes with an outlook on future work.

2. Related Work

WYSIWYG. The term *WYSIWYG* as an acronym for What-You-See-Is-What-You-Get is used in computing to describe a system in which content (text and graphics) displayed on-screen during editing appears in a form closely corresponding to its appearance when printed or displayed as a finished product. The first usage of the term goes back to 1974 in the print industry to express the idea that what the user sees on the screen is what the user gets on the printer. Xerox PARC's Bravo was the first WYSIWYG editor-formatter [23]. It was designed by Butler Lampson and Charles Simonyi who had started working on these concepts around 1970 while at Berkeley. Later on by the emergence of Web and HTML technology, the WYSIWYG concept was also utilized in Web-based text editors. The aim was to reduce the effort required by users to express the formatting directly as valid HTML markup. In a WYSIWYG editor users can edit content in a view which matches the final appearance of published content with respect to fonts, headings, layout, lists, tables, images and structure. Because using a WYSIWYG editor may not require any HTML knowledge, they are often easier for an average computer user to get started with. The first programs for building Web pages with a WYSIWYG interface were Netscape Gold, Claris HomePage, and Adobe PageMill.

WYSIWYG text authoring is meanwhile ubiquitous on the Web and part of most content creation and management workflows. It is part of content management cystems (CMS), weblogs, wikis, fora, product data management systems and online shops, just to mention a few. However, the WYSIWYG model has been criticized, primarily for the verbosity, poor support of semantics and low quality of the generated code and there have been voices advocating a change towards a *WYSIWYM* (What-You-See-Is-What-You-Mean) model [32,30].

WYSIWYM. The first use of the WYSIWYM term occurred in 1995 aiming to capture the separation of presentation and content when writing a document. The *LyX editor*[1] was the first WYSIWYM word processor for structure-based content authoring. Instead of focusing on the format or presentation of the document, a WYSIWYM editor preserves the intended meaning of each element. For example, page headers, sections, paragraphs, etc. are labeled as such in the editing program, and displayed appropriately in the browser. Another usage of the WYSIWYM term was by Power et al. [28] in 1998 as a solution for *Symbolic Authoring*. In symbolic authoring the author generates language-neutral "symbolic" representations of the content of a document, from which documents in each target language are generated automatically, using *Natural Language Generation* technology. In this What-You-See-Is-What-You-Meant approach, the language generator was used to drive the user interface (UI) with support of localization and multilinguality. Using the WYSIWYM natural language generation approach, the system generates a feed-back text for the user that is based on a semantic representation. This representation can be edited directly by the user by manipulating the feedback text.

The WYSIWYM term as defined and used in this paper targets the novel aspect of integrated visualization, exploration and authoring of unstructured and semantic content. The rationale of our WYSIWYM concept is to enrich the existing WYSIWYG presentational view of the content with UI components revealing the *semantics* embedded in the content and enable the exploration and authoring of semantic content. Instead of separating presentation, content and meaning, our WYSIWYM approach aims to integrate these aspects to facilitate the process of *Semantic Content Authoring*. Two "You"s in our WYSIWYM concept refer to the end user (with no or limited knowledge of Semantic Web) who is viewing an unstructured content which is semantically enriched by himself. The "Mean" refers to the metadata or semantics which is encoded in the unstructured content viewed by user. There are already some approaches (i.e. visual mapping techniques), which go into the direction of integrated visualization and authoring of structured content.

Visual Mapping Techniques. Visual mapping techniques are knowledge representation techniques that graphically represent knowledge structures. Most of them have been developed as paper-based techniques for brainstorming, learning facilitation, outlining or to elicit knowledge structures. According to their basic topology, most of them can be related to the following fundamentally different primary approaches [5,31]:

- *Mind-Maps*. Mind-maps are created by drawing one central topic in the middle together with labeled branches and sub-branches emerging from it. Instead of distinct nodes and links, mind-maps only have labeled branches. A mind-map is a connected directed acyclic graph with hierarchy as its only type of relation. *Outlines* are a similar technique to show hierarchical relationships using tree structure. Mind-maps and outlines are not

[1] `http://www.lyx.org/`

suitable for relational structures because they are constrained to the hierarchical model.

– *Concept Maps*. Concept maps consist of labeled nodes and labeled edges linking all nodes to a connected directed graph. The basic node and link structure of a connected directed labeled graph also forms the basis of many other modeling approaches like *Entity-Relationship* (ER) diagrams and *Semantic Networks*. These forms have the same basic structure as concept maps but with more formal types of nodes and links.
– *Spatial Hypertext*. A spatial hypertext is a set of text nodes that are not explicitly connected but implicitly related through their spatial layout, e.g., through closeness and adjacency — similar to a pin-board. Spatial hypertext can show fuzzily related items. To fuzzily relate two items in a spatial hypertext schema, they are simply placed near to each other, but possibly not quite as near as to a third object. This allows for so-called "constructive ambiguity" and is an intuitive way to deal with vague relations and orders. Spatial Hypertext abandons the concept of explicitly interrelating objects. Instead, it uses spatial positioning as the basic structure.

Binding data to UI elements. There are already many approaches and tools which address the binding between data and UI elements for visualizing and exploring structured content. Dadzie and Rowe [3] present the most exhaustive and comprehensive survey to date of these approaches. For example, *Fresnel* [27] is a display vocabulary for core RDF concepts. Fresnel's two foundational concepts are lenses and formats. Lenses define which properties of an RDF resource, or group of related resources, are displayed and how those properties are ordered. Formats determine how resources and properties are rendered and provide hooks to existing styling languages such as CSS.
Parallax, *Tabulator*, *Explorator*, *Rhizomer*, *Sgvizler*, *Fenfire*, *RDF-Gravity*, *IsaViz* and *i-Disc for Topic Maps* are examples of tools available for visualizing and exploring structured data. In these tools the binding between semantics and UI elements is mostly performed implicitly, which limits their versatility. However, an explicit binding as advocated by our WYSIWYM model can be potentially added to some of these tools.

In contrast to the structured content, there are many approaches and tools which allow binding semantic data to UI elements within semantically enriched unstructured content (cf. our comprehensive literature study [11]). As an example, *Dido* [10] is a data-interactive document which lets end users author semantic content mixed with unstructured content in a web-page. Dido inherits data exploration capabilities from the underlying *Exhibit*[2] framework. *Loomp* as a prove-of-concept for the *One Click Annotation* [6] strategy is another example in this context. Loomp is a WYSIWYG web editor for enriching content with RDFa annotations. It employs a partial mapping between UI elements and data to hide the complexity of creating semantic data.

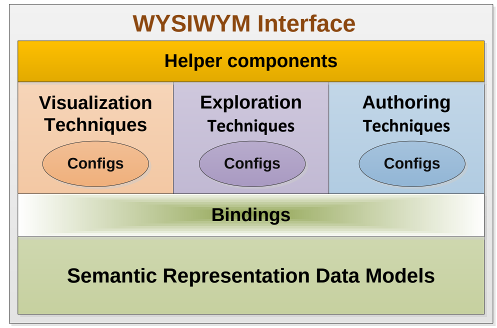

Fig. 1. Schematic view of the WYSIWYM model.

3. WYSIWYM Concept

In this section we introduce the fundamental WYSIWYM concept and formalize key elements of the concept. Formalizing the WYSIWYM concept has a number of advantages: First, the formalization can be used as a basis for design and implementation of novel applications for authoring, visualization, and exploration of semantic content (cf. Section 4). The formalization serves the purpose of providing a terminology for software engineers and UI designers to communicate efficiently and effectively. It provides insights into and an understanding of the requirements as well as corresponding UI solutions for proper design and implementation of semantic content management applications. Secondly, it allows to evaluate and classify existing user interfaces according to the conceptual model in a defined way. This will highlight the gaps in existing applications dealing with semantically enriched documents and will help to optimize them based on the defined requirements.

[2] `http://simile-widgets.org/exhibit/`

Figure 1 provides a schematic overview of the WYSIWYM concept. The rationale is that elements of a knowledge representation formalism (or data model) are connected to suitable UI elements for visualization, exploration and authoring. Formalizing this conceptual model results in three core definitions (1) for the abstract *WYSIWYM model*, (2) *bindings* between UI and representation elements as well as (3) a concrete instantiation of the abstract WYSIWYM model, which we call a *WYSIWYM interface*.

Definition 1 (WYSIWYM model)**.** *The WYSIWYM model can be formally defined as a quintuple* (D, V, X, T, H) *where:*

- D *is a set of semantic representation data models, where each* $D_i \in D$ *has an associated set of data model elements* E_{D_i}*;*
- V *is a set of tuples* (v, C_v)*, where* v *is a visualization technique and* C_v *a set of possible configurations for the visualization technique* v*;*
- X *is a set of tuples* (x, C_x)*, where* x *is an exploration technique and* C_x *a set of possible configurations for the exploration technique* x*;*
- T *is a set of tuples* (t, C_t)*, where* t *is an authoring technique and* C_t *a set of possible configurations for the authoring technique* t*;*
- H *is a set of helper components.*

Semantic representation data models are techniques to define the meaning of data within the context of its interrelationships with other data (cf. Section 3.1). Tree, Graph and Hypergraph are examples of commonly used data models. *Visualization techniques* include UI techniques for highlighting, associating and detail viewing of semantic entities (cf. Section 3.2). *Exploration techniques* include UI techniques for efficient browsing and navigating semantic data (cf. Section 3.3). *Authoring techniques* include UI techniques for adding and editing semantic entities and their relations (cf. Section 3.4). *Helper components* are cross-cutting aspects to enhance and customize the user/application requirements of a WYSIWYM interface (cf. Section 3.6).

The WYSIWYM model represents an abstract concept from which concrete interfaces can be derived by means of bindings between semantic representation model elements and configurations of particular UI elements.

Definition 2 (Binding)**.** *A binding* b *is a function which maps each element of a semantic representation model* e *(*$e \in E_{D_i}$*) to a set of tuples* (ui, c)*, where* ui *is a user interface technique* ui *(*$ui \in V \cup X \cup T$*) and* c *is a configuration* $c \in C_{ui}$*.*

Figure 4 gives an overview on all data model (columns) and UI elements (rows) and how they can be bound together using a certain configuration (cells). The shades of gray in a certain cell indicate the suitability of a certain binding between a particular UI and data model element.

For example, having *tree-based* semantic representation model, *framing and segmentation* UI techniques can be used as external augmentation to *visualize* the *items* in the *text*. It is also possible to use *text formatting* techniques as inline augmentation for highlighitng the items but since they might interfere with the current text format, we assume a partial binding for them. A possible configuration for this example binding is to set different border and text colors to distinguish different item types.

Once a selection of data models and UI elements was made and both are bound to each other encoding a certain configuration in a binding, we attain a concrete instantiation of our WYSIWYM model called WYSIWYM interface.

Definition 3 (WYSIWYM interface)**.** *An instantiation of the WYSIWYM model* I *called WYSIWYM interface now is a hextuple* $(D_I, V_I, X_I, T_I, H_I, b_I)$*, where:*

- D_I *is a selection of semantic representation data models (*$D_I \subset D$*);*
- V_I *is a selection of visualization techniques (*$V_I \subset V$*);*
- X_I *is a selection of exploration techniques (*$X_I \subset X$*);*
- T_I *is a selection of authoring techniques (*$T_I \subset T$*);*
- H_I *is a selection of helper components (*$H_I \subset H$*);*
- b_I *is a binding which binds a particular occurrence of a data model element to a visualization, exploration and/or authoring technique.*

Note, that we limit the definition to one binding, which means that only one semantic representation model is supported in a particular WYSIWYM interface at a time. It could be also possible to support several semantic representation models (e.g. RDFa and Microdata) at the same time. However, this can be confusing to the user, which is why we deliberately excluded this case in our definition. In the remainder of this sections we discuss the different parts of the WYSIWYM concept in more detail.

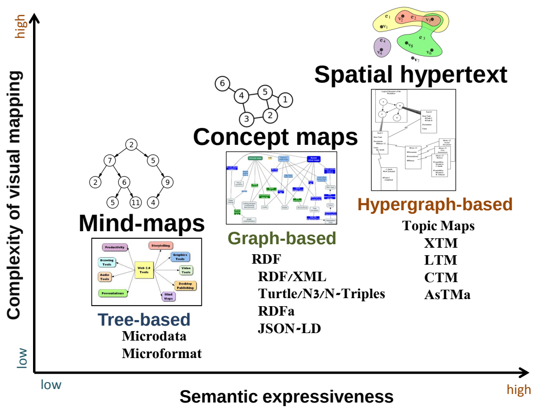

Fig. 2. Comparison of existing visual mapping techniques in terms of semantic expressiveness and complexity of visual mapping.

3.1. Semantic Representation Models

Semantic representation models are conceptual data models to express the meaning of information thereby enabling representation and interchange of knowledge. Based on their expressiveness, we can roughly divide popular semantic representation models into the three categories *tree-based*, *graph-based* and *hypergraph-based* (cf. Figure 2). Each semantic representation model comprises a number of representation elements, such as various types of entities and relationships. For visualization, exploration and authoring it is of paramount importance to bind the most suitable UI elements to respective representation elements. In the sequel we briefly discuss the three different types of representation models.

Tree-based. This is the simplest semantic representation model, where semantics is encoded in a tree-like structure. It is suited for representing taxonomic knowledge, such as thesauri, classification schemes, subject heading lists, concept hierarchies or mind-maps. It is used extensively in biology and life sciences, for example, in the *APG III system* (Angiosperm Phylogeny Group III system) of flowering plant classification, as part of the Dimensions of the *XBRL* (eXtensible Business Reporting Language) or generically in the *SKOS* (Simple Knowledge Organization System). Elements of tree-based semantic representation usually include:

- E_1: Item – e.g. `Magnoliidae`, the item representing all flowering plants.
- E_2: Item type – e.g. `biological term` for `Magnoliidae`.
- E_3: Item-subitem relationships – e.g. `Magnoliidae` referring to subitem `magnolias`.
- E_4: Item property value – e.g. the synonym `flowering plant` for the item `Magnoliidae`.
- E_5: Related items – e.g. the sibling item `Eudicots` to Magnoliidae.

Tree-based data can be serialized as Microdata or Microformats.

Graph-based. This semantic representation model adds more expressiveness compared to simple tree-based formalisms. The most prominent representative is the *RDF* data model, which can be seen as a set of triples consisting of subject, predicate, object, where each component can be a URI, the object can be a literal and subject as well as object can be a blank node. The most distinguishing features of RDF from a simple tree-based model are: the distinction of entities in classes and instances as well as the possibility to express arbitrary relationships between entities. The graph-based model is suited for representing combinatorial schemes such as concept maps. Graph-based models are used in a very broad range of domains, for example, in the *FOAF* (Friend of a Friend) for describing people, their interests and interconnections in a social network, in *MusicBrainz* to publish information about music albums, in the medical domain (e.g. DrugBank, Diseasome, ChEMBL, SIDER) to describe the relations between diseases, drugs and genes, or generically in the *SIOC* (Semantically-Interlinked Online Communities) vocabulary. Elements of RDF as a typical graph-based data model are:

- E_1: Instances – e.g. `Warfarin` as a drug.
- E_2: Classes – e.g. `anticoagulants drug` for Warfarin.
- E_3: Relationships between entities (instances or classes) – e.g. the `interaction` between `Aspirin` as an antiplatelet drug and `Warfarin` which will increase the risk of bleeding.
- E_4: Literal property values – e.g. the halflife for the Amoxicillin.
  - E_{4_1}: Value – e.g. `61.3` minutes.
  - E_{4_2}: Language tag – e.g. `en`.
  - E_{4_3}: Datatype – e.g. `xsd:float`.

RDF-based data can be serialized in various formats, such as RDFa, RDF/XML, JSON-LD or Turtle/N3/N-Triples.

Hypergraph-based. A hypergraph is a generalization of a graph in which an edge can connect any number of vertices. Since hypergraph-based models allow n-ary relationships between arbitrary number of nodes, they provide a higher level of expressiveness compared to tree-based and graph-based models. The most prominent representative is the *Topic Maps* data model developed as an ISO/IEC standard which consists of topics, associations and occurrences. The semantic expressivity of Topic Maps is, in many ways, equivalent to that of RDF, but the major differences are that Topic Maps (i) provide a higher level of semantic abstraction (providing a template of topics, associations and occurrences, while RDF only provides a template of two arguments linked by one relationship) and (hence) (ii) allow n-ary relationships (hypergraphs) between any number of nodes, while RDF is limited to triplets. The hypergraph-based model is suited for representing complex schemes such as spatial hypertext. Hypergraph-based models are used for a variety of applications. Amongst them are *musicDNA*[3] as an index of musicians, composers, performers, bands, artists, producers, their music, and the events that link them together, *TM4L* (Topic Maps for e-Learning), clinical decision support systems and enterprise information integration. Elements of Topic Maps as a typical hypergraph-based data model are:

- E_1: Topic name – e.g. `University of Leipzig`.
- E_2: Topic type – e.g. `organization` for University of Leipzig.
- E_3: Topic associations – e.g. `member of a project` which has other organization partners.
- E_4: Topic role in association e.g. `coordinator`.
- E_5: Topic occurrences – e.g. `address`.
  - E_{5_1}: value – e.g. `Augustusplatz 10, 04109 Leipzig`.
  - E_{5_2}: datatype – e.g. `text`.

Topic Maps-based data can be serialized as an XML-based syntax called XTM (XML Topic Map), LTM (Linear Topic Map Notation), CTM (Compact Topic Maps Notation) and AsTMa (Asymptotic Topic Map Notation).

3.2. Visualization

The primary objectives of visualization are to present, transform, and convert semantic data into a visual representation, so that, humans can read, query and edit them efficiently. We divide existing techniques for visualization of knowledge encoded in text, images and videos into the three categories *Highlighting*, *Associating* and *Detail view*. Highlighting includes UI techniques which are used to distinguish or highlight a part of an object (i.e. text, image or video) from the whole object. Associating deals with techniques that visualize the relation between some parts of an object. Detail view includes techniques which reveal detailed information about a part of an object. For each of the above categories, the related UI techniques are as follows:

- Highlighting.

- V_1: *Framing and Segmentation* (borders, overlays and backgrounds). This technique can be applied to text, images and videos, we enclose a semantic entity in a coloured border, background or overlay. Different border styles (colours, width, types), background styles (colours, patterns) or overlay styles (when applied to images and videos) can be used to distinguish different types of semantic entities (cf. Figure 3 no. 1, 2). The technique is already employed in social networking websites such as *Google Plus* and *Facebook* to tag people within images.
- V_2: *Text formatting* (color, font, size, margin, etc.). In this technique different text styles such as font family, style, weight, size, colour, shadows, margin and other text decoration techniques are used to distinguish semantic entities within a text (cf. Figure 3 no. 6). The problem with this technique is that in an HTML document, the applied semantic styles might overlap with existing styles in the document and thereby add ambiguity to recognizing semantic entities.
- V_3: *Image color effects*. This technique is similar to text formatting but applied to images and videos. Different image color effects such as brightness/contrast, shadows, glows, bevel/emboss are used to highlight semantic entities within an image (cf. Figure 3 no. 7). This technique suffers from the problem that the applied effects might overlap with the existing effects in the image thereby making it hard to distinguish the semantic entities.
- V_4: *Marking* (icons appended to text or image). In this technique, which can be applied to text, images and videos, we append an icon as a marker to the part of object which includes the semantic entity (cf. Figure 3 no. 9). The most popular use

[3]http://www.musicdna.info/

Fig. 3. Screenshots of user interface techniques for visualization and exploration: 1-framing using borders, 2-framing using backgrounds, 3-video subtitle, 4-line connectors and arrow connectors, 5-bar layouts, 6-text formatting, 7-image color effects, framing and line connectors, 8-expandable callout, 9-marking with icons, 10-tooltip callout, 11-faceting

of this technique is currently within maps to indicate specific points of interest. Different types of icons can be used to distinguish different types of semantic or correlated entities.

– V_5: *Bleeping*. A bleep is a single short high-pitched signal in videos. Bleeping can be used to highlight semantic entities within a video. Different type of bleep signals can be defined to distinguish different types of semantic entities.
– V_6: *Speech* (in videos). In this technique a video is augmented by some speech indicating the semantic entities and their types within the video.

- Associating.

– V_7: *Line connectors*. Using line connectors is the simplest way to visualize the relation between semantic entities in text, images and videos (cf. Figure 3 no. 4). If the value of a property is available in the text, line connectors can also reflect the item property values. Problematic is that normal line connectors can not express the direction of a relation.
– V_8: *Arrow connectors*. Arrow connectors are extended line connectors with arrows to express the direction of a relation in a directed graph.

Besides the line and arrow connectors techniques which explicitly visualize the association between entities, implicit techniques defined as *Gestalt principles* [9] can be used for modeling association. These techniques are psychological assumptions that impose structure for human visual perception. Principles such as proximity, similarity, continuity, closure, symmetry, figure/ground and common fate can be used to affect our perception of whether and how the objects are organized into groups. Discussing these principles are out of the scope of this paper.

- Detail view.

– V_9: *Callouts*. A callout is a string of text connected by a line, arrow, or similar graphic to a part of text, image or video giving information about that part. It is used in conjunction with a cursor, usually a pointer. The user hovers the pointer over an item, without clicking it, and a callout appears (cf. Figure 3 no. 10). Callouts come in different

styles and templates such as infotips, tooltips, hint and popups. Different sort of metadata can be embedded in a callout to indicate the type of semantic entities, property values and relationships. Another variant of callouts is the *status bar* which displays metadata in a bar appended to the text, image or video container. A problem with dynamic callouts is that they do not appear on mobile devices (by hover), since there is no cursor.
- V_{10}: *Video subtitles*. Subtitles are textual versions of the dialog or commentary in videos. They are usually displayed at the bottom of the screen and are employed for written translation of a dialog in a foreign language. Video subtitles can be used to reflect detailed semantics embedded in a video scene when watching the video. A problem with subtitles is efficiently scaling the text size and relating text to semantic entities when several semantic entities exist in a scene.

3.3. Exploration

To increase the effectiveness of visualizations, users need to be capable to dynamically navigate and explore the visual representation of the semantic data. The dynamic exploration of semantic data will result in faster and easier comprehension of the targeted content. Techniques for exploration of semantics encoded in text, images and videos include:

- X_1: *Zooming*. In a zoomable UI, users can change the scale of the viewed area in order to see more detail or less. The zooming elements and techniques vary on different applications. Zooming in a semantic entity can reveal further details such as property value or entity type. Zooming out can be employed to reveal the relations between semantic entities in a text, image or video. Supporting rich dynamics by configuring different visual representations for semantic objects at different sizes is a requirement for a zoomable UI. The *iMapping* approach[5] which is implemented in the semantic desktop is an example of the zooming technique.
- X_2: *Faceting*. Faceted browsing is a technique for accessing information organized according to a faceted classification system, allowing users to explore a collection of information by applying multiple filters (cf. Figure 3 no. 11). Defining facets for each component of the predefined semantic models enable users to browse the underlying knowledge space by iteratively narrowing the scope of their quest in a predetermined order. One of the main problems with faceted browsers is the increased number of choices presented to the user at each step of the exploration [4].
- X_3: *On-demand highlighting*. Unlike the highlighting approach discussed in the visualization methods, on-demand highlighting is used to navigate the semantic entities encoded in text in a dynamic manner. One technique to realize on-demand highlighting is *Bar layout*. In the bar layout, each semantic entity within the text is indicated by a vertical bar in the left or right margin (cf. Figure 3 no. 5). The colour of the bar reflects the type of the entity. The bars are ordered by length and order in the text. Nested bars can be used to show the hierarchies of entities. Semantic entities in the text are highlighted by a mouse-over the corresponding bar. This approach is employed in Loomp [21].
- X_4: *Expanding & Drilling down*. Expandable callouts are interactive and dynamic callouts which enable users to explore the semantic data associated to a predefined semantic entity (cf. Figure 3 no. 8). Drilling down in a callout enables users to move from summary information to detailed data by focusing in on entities. This technique is employed in *OntosFeeder* [16].

3.4. Authoring

Semantic authoring aims to add more meaning to digitally published documents. If users do not only publish the content, but at the same time describe what it is they are publishing, then they have to adopt a structured approach to authoring. A semantic authoring UI is a human accessible interface with capabilities for writing and modifying semantically enriched documents. The following techniques can be used for authoring of semantics encoded in text, images and videos:

- T_1: *Form editing*. In form editing, a user employs existing form elements such as input/check/radio boxes, drop-down menu, slider, spinner, buttons, date/color picker etc. for content authoring.
- T_2: *Inline edit*. Inline editing is the process of editing items directly in the view by performing simple clicks, rather than selecting items and then navigating to an edit form and submitting changes from there.

- T_3: *Drawing*. Drawing as part of informal user interfaces [19], provides a natural human input to annotate an object by augmenting the object with human-understandable sketches. For instance, users can draw a frame around semantic entities, draw a line between related entities etc. Special shapes can be drawn to indicate different entity types or entity roles in a relation.
- T_4: *Drag and drop*. Drag and drop is a pointing device gesture in which the user selects a virtual object by grabbing it and dragging it to a different location or onto another virtual object. In general, it can be used to invoke many kinds of actions, or create various types of associations between two abstract objects.
- T_5: *Context menu*. A context menu (also called contextual, shortcut, or pop-up menu) is a menu that appears upon user interaction, such as a right button mouse click. A context menu offers a limited set of choices that are available in the current state, or context.
- T_6: *(Floating) Ribbon editing*. A ribbon is a command bar that organizes functions into a series of tabs or toolbars at the top of the editable content. Ribbon tabs/toolbars are composed of groups, which are a labeled set of closely related commands. A floating ribbon is a ribbon that appears when user rolls the mouse over a target area. A floating ribbon increases usability by bringing edit functions as close as possible to the user's point of focus. The *Aloha WYSIWYG editor*[4] is an example of floating ribbon based content authoring.
- T_7: *Voice commands*. Voice commands permit the user's hands and eyes to be busy with another task, which is particularly valuable when users are in motion or outside. Users tend to prefer speech for functions like describing objects, sets and subsets of objects [26]. By adding special signals to input voice, users can author semantic content from the scratch.
- T_8: *(Multi-touch) gestures*. A gesture is a form of non-verbal communication in which visible bodily actions communicate particular messages. Technically, different methods can be used for detecting and identifying gestures. Movement-sensor-based and camera-based approaches are two commonly used methods for the recognition of in-air gestures [22]. Multi-touch gestures are another type of gestures which are defined to interact with multi-touch devices such as modern smartphones and tablets. Users can use gestures to determine semantic entities, their types and relationship among them. The main problem with gestures is their high level of abstraction which makes it hard to assert concrete property values. Special gestures can be defined to author semantic entities in text, images and videos.

3.5. Bindings

Figure 4 surveys possible bindings between the user interface and semantic representation elements.

The bindings were derived based on the following methodology:

1. We first analyzed existing semantic representation models and extracted the corresponding elements for each semantic model.
2. We performed an extensive literature study regarding existing approaches for visual mapping as well as approaches addressing the binding between data and UI elements. If the approach was explicitly mentioning the binding composed of UI elements and semantic model elements, we added the binding to our mapping table.
3. We analyzed existing tools and applications which were implicitly addressing the binding between data and UI elements.
4. Finally, we followed a predictive approach. We investigated additional UI elements which are listed in existing HCI glossaries and carefully analyzed their potential to be connected to a semantic model element.

Although we deem the bindings to be fairly complete, new UI elements might be developed or additional data models (or variations of the ones considered) might appear, in this case the bindings can be easily extended.

Partial binding indicates the situation when a UI technique does not completely cover a semantic model element but still can be used in particular cases. For example, different text colors can be used to highlight predefined item types in text but since the colors might interfere with the current colors in the text (in case of HTML document), we assign this binding as partial binding. Another example are the line connectors used to represent the relation between items in a tree or graph-based model. In this case, on the contrary to arrow connectors, since we cannot determine the source

[4] http://aloha-editor.org

and destination of the line, we are unable to model directional relations completely, thereby, a partial binding is assigned.

The asterisks in Figure 4, indicate the cases when the metadata value is explicitly available in the text and the user just needs to provide the connection (e.g. imagine that we have `Berlin` and `Germany` mentioned in the text and we want to assign the relation `isCapitalOf`).

The following binding configurations (extracted from the literature and current tools) are available and referred to from the cells of Figure 4:

- Defining a special border or background style (C_1), text style (C_2), image color effect (C_4), beep sound (C_5), bar style (C_6), sketch (C_7), draggable or droppable shape (C_8), voice command (C_9), gesture (C_{10}) or a related icon (C_3) for each type.
- Progressive shading (C_{11}) by defining continuous shades within a specific color scheme to distinguish items in different levels of the hierarchy.
- Hierarchical bars (C_{12}) by defining special styles for nested bars.
- Grouping by similar border or background style (C_{13}), text style (C_{14}), icons (C_{15}) or image color effects (C_{16}).

For example, a user can define a set of preferred border colors to distinguish different item types (e.g. Persons, Organizations or Locations) or to group related items (e.g. all the cities in Germany).

3.6. Helper Components

In order to facilitate, enhance and customize the WYSIWYM model, we utilize a set of helper components, which implement cross-cutting aspects.

A helper component acts as an extension on top of the core functionality of the WYSIWYM model. The following components can be used to improve the quality of a WYSIWYM UI depending on the requirements defined for a specific application domain:

- H_1: *Automation* means the provision of facilities for automatic annotation of text, images and videos to reduce the need for human work and thereby facilitating the efficient annotation of large item collections. For example, users can employ existing NLP services (e.g. named entity recognition, relationship extraction) for automatic text annotation.
- H_2: *Real-time tagging* is an extension of automation, which allows to create annotations proactively while the user is authoring a text, image or video. This will significantly increase the annotation speed and users are not distracted since they do not have to interrupt their current authoring task.
- H_3: *Recommendation* means providing users with pre-filled form fields, suggestions (e.g. for URIs, namespaces, properties), default values etc. These facilities simplify the authoring process, as they reduce the number of required user interactions. Moreover, they help preventing incomplete or empty metadata. In order to leverage other user's annotations as recommendations, approaches like *Paragraph Fingerprinting* [8] can be implemented.
- H_4: *Personalization and context-awareness* describes the ability of the UI to be configured according to users' contexts, background knowledge and preferences. Instead of being static, a personalized UI dynamically tailors its visualization, exploration and authoring functionalities based on the user profile and context.
- H_5: *Collaboration and crowdsourcing* enables collaborative semantic authoring, where the authoring process can be shared among different authors at different locations. There are a vast amounts of amateur and expert users which are collaborating and contributing on the Social Web. Crowdsourcing harnesses the power of such crowds to significantly enhance and widen the results of semantic content authoring and annotation. Generic approaches for exploiting single-user Web applications for shared editing [7] can be employed in this context.
- H_6: *Accessibility* means providing people with disabilities and special needs with appropriate UIs. The underlying semantic model in a WYSIWYM UI can allow alternatives or conditional content in different modalities to be selected based on the type of the user disability and information need.
- H_7: *Multilinguality* means supporting multiple languages in a WYSIWYM UI when visualizing, exploring or authoring the content.

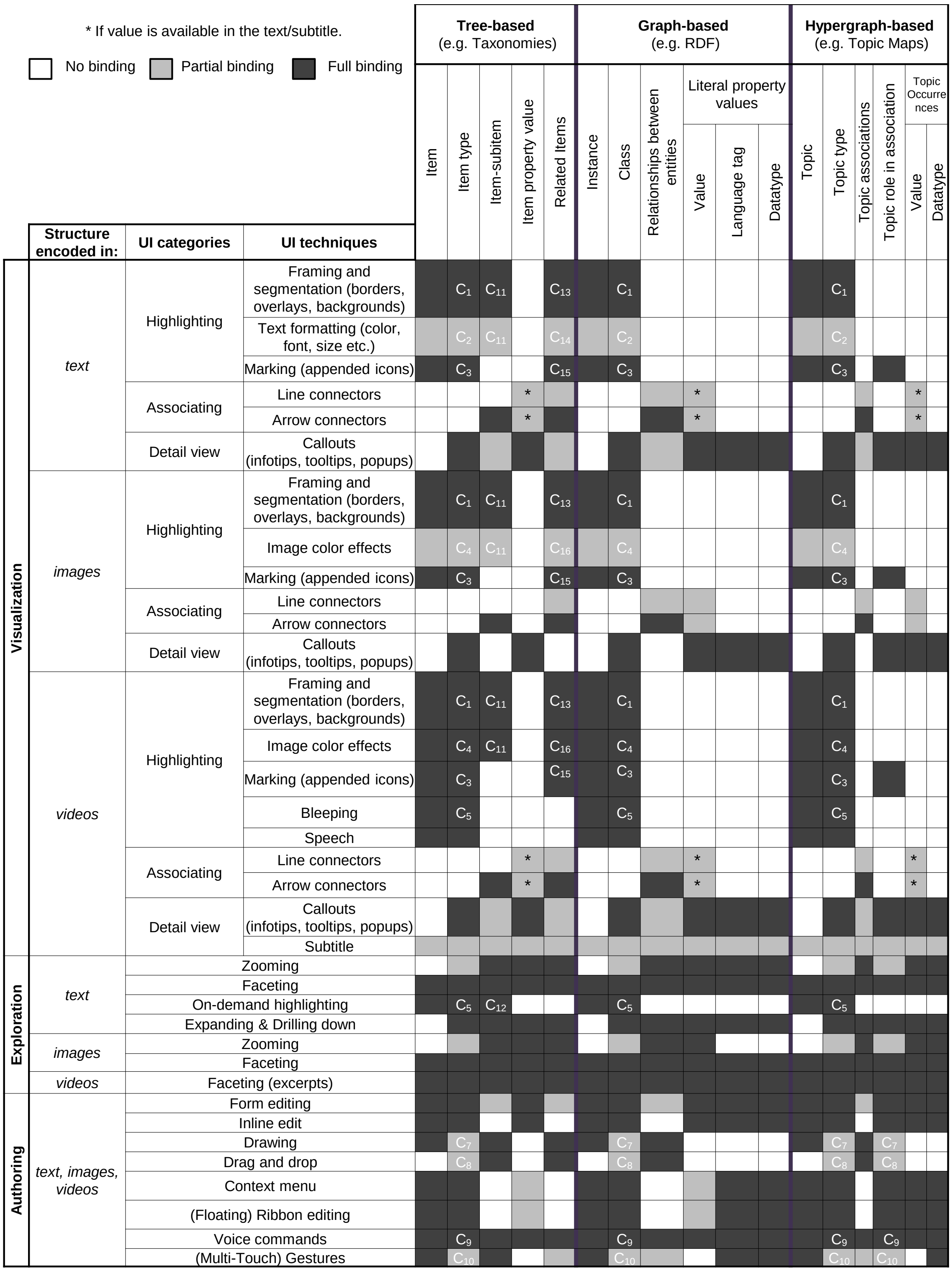

Fig. 4. Possible bindings between user interface and semantic representation model elements.

4. Implementation and Evaluation

In order to evaluate the WYSIWYM model, we implemented the three applications RDFaCE, Pharmer and conTEXT, which we present in the sequel.

RDFaCE. RDFaCE (RDFa Content Editor) [13] is a WYSIWYM interface for semantic content authoring. It is implemented on top of the *TinyMCE* rich text editor. RDFaCE extends the existing WYSIWYG user interfaces to facilitate semantic authoring within popular CMSs, such as blogs, wikis and discussion forums. The RDFaCE implementation (cf. Figure 5, left) is open-source and available for download together with an explanatory video and online demo at `http://aksw.org/Projects/RDFaCE`. RDFaCE as a WYSIWYM instantiation can be described using the following hextuple:

- D: RDFa, Microdata[5].
- V: Framing using borders (C: special border color defined for each type), Callouts using dynamic tooltips.
- E: Faceting based on the type of entities.
- T: Form editing, Context Menu, Ribbon editing.
- H: Automation, Recommendation.
- b: bindings defined in Figure 4.

RDFaCE comes with a special edition [12] customized for `Schema.org` vocabulary. In this version, different color schemes are assigned to different schemas defined in Schema.org. Users are able to create a subset of Schema.org schemas for their intended domain and customize the colors for this subset. In this version, nested forms are dynamically generated from the selected schemas for authoring and editing of the annotations.

In order to evaluate RDFaCE usability, we conducted an experiment with 16 participants of the ISSLOD 2011 summer school[6]. The user evaluation comprised the following steps: First, some basic information about semantic content authoring along with a demo showcasing different RDFaCE features was presented to the participants as a 3 minutes video. Then, participants were asked to use RDFaCE to annotate three text snippets – a wiki article, a blog post and a news article. For each text snippet, a timeslot of five

Usability Factor/Grade	Poor	Fair	Neutral	Good	Excellent
Fit for use	0%	12.50%	31.25%	43.75%	12.50%
Ease of learning	0%	12.50%	50%	31.25%	6.25%
Task efficiency	0%	0%	56.25%	37.50%	6.25%
Ease of remembering	0%	0%	37.50%	50%	12.50%
Subjective satisfaction	0%	18.75%	50%	25%	6.25%
Understandability	6.25%	18.75%	31.25%	37.50%	6.25%

Table 1

Usability evaluation results for RDFaCE.

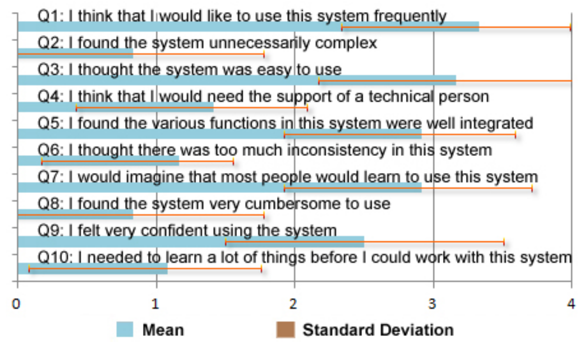

Fig. 6. Usability evaluation results for Pharmer (0: Strongly disagree, 1: Disagree, 2: Neutral, 3: Agree, 4: Strongly agree).

minutes was available to use different features of RDFaCE for annotating occurrences of persons, locations and organizations with suitable entity references. Subsequently, a survey was presented to the participants where they were asked questions about their experience while working with RDFaCE. Questions were targeting six factors of usability [17,25] namely *Fit for use*, *Ease of learning*, *Task efficiency*, *Ease of remembering*, *Subjective satisfaction* and *Understandability*. Results of the survey are shown in Table 1. They indicate on average good to excellent usability for RDFaCE. A majority of the users deem RDFaCE being fit for use and its functionality easy to remember. Also, easy of learning and subjective satisfaction was well rated by the participants. There was a slightly lower (but still above average) assessment of task efficiency and understandability, which we attribute to the short time participants had for familiarizing themselves with RDFaCE and the quite comprehensive functionality, which includes automatic annotations, recommendations and various WYSIWYM UI elements.

Pharmer. Pharmer [15] is a WYSIWYM interface for the authoring of semantically enriched electronic prescriptions. It enables physicians to embed drug-related metadata into e-prescriptions thereby reducing the medical errors occurring in the prescriptions and increasing the awareness of the patients about

[5] Microdata support is implemented in RDFaCE-Lite available at `http://rdface.aksw.org/lite`

[6] Summer school on Linked Data: `http://lod2.eu/Article/ISSLOD2011`

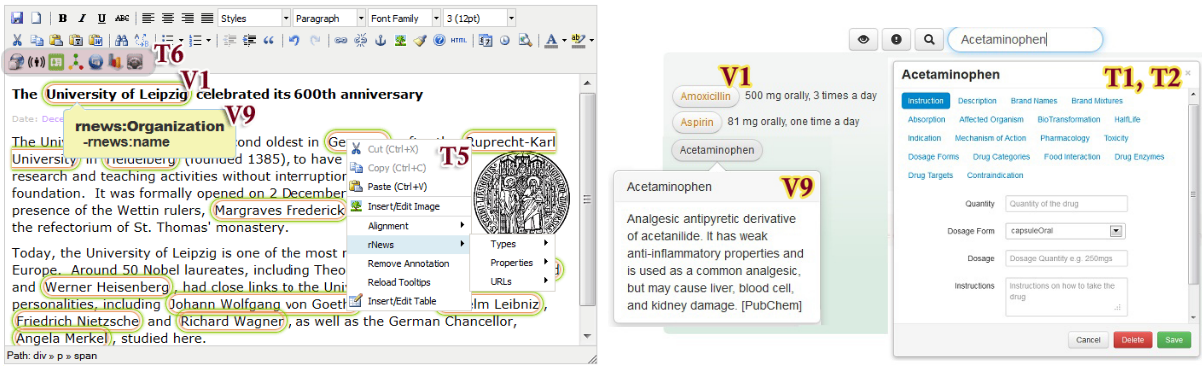

Fig. 5. Screenshots of our implemented WYSIWYM interfaces. *Left:* RDFaCE for semantic text authoring (T6 indicates the RDFaCE menu bar, V1 – the framing of named entities in the text, V9 – a callout showing additional type information, T5 – a context menu for revising annotations). *Right:* Pharmer for authoring of semantic prescriptions (V1 – highlighting of drugs through framing, V9 – additional information about a drug in a callout, T1/T2 combined form and inline editing of electronic prescriptions).

the prescribed drugs and drug consumption in general. In contrast to database-oriented e-prescriptions, semantic prescriptions are easily exchangeable among other e-health systems without need to changing their related infrastructure. The Pharmer implementation (cf. Figure 5, right) is open-source and available for download together with an explanatory video and online demo[7] at `http://code.google.com/p/pharmer/`. It is based on HTML5 *contenteditable* element. Pharmer as a WYSIWYM instantiation is defined using the following hextuple:

- D: RDFa.
- V: Framing using borders and background (C: special background color defined for each type), Callouts using dynamic popups.
- E: Faceting based on the type of entities.
- T: Form editing, Inline edit.
- H: Automation, Real-time tagging, Recommendation.
- b: bindings defined in Figure 4.

In order to evaluate the usability of Pharmer, we performed a user study with 13 subjects. Subjects were 3 physicians, 4 pharmacist, 3 pharmaceutical researchers and 3 students. We first showed them a 3-minute tutorial video of using different features of Pharmer then asked each one to create a semantic prescription with Pharmer. After finishing the task, we asked the participants to fill out a questionnaire. We used the *System Usability Scale* (SUS) [18] as a standardized, simple, ten item Likert scale-based questionnaire to grade the usability of Pharmer. SUS yields a single number in the range of 0 to 100 which represents a composite measure of the overall usability of the system. The results of our survey (cf. Figure 6) showed a mean usability score of **75** for Pharmer WYSIWYM interface which indicates a good level of usability. Participants particularly liked the integration of functionality and the ease of learning and use. The confidence in using the system was slightly lower, which we again attribute to the short learning phase and diverse functionality.

conTEXT. conTEXT [14] is a WYSIWYM interface which allows users to semantically analyze text corpora (such as blogs, RSS/Atom feeds, Facebook, G+, Twitter) and provides novel ways for browsing and visualizing the results. It helps non-programmer Web users to use sophisticated NLP techniques for text analytics and to give feedback to the NLP services for improving their quality for named entity recognition. The conTEXT implementation (cf. Figure 7) together with an explanatory video and online demo is available at `http://context.aksw.org`.

conTEXT as a WYSIWYM instantiation is defined using the following hextuple:

- D: RDFa.
- V: Framing using borders and background (C: special background color defined for each type), Callouts using dynamic popups, Text margin format for hierarchies, Line collectors for entity relations.
- E: Faceting based on the type of entities.
- T: Form editing, Inline edit.
- H: Recommendation.
- b: bindings defined in Figure 4.

[7] `http://bitili.com/pharmer`

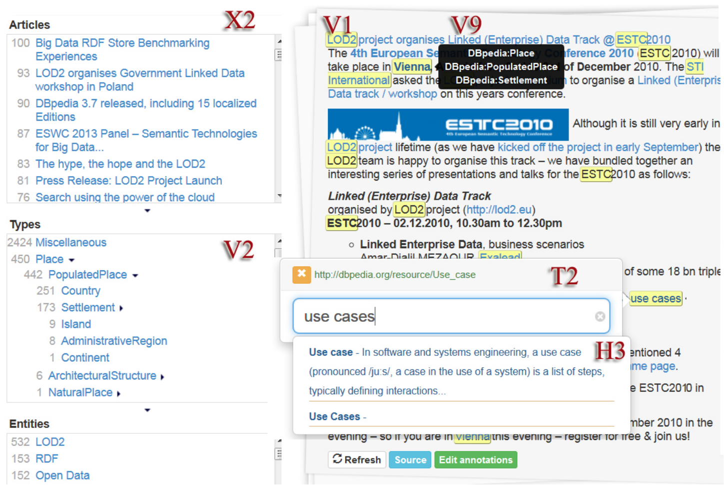

Fig. 7. Screenshots of the conTEXT WYSIWYM interfaces (T2 indicates the inline editing UI, V1 – the framing of named entities in the text, V2 – text margin formatting for visualizing hierarchy, V7 – line connectors to show the relation between entities, V9 – a callout showing additional type information, X2 – faceted browsing, H3 – recommendation for NLP feedback).

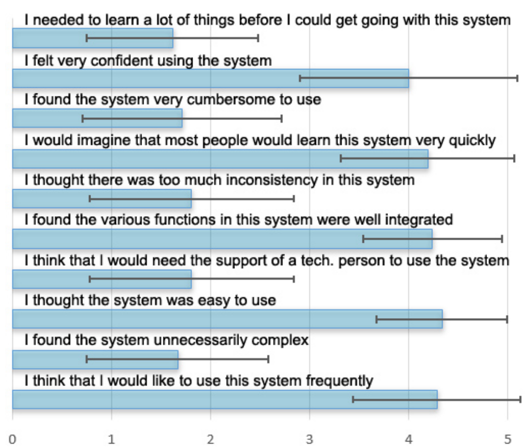

Fig. 8. Usability evaluation results for conTEXT (0: Strongly disagree, 1: Disagree, 2: Neutral, 3: Agree, 4: Strongly agree).

Faceted browsing view in conTEXT clearly shows the concept of integrated unstructured and structured view in a WYSIWYM interface. Users see unstructured text enriched with highlighted entities and detail description of the entities. In addition to that, different facets (e.g. entity type tree) allow users to filter out the text by their preferences. Users can also use inline editing within unstructured text to refine the annotations and to send feedback to NLP services.

In order to evaluate the usability of conTEXT, we performed a user study with 25 subjects (20 PhD students having different backgrounds from computer software to life sciences, 2 MSc students and 3 BSc students with good command of English) on a set of 10 questions pertaining to knowledge discovery in corpora of unstructured data. Similar to Pharmer evaluation, we used the *SUS* questionnaire to grade the usability of conTEXT. The results of our survey (cf. Figure 8) showed a mean usability score of **82** for conTEXT WYSIWYM interface which indicates a good level of usability. The responses to question 1 suggests that our system is adequate for frequent use by users. While a small fraction of the functionality is deemed unnecessary by some users, the users deem the system easy to use. Only one user suggested that he/she would need a technical person to use the system, while all other users were fine without one. The modules of the system in itself were deemed to be well integrated. Overall, the output of the system seems to be easy to understand while users even without training assume themselves capable of using the system.

5. Conclusions

Bridging the gap between unstructured and semantic content is a crucial aspect for the ultimate success of semantic technologies. With the WYSIWYM concept we presented in this article an approach for integrated visualization, exploration and authoring of unstructured *and* semantic content. The WYSIWYM model binds elements of a knowledge representation formalism (or data model) to a set of suitable UI elements for visualization, exploration and authoring. Based on such a declarative binding mechanism, we aim to increase the flexibility, reusability and development efficiency of semantics-rich user interfaces

We deem this work as a first step in a larger research agenda aiming at improving the usability of semantic user interfaces, while retaining semantic richness and expressivity. In future work we envision to adopt a model-driven approach to enable automatic implementation of WYSIWYM interfaces by user-defined preferences. This will help to reuse, re-purpose and choreograph WYSIWYM UI elements to accommodate the needs of dynamically evolving information structures and ubiquitous interfaces. We also aim to bootstrap an ecosystem of WYSIWYM instances and UI elements to support structure encoded in different modalities, such as images and videos. Creating live and context-sensitive WYSIWYM interfaces which can be generated on-the-fly based on the ranking of available UI elements is another promising research venue.

Acknowledgments

We would like to thank our colleagues from the AKSW research group for their helpful comments during the development of the WYSIWYM model. This work was partially supported by a grant from the European Union's 7th Framework Programme provided for the project LOD2 (GA no. 257943).

References

[1] A. Ankolekar, M. Krötzsch, T. Tran, and D. Vrandečić. The two cultures: Mashing up Web 2.0 and the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1):70 – 75, 2008. Semantic Web and Web 2.0.

[2] G. Burel, A. E. Cano1, and V. Lanfranchi. Ozone Browser: Augmenting the Web with Semantic Overlays. In S. Auer, C. Bizer, and G. A. Grimnes, editors, *Proc. of 5th Workshop on Scripting and Development for the Semantic Web at ESWC 2009*, volume 449 of *CEUR Workshop Proceedings, ISSN 1613-0073*, June 2009.

[3] A.-S. Dadzie and M. Rowe. Approaches to visualising Linked Data: A survey. *Semantic Web*, 2(2):89 – 124, 2011.

[4] L. Deligiannidis, K. J. Kochut, and A. P. Sheth. RDF data exploration and visualization. In P. Mitra, C. L. Giles, and L. Carr, editors, *Proceedings of the ACM first workshop on CyberInfrastructure: information management in eScience (CIMS 2007)*, pages 39 – 46. ACM, 2007.

[5] H. Haller and A. Abecker. iMapping: a zooming user interface approach for personal and semantic knowledge management. *ACM SIGWEB Newsletter*, pages 4:1 – 4:10, Sept. 2010.

[6] R. Heese, M. Luczak-Rösch, A. Paschke, R. Oldakowski, and O. Streibel. One Click Annotation. In G. T. W. G. A. Grimnes, editor, *CEUR Workshop Proceedings*, volume 699. CEUR-WS.org, February 2010.

[7] M. Heinrich, F. Lehmann, T. Springer, and M. Gaedke. Exploiting single-user web applications for shared editing: a generic transformation approach. In A. Mille, F. L. Gandon, J. Misselis, M. Rabinovich, and S. Staab, editors, *Proceedings of the 21st World Wide Web Conference (WWW 2012)*, pages 1057 – 1066. ACM, 2012.

[8] L. Hong and E. H. Chi. Annotate once, appear anywhere: collective foraging for snippets of interest using paragraph fingerprinting. In D. R. O. Jr., R. B. Arthur, K. Hinckley, M. R. Morris, S. E. Hudson, and S. Greenberg, editors, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1791 – 1794. ACM, 2009.

[9] J. Johnson. *Designing with the Mind in Mind, Second Edition: Simple Guide to Understanding User Interface Design Guidelines*. Morgan Kaufmann Publishers Inc., 2014.

[10] D. R. Karger, S. Ostler, and R. Lee. The Web Page As a WYSIWYG End-user Customizable Database-backed Information Management Application. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '09, pages 257 – 260, New York, NY, USA, 2009. ACM.

[11] A. Khalili and S. Auer. User interfaces for semantic authoring of textual content: A systematic literature review. *Web Semantics: Science, Services and Agents on the World Wide Web*, 22(0):1 – 18, 2013.

[12] A. Khalili and S. Auer. WYSIWYM Authoring of Structured Content Based on Schema.org. In X. Lin, Y. Manolopoulos, D. Srivastava, and G. Huang, editors, *Web Information Systems Engineering – WISE 2013*, volume 8181 of *Lecture Notes in Computer Science*, pages 425 – 438. Springer Berlin Heidelberg, 2013.

[13] A. Khalili, S. Auer, and D. Hladky. The RDFa Content Editor - From WYSIWYG to WYSIWYM. In X. Bai, F. Belli, E. Bertino, C. K. Chang, A. Elçi, C. C. Seceleanu, H. Xie, and M. Zulkernine, editors, *IEEE Computer Software and Applications Conference (COMPSAC 2012)*, pages 531 – 540. IEEE Computer Society, 2012.

[14] A. Khalili, S. Auer, and A.-C. Ngonga Ngomo. conTEXT – Lightweight Text Analytics Using Linked Data. In V. Presutti, C. d'Amato, F. Gandon, M. d'Aquin, S. Staab, and A. Tordai, editors, *The Semantic Web: Trends and Challenges*, volume 8465 of *Lecture Notes in Computer Science*, pages 628 – 643. Springer International Publishing, 2014.

[15] A. Khalili and B. Sedaghati. Semantic Medical Prescriptions - Towards Intelligent and Interoperable Medical Prescriptions. In *IEEE Seventh International Conference on Semantic Computing (ICSC 2013)*, pages 347 – 354. IEEE, 2013.

[16] A. Klebeck, S. Hellmann, C. Ehrlich, and S. Auer. OntosFeeder – A Versatile Semantic Context Provider for Web Content Authoring. In G. Antoniou, M. Grobelnik, E. Simperl, B. Parsia, D. Plexousakis, P. De Leenheer, and J. Pan, editors, *The Semanic Web: Research and Applications*, volume 6644 of *Lecture Notes in Computer Science*, pages 456 – 460. Springer Berlin Heidelberg, 2011.

[17] S. Lauesen. *User Interface Design: A Software Engineering Perspective*. Pearson/Addison-Wesley, Harlow, England, New York, 2005.

[18] J. Lewis and J. Sauro. The Factor Structure of the System Usability Scale. In M. Kurosu, editor, *Human Centered Design*, volume 5619 of *Lecture Notes in Computer Science*, pages 94 – 103. Springer Berlin Heidelberg, 2009.

[19] J. Lin, M. Thomsen, and J. A. Landay. A visual language for sketching large and complex interactive designs. In D. R. Wixon, editor, *Proceedings of the CHI 2002 Conference on Human Factors in Computing Systems: Changing our World, Changing ourselves*, CHI '02, pages 307 – 314. ACM, 2012.

[20] V. Lopez, V. Uren, M. Sabou, and E. Motta. Is Question Answering Fit for the Semantic Web?: A Survey. *Semantic Web Journal*, 2(2):125 – 155, Apr. 2011.

[21] R. H. M. Luczak-Roesch. Linked Data Authoring for Non-Experts. In C. Bizer, T. Heath, T. Berners-Lee, and K. Idehen, editors, *Proceedings of the WWW2009 Workshop on Linked Data on the Web (LDOW 2009)*, volume 538 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[22] A. Löcken, T. Hesselmann, M. Pielot, N. Henze, and S. Boll. User-centred process for the definition of free-hand gestures applied to controlling music playback. *Multimedia Systems*, 18(1):15 – 31, 2012.

[23] B. A. Myers. A brief history of human-computer interaction technology. *interactions*, 5(2):44 – 54, 1998.

[24] W. Müller, I. Rojas, A. Eberhart, P. Haase, and M. Schmidt. A-R-E: The Author-Review-Execute Environment. *Procedia Computer Science*, 4(0):627 – 636, 2011. Proceedings of the International Conference on Computational Science, {ICCS} 2011.

[25] J. Nielsen. Introduction to Usability. `http://www.useit.com/alertbox/20030825.html`, 2012.

[26] S. Oviatt, P. Cohen, L. Wu, J. Vergo, L. Duncan, B. Suhm, J. Bers, T. Holzman, T. Winograd, J. Landay, J. Larson, and D. Ferro. Designing the user interface for multimodal speech and pen-based gesture applications: state-of-the-art systems and future research directions. *Human-Computer Interaction*, 15(4):263 – 322, Dec. 2000.

[27] E. Pietriga, C. Bizer, D. Karger, and R. Lee. Fresnel: A Browser-Independent Presentation Vocabulary for RDF. In I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, editors, *The Semantic Web - ISWC 2006*, volume 4273 of *Lecture Notes in Computer Science*, pages 158 – 171. Springer Berlin Heidelberg, 2006.

[28] R. Power, D. Scott, and R. Evans. What You See Is What You Meant: direct knowledge editing with natural language feedback. In *European Conference on Artificial Intelligence (ECAI)*, pages 677 – 681, 1998.

[29] M. Sah, W. Hall, N. M. Gibbins, and D. C. D. Roure. SEMPort - A Personalized Semantic Portal. In *18th ACM Conf. on Hypertext and Hypermedia*, pages 31 – 32, 2007.

[30] C. Sauer. What you see is Wiki – Questioning WYSIWYG in the Internet Age. In *Proceedings of Wikimania 2006*, 2006.

[31] B. Shneiderman. Creating creativity: user interfaces for supporting innovation. *ACM Transactions on Computer-Human Interaction (TOCHI) - Special issue on human-computer interaction in the new millennium*, 7(1):114 – 138, Mar. 2000.

[32] J. Spiesser and L. Kitchen. Optimization of HTML Automatically Generated by WYSIWYG Programs. In S. I. Feldman, M. Uretsky, M. Najork, and C. E. Wills, editors, *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 355 – 364, New York, NY, USA, 2004. ACM.

[33] D. Tunkelang. *Faceted Search (Synthesis Lectures on Information Concepts, Retrieval, and Services)*. Morgan and Claypool Publishers, June 2009.