WYSIWYM – Direct Manipulation of Semantically Structured Content in Conventional Modalities

Ali Khalili , Sören Auerr
IFI/AKSW, University of Leipzig, Germany
http://aksw.org, lastname@informatik.uni-leipzig.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 and 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* semantic information in a holistic 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 and Pharmer we present and evaluate two complementary showcases implementing the WYSIWYM concept for different application domains.**

I. Introduction

The Semantic Web and Linked Data 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. Enriching documents with semantic representations facilitates a number of important aspects of information management such as *search and retrieval*, *information presentation*, *information integration*, *personalization*, *reusability* and *interoperability*.

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 (e.g. *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* semantic information in a holistic manner.

In this paper, we present the WYSIWYM (What You See Is What You Mean) concept, which addresses the direct manipulation of semantically structured content in conventional modalities. 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 extract a list of semantic representation elements, such as entities, various relationships and attributes. We survey 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 interface for the purpose of automation, annotation, recommendation, personalization etc.

With *RDFaCE* and *Pharmer* we present and evaluate two complementary showcases implementing the WYSIWYM concept for different domains. RDFaCE is a domain-agnostic text editor with embedded semantic in the form of RDFa or Microdata. Pharmer is a WYSIWYM UI for the authoring of semantic prescriptions in the medical domain. Our evaluation of both tools with end-users (RDFaCE) and domain experts (Pharmer) shows, that WYSIWYM UIs provide good usability, while retaining benefits of a truly semantic representation.

Our contributions are in particular:

1) A formalization of the WYSIWYM concept based on definitions for the model, binding and concrete interfaces.
2) A comprehensive 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) Two complementary use cases, which evaluate different, concrete WYSIWYM interfaces in a generic as well as domain specific context.

The remainder of this article is structured as follows: After an overview of the related work in Section II, we describe the fundamental WYSIWYM concept proposed in the paper and present the different components of the WYSIWYM model in Section III. We then introduce two implemented WYSIWYM interfaces together with their evaluation results in Section IV. In Section V we conclude with an outlook on future work.

II. Related Work

1) 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. WYSIWYG text authoring is meanwhile ubiquitous

on the Web and part of most content creation and management workflows (e.g. content management systems , weblogs, wikis). 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 [1], [2].

2) WYSIWYM: The first use of the WYSIWYM term occurred in 1995 aiming to capture the separation of presentation and content when writing a document. Instead of focusing on the format or presentation of the document, a WYSIWYM editor preserves the intended meaning of each element (e.g. page headers, sections, paragraphs, etc. are labeled as such in the editing program, and displayed appropriately in the browser). The term WYSIWYM was also used for *Symbolic Authoring* [3], where 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. The WYSIWYM term as defined and used in this work targets the novel aspect of integrated UIs for the management of unstructured and semantic content. The rationale of our WYSIWYM concept is to enrich the existing WYSIWYG presentational view 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 concept integrates these aspects to facilitate the process of *Semantic Content Authoring*.

3) Binding data to UI elements: There are already few approaches and tools which address the binding between data and UI elements for visualizing and exploring semantically structured data. Dadzie and Rowe [4] present the most exhaustive and comprehensive survey to date of these approaches. For example, *Fresnel* [5] is a display vocabulary for RDF based on 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 semantically 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 unstructured content (cf. the literature review [6]). As an example, *Dido* [7] describes 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*[1] framework. Another example is *Loomp*, which implements the *One Click Annotation* [8] strategy. Loomp is a WYSIWYG web editor for enriching content with RDFa, which employs a partial mapping between UI elements and data to hide the complexity of creating semantic data.

[1] http://simile-widgets.org/exhibit/

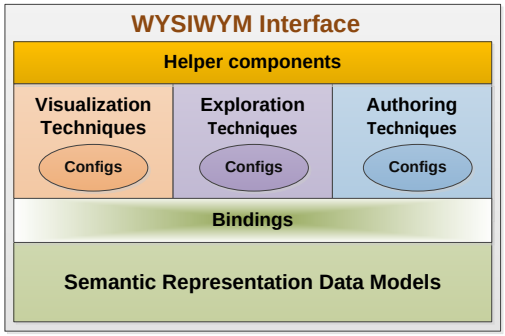

Fig. 1. Schematic view of the WYSIWYM model.

III. 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. The formalization serves the purpose of providing a terminology for software engineers, user interface and domain experts 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 semantic content.

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 the 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 is 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 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 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 t;
- H is a set of helper components.

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 UI technique ui ($ui \in V \cup X \cup T$) and c is a configuration $c \in C_{ui}$.

Figure 2 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. 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 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 between a particular occurrence of a data model element and a visualization, exploration and/or authoring technique[2].

In the remainder of this section we discuss the different elements of the WYSIWYM concept in more detail.

A. 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*. 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 import to bind the most suitable UI elements to respective representation elements. In the sequel we briefly discuss the three different types of representation models.

1) 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.

2) 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 compared to the 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) vocabulary for describing people, their interests and relationships in a social network, in the medical domain (e.g. DrugBank, Diseasome, ChEMBL) to describe relations between diseases, drugs and genes, or generically in the *SIOC* (Semantically-Interlinked Online Communities) vocabulary. Elements of 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`.

3) 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 which consists of topics, associations and occurrences. The semantic expressivity of Topic Maps is, in many ways, equivalent to that of RDF, with the major difference 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 triples. 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*, an

[2]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.

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, Leipzig`.
  - E_{5_2}: datatype – e.g. `text`.

B. 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:

a) Highlighting:

- V_1: *Framing and Segmentation* (borders, overlays and backgrounds) can be applied to text, images and videos by enclosing 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.
- V_2: *Text formatting* (color, font, size, etc.) through different text styles helps to distinguish semantic entities within a text. Applied in an HTML document, the semantic styles might overlap with existing styles in the document and thereby add ambiguity to recognizing semantic entities.
- V_3: *Image color effects* (brightness/contrast, shadows, glows, bevel/emboss) are similar to text formatting but applied to images and videos to highlight semantic entities within an image. Applied effects might overlap with existing effects in the image thereby making it hard to distinguish semantic entities.
- V_4: *Marking* (icons appended to text or image) can be applied to text, images and videos where an icon is appended as a marker to the part of object which references the semantic entity. The most popular use of this technique is currently within maps to indicate points of interest. Different types of icons can be used to distinguish different types of semantic or correlated entities.
- V_5: *Bleeping* refers to a single short high-pitched signal in videos, which can be used to highlight semantic entities. Different types of bleep signals can be defined to distinguish different types of semantic entities.
- V_6: *Speech* (in videos) can indicate the semantic entities and their types within the video.

b) Associating:

- V_7: *Line connectors* are the simplest way to visualize the relation between semantic entities in text, images and videos. If the value of a property is available in the text, line connectors can also reflect the item property values. Normal line connectors can not express the direction of a relation.
- V_8: *Arrow connectors* are extended line connectors with arrows to express the direction of a relation in a directed graph.

c) Detail view:

- V_9: *Callouts* are strings 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. Callouts come in different styles and templates such as infotips, tooltips, hints and popups. Different types of semantic information 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 semantic information in a bar appended to the text, image or video container. Dynamic callouts do not appear on mobile devices (by hover), since there is no cursor.
- V_{10}: *Video 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 translations or transcriptions. Video subtitles can be used to reflect detailed semantics embedded in a video scene when watching the video. Problematic is efficiently scaling the text size and relating text to semantic entities when several semantic entities exist in a scene.

C. Exploration

To increase the effectiveness of visualizations, users need to be empowered to dynamically 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 include:

- X_1: *Zooming* allows users to change the scale of the viewed area in order to see more or less detail. Zooming into a semantic entity can reveal further details such as property values or the 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. *iMapping* [9], implemented in the semantic desktop, is an example of the zooming technique.

- X_2: *Faceting* 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. Defining facets for each component of the predefined semantic models enables users to browse the underlying knowledge space by iteratively narrowing the scope of their quest. Problematic is the increased number of choices presented to the user at each step of the exploration [10].
- X_3: *Bar layouts* indicate each semantic entity within the text by a vertical bar in the left or right margin (cf. Loomp [11]). The colour of the bar reflects the type of the entity. Bars are ordered by length and order in the text. Nested bars show hierarchies of entities. Semantic entities in the text are highlighted by a mouse-over the corresponding bar.
- X_4: *Expandable callouts* are interactive and dynamic callouts to explore the semantic data associated with a predefined semantic entity (cf. *OntosFeeder* [**?**]).

D. 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 semantic documents. The following techniques can be used for authoring of semantics encoded in text, images and videos:

- T_1: *Form editing* comprises existing form elements such as input/check/radio boxes, drop-down menu, slider, spinner, buttons, date/color picker etc.
- T_2: *Inline editing* allows editing items directly in the view by performing simple clicks, rather than selecting items and then navigating to an edit form.
- T_3: *Drawing* as part of informal user interfaces [12], 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 indicate different entity types or entity roles in a relation.
- T_4: *Drag and drop* can be used to create various types of associations between two abstract objects.
- T_5: *Context menu*. 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 comprising groups of closely related commands at the top of the editable content. A floating ribbon appears when user rolls the mouse over a target area and thus increases usability by bringing edit functions as close as possible to the user's point of focus.
- T_7: *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 [13].
- T_8: *(Multi-touch) gestures* enable a form of non-verbal communication in which visible bodily actions communicate particular messages. Users can use gestures to determine semantic entities, their types and relationship among them. Problematic is their high level of abstraction which makes it hard to assert concrete property values.

E. Bindings

Figure 2 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 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.
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 might appear, in this case the bindings can be easily extended. The following binding configurations are available and refereed to from the cells of Figure 2:

- 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}).

F. Helper Components

In order to facilitate, enhance and customize the WYSIWYM model, we define 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 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 human work and thereby facilitating the efficient

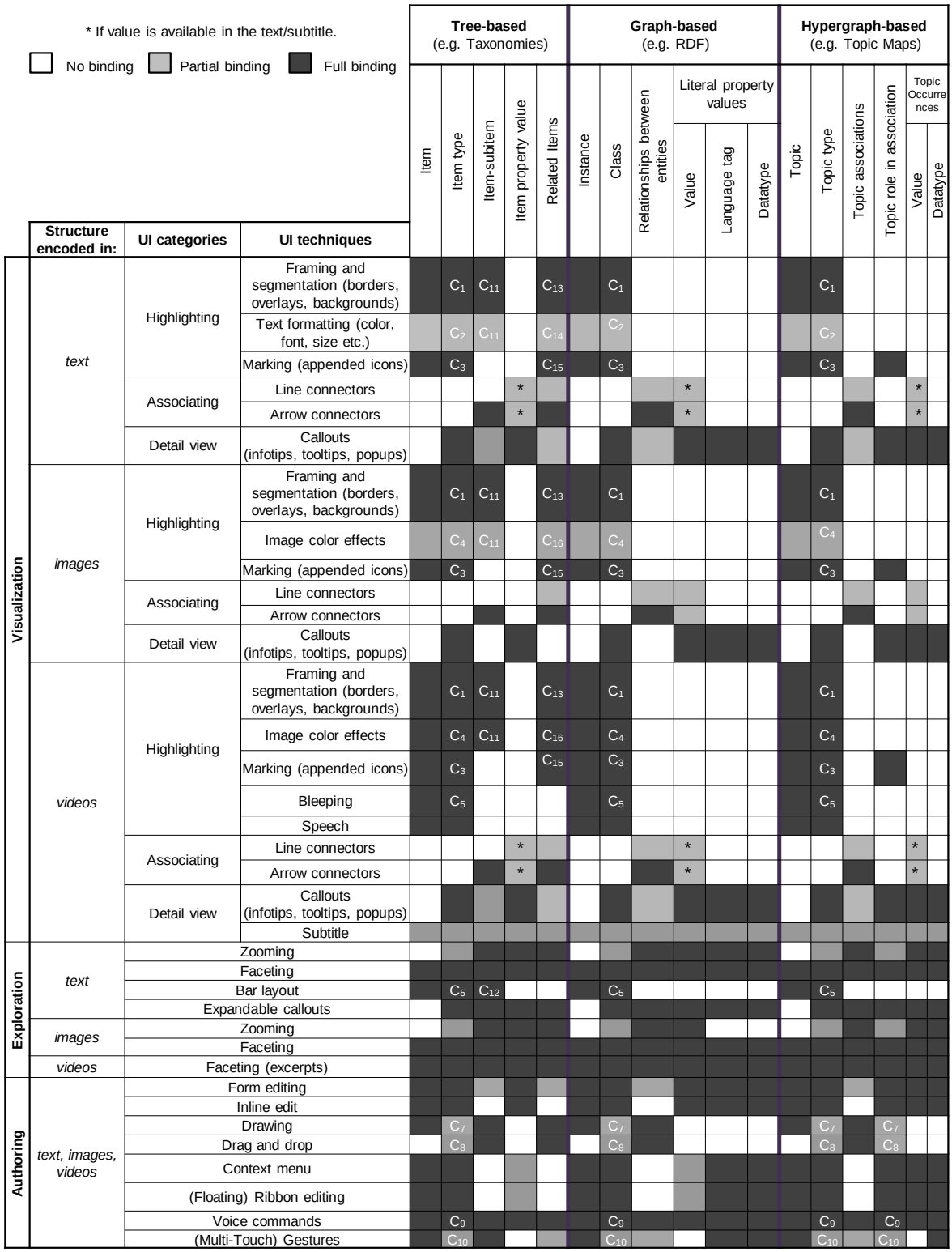

Fig. 2. Possible bindings between user interface and semantic representation model elements.

annotation of large item collections. For example, users can employ existing NLP services 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 significantly increases the annotation speed and users 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, and help preventing incomplete or empty metadata.
- 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 functionality 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. Crowdsourcing harnesses the power of online communities to significantly enhance and widen the results of semantic content authoring and annotation. Generic approaches for exploiting single-user Web applications for shared editing [14] can be employed in this context.
- H_6: *Accessibility* means providing people with disabilities and special needs with appropriate UIs. The underlying semantic model allows 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 when visualizing, exploring or authoring the content.

IV. Implementation and Evaluation

In order to evaluate the WYSIWYM model, we implemented the two applications RDFaCE and Pharmer.

RDFaCE (RDFa Content Editor) is a WYSIWYM interface for semantic text authoring implemented on top of the *TinyMCE* rich text editor (cf. Figure 3, left). RDFaCE extends the existing WYSIWYG UI to facilitate semantic authoring within popular CMSs, such as blogs, wikis and discussion forums. RDFaCE 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 as follows:

- D: RDFa, Microdata.
- 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 2.

In order to evaluate RDFaCE usability, we conducted an experiment with 16 participants of the ISSLOD 2011 summer school.The user evaluation comprised the following steps: First, basic information about semantic content authoring along with a demo showcasing different RDFaCE features was presented to the participants as a 3-minute 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 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 [15] namely *Fit for use*, *Ease of learning*, *Task efficiency*, *Ease of remembering*, *Subjective satisfaction* and *Understandability*. Results of the survey are shown in Table I. They indicate on average good to excellent usability. 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.

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 I
USABILITY EVALUATION RESULTS FOR RDFACE.

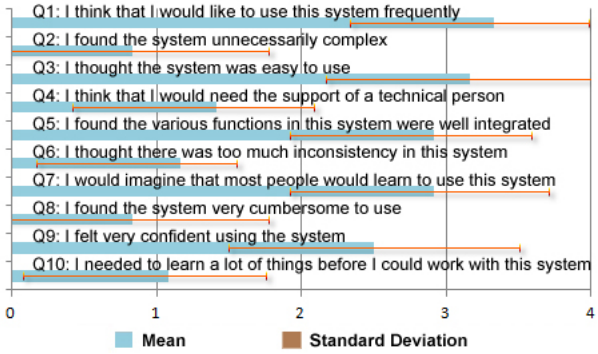

Fig. 4. Usability evaluation results for Pharmer (0: Strongly disagree, 1: Disagree, 2: Neutral, 3: Agree, 4: Strongly agree).

Pharmer 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 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 3, right) is open-source and available for download together with an explanatory video and online demo at http://code.google.com/p/pharmer/. Pharmer as a WYSIWYM instantiation is defined using the following hextuple:

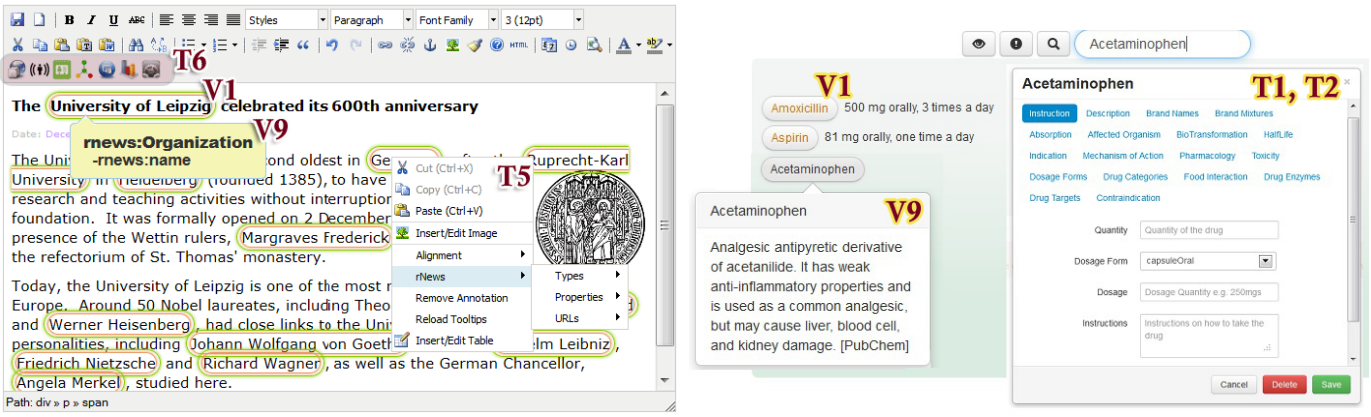

Fig. 3. Screenshots of our two implemented WYSIWYM interfaces. *Left:* RDFaCE for semantic text authoring (T6 indicates the RDFaCE menu bar, V1 – the framing of named entities in the text, V9 – a callout showing additional type information, T5 – a context menu for revising annotations). *Right:* Pharmer 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).

- D: RDFa.
- V: Framing using 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 2.

We performed a user study with 13 subjects: 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. Afterwards, we asked the participants to fill out the standardized, ten item Likert scale-based *System Usability Scale* (SUS) [16] questionnaire to grade the usability of Pharmer. SUS yields a single number in the range of 0 to 100 as a composite measure of the overall system usability. The results of our survey (cf. Figure 4) showed a mean usability score of **75** indicating 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.

V. Conclusion and Future Work

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 direct manipulation of semantically structured content in conventional modalities. 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 avenue.

References

[1] J. Spiesser and L. Kitchen, "Optimization of html automatically generated by wysiwyg programs," in *WWW 2004*, pp. 355–364.

[2] C. Sauer, "What you see is wiki – questioning WYSIWYG in the Internet age," in *Proceedings of Wikimania 2006*, 2006.

[3] R. Power, R. Power, D. Scott, D. Scott, R. Evans, and R. Evans, "What you see is what you meant: direct knowledge editing with natural language feedback," 1998.

[4] A.-S. Dadzie and M. Rowe, "Approaches to visualising linked data: A survey," *Semantic Web*, vol. 2, no. 2, pp. 89–124, 2011.

[5] E. Pietriga, C. Bizer, D. R. Karger, and R. Lee, "Fresnel: A browser-independent presentation vocabulary for rdf." in *ISWC*, ser. LNCS. Springer, 2006, pp. 158–171.

[6] A. Khalili and S. Auer, "User interfaces for semantic authoring of textual content: A systematic literature review," 2012.

[7] D. R. Karger, S. Ostler, and R. Lee, "The web page as a wysiwyg end-user customizable database-backed information management application," in *UIST 2009*. ACM, 2009, pp. 257–260.

[8] "One Click Annotation," ser. CEUR Workshop Proceedings, G. T. W. G. A. Grimnes, Ed., vol. 699, February 2010.

[9] H. Haller and A. Abecker, "imapping: a zooming user interface approach for personal and semantic knowledge management," *SIGWEB Newsl.*, pp. 4:1–4:10, Sep. 2010.

[10] L. Deligiannidis, K. J. Kochut, and A. P. Sheth, "RDF data exploration and visualization," in *CIMS 2007*. ACM, 2007, pp. 39–46.

[11] R. H. M. Luczak-Roesch, "Linked data authoring for non-experts," in *WWW WS on Linked Data on the Web (LDOW2009)*, 2009.

[12] A. Klebeck, S. Hellmann, C. Ehrlich, and S. Auer, "Ontosfeeder - a versatile semantic context provider for web content authoring," in *ISWC 2011*, ser. LNCS, vol. 6644, pp. 456–460.

[13] J. Lin, M. Thomsen, and J. A. Landay, "A visual language for sketching large and complex interactive designs," ser. CHI '02. ACM, pp. 307–314.

[14] 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," *Hum.-Comput. Interact.*, vol. 15, no. 4, pp. 263–322, Dec. 2000.

[15] M. Heinrich, F. Lehmann, T. Springer, and M. Gaedke, "Exploiting single-user web applications for shared editing: a generic transformation approach," in *WWW 2012*. ACM, 2012, pp. 1057–1066.

[16] S. Lauesen, *User Interface Design: A Software Engineering Perspective*. Addison Wesley, Feb. 2005.

[17] J. Lewis and J. Sauro, "The Factor Structure of the System Usability Scale," in *Human Centered Design*, ser. LNCS, 2009, vol. 5619, pp. 94–103.