Semantic Tools for Web 2.0 Educational Platforms

Vlad Catalin Merticariu

Computer Science
Jacobs University Bremen
Campus Ring 1
28759 Bremen
Germany

Type: *Guided Research Thesis*
Date: *May 11, 2012*
Supervisor: *Prof. Dr. M. Kohlhase*

Abstract

Today's web is moving towards a Semantic state and, with it, the existing educational platforms will shortly follow up. Having a set of tools which allow the gradual migration from Web 2.0 to Web 3.0 constitutes an important asset when trying to avoid losing the features and advantages that the current implementations offer.

Localized discussions, the first of the proposed tools, has the purpose of improving the level of interaction of the user with the content, allowing him to make context related comments following a natural work flow.

Books 3.0, the second semantic tool, represents a framework for publishing modular content objects in an independent, uncoupled manner, with high emphasis on context adaptation.

Contents

1 Introduction

The web is moving towards a new form, one that facilitates participatory information sharing, interoperability and collaboration between users. Following this idea, the project proposes two tools which aim at building a system in which the user represents one of the main components, increasing his incentive to participate in the development of the community and providing new means of access to the content.

Transitioning from Web 2.0 to Web 3.0 [1] is something to be done incrementally for the existing platforms, in such a way that the current functionality is not replaced, but rather used as a base on top of which new pieces are built. This allows the user to take advantage of the new features, without forcing him to use them, but rather encouraging him to discover them by himself.

The first focus point of the project is the improvement of the level of interaction that the user has regarding the content, by introducing the concept of localized discussions: comments can be made on specific parts of content so that the context of the discussions is always visible. Especially useful on educational platforms, where the subjects are often open for discussion and the users encouraged to communicate, localized discussions create the perfect environment for developing new ideas and improving existing ones.

A simple use case of this tool is represented by the course notes of a general university course. Students have the possibility to discuss, together with the instructors, specific parts of the presented content, directly referencing it in their comments. Besides saving important time for all the categories of users, this also helps the students to gain a better understanding of the subject and the instructors to identify the parts which raise the greatest difficulties.

The second focus point of the project is the creation of a framework which allows independent content modules to be published in a context dependent manner [1]. Starting from a repository of semantically annotated documents, the tool allows documents to be published in the form of hierarchically organized books. Unrelated before publishing, the documents are adapted to the context in which they are displayed at the presentation time, making their reuse cost and time effective.

The most prominent use case for this tool comes again in the academic environment, or for educational platforms in general, where parts of the same document are often presented in different contexts.

[1] We adopt the nomenclature where Web 3.0 stands for extension of the Social Web with Semantic Web/Linked Open Data technologies.

2 Related Work

Localized discussions are a relatively new concept in WWW. A few implementations of the idea exist, the most reliable one being on TurnItIn website [2]. It focuses on the user's experience, addressing issues of navigation, personalization and presentation.

Comments are added using a navigation menu implemented by a Flash application [3] and their presentation is done by marking the commented text and displaying, both on place and outside the page, in an overview menu, the discussions. Their model works well, receiving a great amount of positive feedback on communities as Stackoverflow[4], reason for which I decided to use the same main ideas of implementation.

Electronic books are a much older concept, considered to have started in the early 1960s, with the NLS project headed by Doug Engelbart at Stanford Research Institute (SRI), and the Hypertext Editing System and FRESS projects headed by Andries van Dam at Brown University [2] [3] [4].

Current implementation vary widely, the ones that drew my attention being the system implemented for the Jacobs University's library [5] and the existing implementation of Drupal's Books Module [5]. Common features include hierarchical navigation and presentation, present in both implementations and developed under constant user feedback. From this point of view, following their model seemed to be the best choice in order to come up with a tool which satisfies the needs of most of the users.

None of them, however, provides means for context adaptation, limiting this way the conceptual models for the second part of Books 3.0.

3 Framework - The Planetary System and Drupal

Building a new framework would require a great amount of work and time investment, that is why I focused my attention on the already existing systems that fit my needs. I used The Planetary System, a semantic environment for document collections based on Drupal, for several reasons which will be discussed in the following lines [6].

Drupal code is freely available under the terms of GNU GPL [6], and it is required

[2] http://turnitin.com

[3] Adobe Flash (formerly Macromedia Flash) is a multimedia platform used to add animation, video, and interactivity to web pages.

[4] http://stackoverflow.com

[5] http://www.jacobs-university.de/library

[6] http://www.gnu.org/licenses/gpl.html

to comply with the Open Source Initiatives (OSI) criteria for open source software. These criteria mean open and accessible source code, free redistribution and a non-restrictive license. The Principles of Drupal require the code to have low demands on resources, to be standardsbased (XHTML and CSS), to be documented and to be modular, extensible, scalable, and maintainable.

There are several categories of features that gave Drupal superiority compared to other systems:

- The user management system - Through the finetuned user management system, the site administrator can customize access levels for different parts of the site and can define user groups, roles, and permissions assigned to roles. Drupal also has a version control feature, which tracks the details of content update and allows the incorporation of feedback from the community on the website.
- The library modules - The Drupal core comes with tools for user management, metadata functionalities using controlled vocabularies and XML publishing for content sharing purposes, tools for content creation, management, publishing, and presentation.
- The Books module - Provides the implementation needed for hierarchical navigation and storage [7].

4 Components

4.1 Localized Discussion

4.1.1 Description

Localized comments and references have the purpose of increasing the platform's usability by making pieces of relevant information available right away, in the context they are needed, and also by giving the user the opportunity to surprise important aspects of the article in a content dependent manner.

The implementation of this feature is based on Asynchronous JavaScript and XML (AJAX) and consists of two parts: viewing and adding comments.

Inserting a localized comment is done by highlighting the area to which it corresponds (Figure 1) and accessing a simple form which opens in a pop-up window, so the navigation through content is not interrupted (Figure 2) .

Figure 1: Selecting the words ”Pellentesque habitant” in order to add a localized comment.

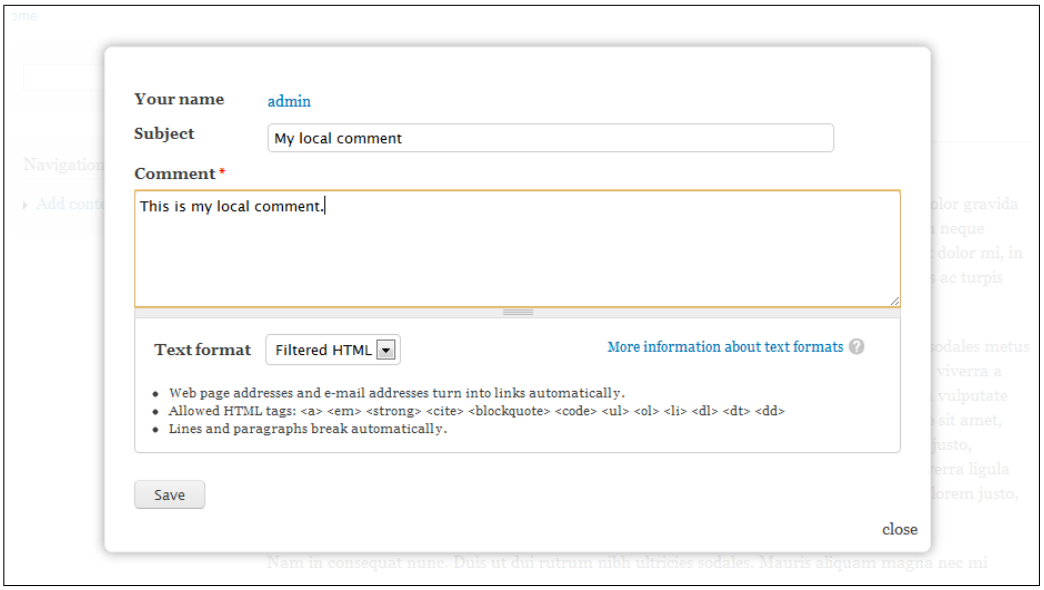

Figure 2: Pop-up modal box opened in order to insert the localized comment.

For the presentation of the comments I used the already running model of TurnItIn [7], which consists in highlighting the area to which a comment corresponds and displaying it on hover or on click, depending on the user's preferences (Figure 3) . A summary of all localized comments on the page is also available in the comments section, making navigation through the article easier when it comes to following suggestions or questions from the community.

A further component of this module is the integration with Drupal's default user

[7] http://turnitin.com

Figure 3: Localized comment displayed on hover.

profile, leaving him in control of basic settings as the one mentioned above, in the attempt to increase the user-friendliness of the platform [8].

At the infrastructure level, localized discussions are integrated into Drupal's default commenting system, fact that allows us to take advantage of the core features for user roles and permissions [9].

4.1.2 Technical details

Implementing localized discussions raises several problems, from finding a meaningful way of binding a comment to a piece of text, to making its insertion and display in ways that don't appear unnatural to the user. The implementation and generated work flow follow several steps, which we are going to explore in the next lines.

- **Preparing the content**
  When displaying an article or a page, a way of identifying each word's position is needed. Creating a localized discussion involves a relation between the comment itself and the context to which it is bind, so the page should provide an easy way to access and store it. There are three obvious ideas which could solve this problem, however none of them works straight forward.

  Using the actual words as context to be stored is the first thing that may come to one's mind because retrieving their value can be easily done with Javascript. There are several reasons for which this method doesn't work right away. The first, most important one, is that the same word can appear in several places in the same article. This implies that the relation between comments and context is not one to one, but one to many, which would make the linking between the discussion and the context impossible. Moreover, editing a word on which a comment exists or adding a similar word would

cause the bind to malfunction. Thus, this method wouldn't be suitable, even though the implementation effort would be minimal.

Another method for binding a comment to an article is using the page coordinates of the point to which the discussion has been initialized. Even tough it appears to do the job, this method is very artificial, unaware of the context and very unstable between browsers. Moreover, any content modification would cause all the comments to be misplaced, a problem that would be very expensive to solve and which would occur frequently. For this reasons I decided to look further, searching for less obvious solutions, which, at the cost of a more difficult implementation, would confer more stability.

A third solution to this issue is finding a way to establish a word counter and bind the discussions to the number or numbers of the selected words. This has to provide a stable way of binding the comments to the article's text and has to be invariant to text editing. For this reason, the counting is done right after the document has been displayed, on the client, every time it is accessed. Its implementation is done using Javasript, operating on the HTML of the page in the following way:

- Extract the HTML of the page.
- Define and initialize a global counter.
- Define and initialize a resulting HTML.
- Use a regular expression to identify the HTML tags, so only the text is affected.
- Iterate over the text and, if an HTML tag is found, join it to the result.
- If a word is found, wrap it in a *span* HTML element, using the global counter as id, increase the counter value and join the wrapped word to the result.
- Use the result to replace the HTML of the page.

Applying this method causes the following HTML code to be transformed as shown:

```
<p>
  Lorem ipsum dolor sit
  <a href = "#">
    amet.
  </a>
</p>
```

becomes:

```
<p>
  <span class = "lcomment" id = "0">Lorem </span>
  <span class = "lcomment" id = "1">ipsum </span>
  <span class = "lcomment" id = "2">dolor </span>
  <span class = "lcomment" id = "3">sit </span>
  <a href = "#">
    <span class = "lcomment" id = "4">amet.</span>
  </a>
</p>
```

Now the page is prepared for inserting comments because selecting a word or a group of words returns their counters as well, as ids of the wrapping HTML elements. This is enough for creating a one to one relation between comments and context, but requires additional actions when storing the comment, in order to ensure invariance to editing, actions which we will discuss in the next sections.

- **Using a modal window to display the comment form and injecting HTML**

  In order not to interrupt the navigation throughout the article, the default comment form provided by Drupal is used in a modal view. A simple service is implemented for retrieving the form corresponding to the specified article, which is then displayed in an iframe based on the Colorbox jQuery Plugin [8], using an AJAX request to the service page.

  Using the default comment form allows us to take advantage of Drupal's role-permission system, localized discussions being treated as normal comments. Moreover, the binding between the comment and the corresponding article is done automatically, leaving only the task of binding the comment to the context.

  From the page preparation part, word counters are returned and stored together with the unique identifier of the comment, by overriding the actions performed at the submission of the comment form. The only remaining problem is invariance to editing. If words are added or removed to or from the article, the counter changes. In order to avoid changing the binding of the comments together with the counter, at the time the comment form is submitted, HTML markers are injected around the targeted words and a new version of the document is saved. This ensures that the comments are going to remain in the same context as they were initially inserted, no matter how many changes are made in the article.

  Deleting a piece of content which has a comment attached doesn't trigger any action initially. This problem is handled at the display time.

[8] http://www.jacklmoore.com/colorbox

- **Displaying localized discussions**

  The display of the localized discussions is done in two steps:

  - As normal comments, creating an overview of all the discussions on the page, handled by Drupal by default.
  - In place, on the page, at hover or click on a commented word, handled by a Javascript binding.

  The binding first checks whether commented words exist, searching for HTML markups inserted in the previous step. Several actions follow:

  - If the markup exists and is empty (i.e. the word has been deleted), the markup is removed and the binding between the comment and the word is unset, keeping the comment as unreferenced.
  - For the markups which are not empty, special CSS styles are applied, comment bodies are loaded from the same page (they exist as normal comments so they are rendered together with the page) and are registered for display at the user's chosen events (hover or click)[10] [11].
  - When the registered event is triggered, the comments are displayed in place, in a tooltip, using the Selected Text Sharer jQuery Plugin [9].

- **Granularity**

  The smallest granularity accepted by the module is a word. Localized comments can be added on single characters, but at storing and processing time they will be bind to the word containing the character.

  There is no largest granularity imposed, a comment can be bind to as many words as the document contains. The storage of the discussions bind to several words is done taking advantage of the fact that the counter ids are consecutive. When saving the comment, the relation between itself and the context is saved relative to the smallest and largest ids of the selected words and the HTML injected markers will wrap the entire selection.

4.2 Books 3.0 - Framework for Semantic Publishing of Modular Content Objects

4.2.1 Description

The Books 3.0 component is built on top of Drupal's core module with the same name. The module allows the user to create a set of pages tied together in a hierarchical sequence, with chapters, sections and subsections. An important feature

[9] http://www.aakashweb.com/jquery-plugins/selected-text-sharer/

available in Drupal is the possibility of defining parents for each page of content type book, feature which allows us to store the content objects in the form of trees [5].

Books 3.0 handles, in the first instance, the import of semantically annotated documents together with semantic background ontologies from TNTBase [10], starting from a single document and recursively finding all its children until the deepest level is explored. An example can be observed in Figure 4, where the General Computer Science course slides, at Jacobs University Bremen, have been imported as a book in Drupal [11], starting from the cover, named "This Document".

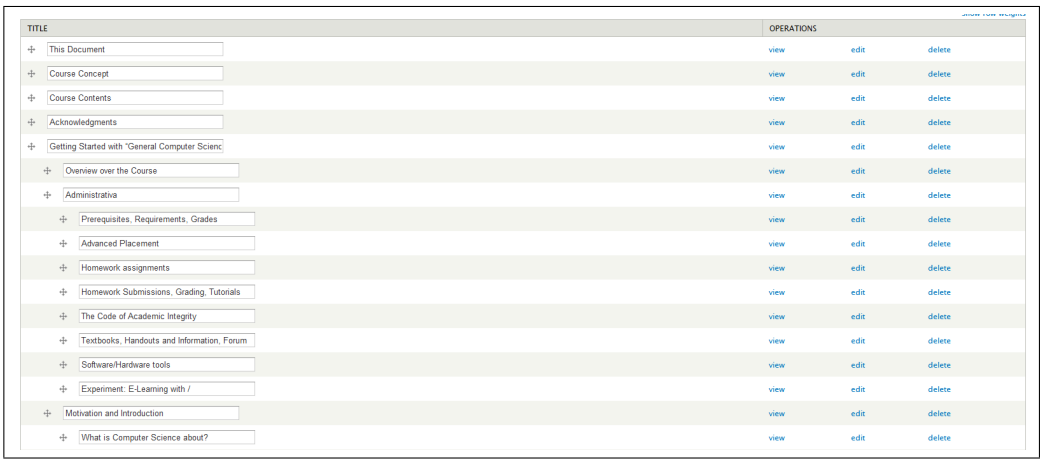

Figure 4: The outline of the first chapters of the General Computer Science course slides, hierarchically displayed.

After the book is imported, a set of Javascript bindings is applied in order to increase the usability, ease navigation and bind the independent content modules, representing now pages, to the context in which they are presented.

In the first phase, the references to the sub-chapters from the parent documents, received as TNT Paths from TNTBase, are replaced with the corresponding document titles. This is a necessary step in creating a suitable presentation for the book pages, as relative paths are irrelevant for the users of a platform unrelated to TNTBase. To each of this chapters, a further binding is applied in order to make it expandable in place. This makes the navigation a lot smoother because it allows the user to iterate through the entire book without leaving the page. Moreover, relevant pieces of content from different chapters are made available on the same page, emphasizing the context in which the information is presented (Figure 5).

A further binding in then applied in order to connect the pages to the context

[10] TNTBase is an open-source versioned storage for XML. http://tntbase.org
[11] http://panta.kwarc.info/GenCS-Notes

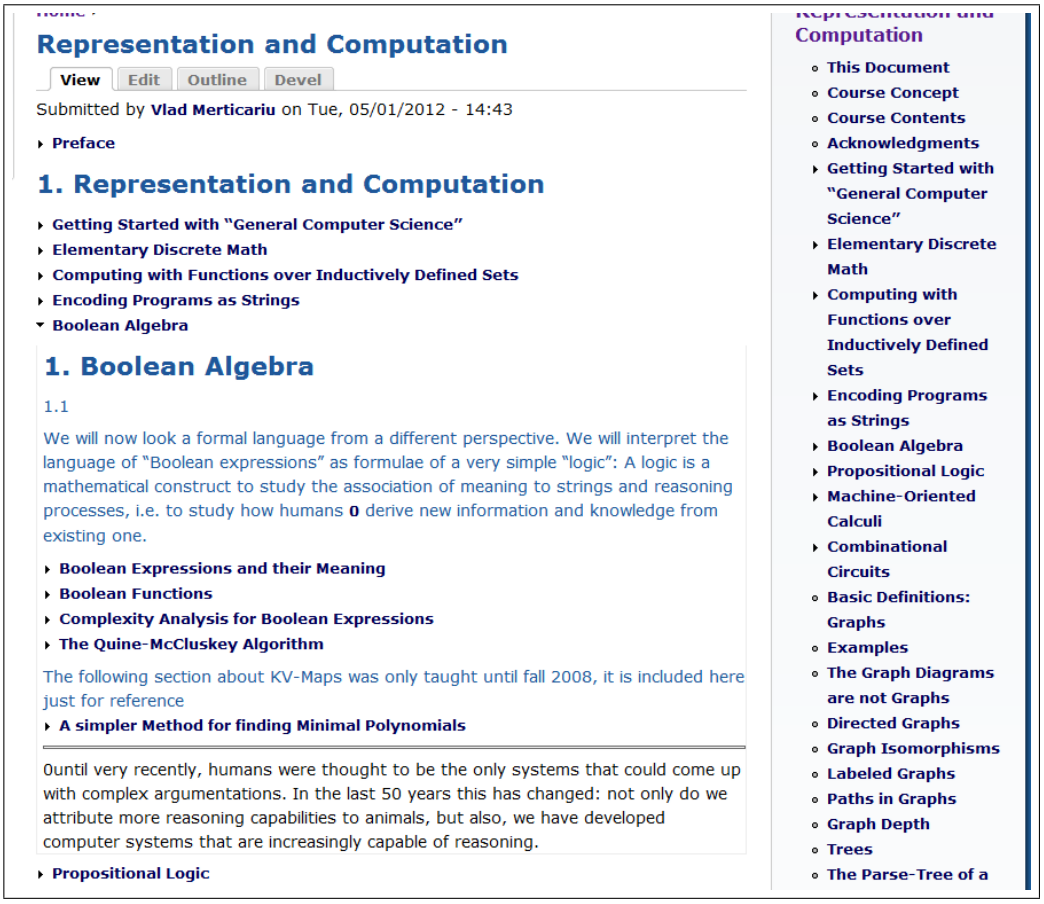

Figure 5: The first page of the ”Representation and Computation” book, with the chapter ”Boolean Algebra” expanded in place.

in which they are presented, by modifying the numbering of the chapters, sub-chapters and sections relative to the parent of the document. This feature allows the reuse of any module of content in different books without any effort from the user (Figure 6). For example, if the same page is used in notes for different courses, renumbering of the chapters to match the structure of the current notes is not necessary, as this is done automatically at the rendering time.

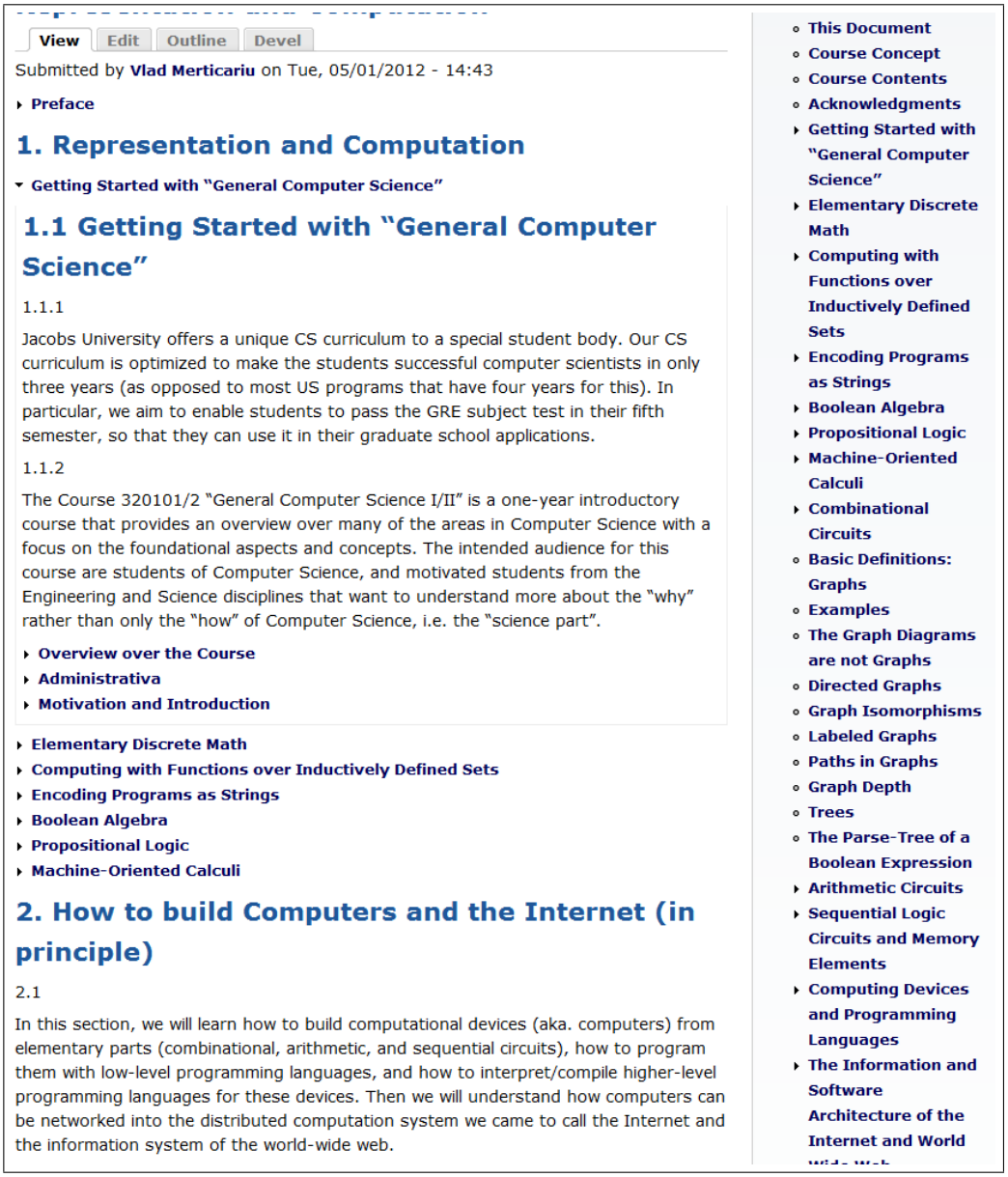

Figure 6: Numbering of the sub-chapters and sections in the "Representation and Computation" book, relative to the parent pages.

4.3 Technical Details

- **Importing documents as modular content**

  The first focus point of Books 3.0 is the import of semantically annotated

documents from TNTBase into Drupal. This is done following several steps:

- Every document contains in its body the list of children.
- A parent document is set, its path being the root of the tree we are building.
- A XQuery request is sent to TNTBase and the relative paths of the children of the root document are received [12].
- The paths are resolved into absolute paths and added on the next level of the tree.
- The procedure is repeated for every child .

Applying this simple algorithm starting from the root document, a tree representing the entire book is constructed.

- **Building books**

  The next step is creating the actual book in Drupal. The first implementation idea for this feature involved the creation of book pages in Drupal at the time the tree is built. This didn't work however as the Drupal API [12] requires the entire book tree in order to be able to correctly assign weights [13] to the pages. Because of this drawback the books are built in the following way:

  - A request is made to TNTBase and the XHTML presentation of the root of the tree is received.
  - A book page is created and is set as book cover, having no parent.
  - For every child on the following level of the tree, the XHTML presentation is requested.
  - Book pages are created for each of them, setting their parent to be the parent node in the tree.
  - Repeat for every level, until the deepest one is reached.

  After the iteration through the entire tree is completed, the documents are available in Drupal in the form of nodes [14] of type book.

- **Defining a service for easily retrieving document names and bodies**

  During the implementation, I often encountered the need to access the titles and bodies of the book pages from the client side. For this reason I imple-

[12] An application programming interface (API) is a specification intended to be used as an interface by software components to communicate with each other.

[13] In Drupal, weight is used to establish an order among content with the same parents: content with lower weight will float to the top of lists, while heavier items will sink.

[14] All content on a Drupal website is stored and treated as "nodes." A node is any posting, such as a page, poll, article, forum topic, or book.

mented a simple web service [15] that provides JSON resources for clients to consume it [13].

- **Applying expandable Javascript binding**

  After the creation of the documents in Drupal, a set of Javascript bindings is applied in order to increase their usability. The first of these bindings works following the next steps:

  - In the body of the document, sub-chapters are identified by their paths.
  - For every path found, the name of the document is received through the defined service.
  - Event listeners [16] are added to the chapters for mouse clicks.
  - At click, the body of the chapter is received through the service and displayed on the page.
  - At the following click, the body is hidden, but kept on the page, so no further requests are needed for it. In this way, at most one request per chapter is made, reducing the server load to the minimum.

  The greatest challenge in implementing this feature has been the fact that new elements are constantly added to the page. This means that event listeners have to be added for each of them in a recursive way and at the time they are added, not only when the original document is loaded. The solution to this problem has been using the Javascript delegate pattern in event listeners, which binds the event to a higher element in the DOM tree [17], that can later on distribute to all the children that satisfy the initial access rules [14].

- **Applying numbering Javascript binding**

  A further Javascript binding is applied for context adaptation of the chapters and sections numbering. Especially useful for the re-use of the same content object in different contexts, this has been implemented by the following steps:

  - Chapters and section numbers are identified, using special HTML markers provided by the semantically annotated documents.
  - The first parent of each section is selected, and its corresponding number is used as base.

[15] The W3C defines a "Web service" as "a software system designed to support interoperable machine-to-machine interaction over a network".

[16] In computer programming, event-driven programming or event-based programming is a programming paradigm in which the flow of the program is determined by eventse.g., sensor outputs or user actions (mouse clicks, key presses) or messages from other programs or threads.

[17] The Document Object Model (DOM) is a cross-platform and language-independent convention for representing and interacting with objects in HTML, XHTML and XML documents.

- A global counter is defined and initialized.
- The counter is appended to the base and the number is assigned to the first identified section.
- The counter is increased and the action is repeated until all the sections have been numbered.

5 Future Work

5.1 Localized Discussions

The next step in further developing Localized Discussions is investigating a meaningful way of displaying, in place, an entire thread related to the same piece of content. One idea on how to implement this feature is using a navigation widget in place, with a separate display for the comment bodies. This way, the user can easily navigate through an entire thread while viewing the context of the discussions. The access to the comments is straight forward and, as the navigation is done from a separate widget, the two won't create impediments for each other.

5.2 Books

For Books 3.0, several features can be added in order to increase the usability of the module. Given the strong connection with TNTBase, a synchronization mechanism between Drupal and TNT is the first step in further development. The synchronization consists of two parts:

- Constantly checking for document changes in TNT and update the Drupal version. This can be implemented in the form of a cron job [18], strictly on Drupal's server, the drawback being a high number of requests, or can be implemented both in Drupal and TNT, in the form of a callback defined by Books 3.0 which is accessed by TNT whenever a document is modified, through an HTTP POST operation.
- Synchronize documents edited in Drupal with the one in TNT base. This can be implemented by importing the documents in *omdoc* format, edit it using the Etherpad Latex editor, and recommit it to TNT once done [15] [16].

Another feature that can be integrated in Books 3.0 is creating the possibility of using the same book nodes in several books. Currently, in order to use the same page in several books, a copy of the corresponding node has to be created for

[18] cron is the time-based job scheduler in Unix-like computer operating systems. cron enables users to schedule jobs (commands or shell scripts) to run periodically at certain times or dates.

each of them due to restrictions imposed by Drupal. This can be overridden by creating a further level of abstraction of the book pages, keeping the corresponding nodes only as storage mechanisms and separating their presentation completely from Drupal's defaults.

6 Conclusions

Most of the educational platforms today are built following Web 2.0 standards. While the web is evolving, adopting new means of sharing and integrating information, there is a need for tools which can provide a progressive, incremental adaptation of the existing sites to Web 3.0 standards.

Considering the current state of the web, my focus has be creating a set of modules which centers the platform around the user, with high emphasis on context awareness as well as maintainability and extensibility, exposing an interactive way of aquiring and sharing knowledge and information.

The proposed implementation is built on top of the Planetary System and Drupal, which made a very rich starting point. However, the concepts can be implemented in a framework independent manner as well, so they suit every platform to which they might be applied.

Localized discussions and Books 3.0 constitute a small step towards Web 3.0. Along with the already existing tools provided by the Planetary System, they push the web one step further, extending the current possibilities and helping the user discover new ones.

References

[1] Deyan Ginev Bogdan Matican Vlad Merticariu Stefan Mirea Michael Kohlhase, Catalin David. A Framework for Semantic Publishing of Modular Content Objects. 2012.

[2] Steven J. DeRose and Andries van Dam. Document Structure and Markup in the FRESS Hypertext System. 1999.

[3] Theodor H. Nelson David Rice Steven Carmody, Walter Gross and Andries van Dam. A Hypertext Editing System for the /360. 1969.

[4] Andries van Dam and David E. Rice. Computers and Publishing: Writing, Editing and Printing. 1970.

[5] Dries Buytaert. Book module: Creating structured documents. http://drupal.org/documentation/modules/book. Technical report.

[6] Catalin David Deyan Ginev Andrea Kohlhase Bogdan Matican Stefan Mirea Christoph Lange, Michael Kohlhase and Vyacheslav Zholudev. The planetary system: Executable science, technology, engineering and math papers. 2011.

[7] Cristina Tofan. The application of drupal to website development in academic. 2010.

[8] Ivo Van Geertruyen. Expanding user profiles. http://drupal.org/documentation/modules/profile. Technical report, December 2010.

[9] Paul Brennan. Configuring comments. http://drupal.org/documentation/modules/comment. Technical report, May 2011.

[10] W3C Bert Bos. Cascading style sheets. http://www.w3.org/style/css/. Technical report.

[11] Drew Douglas. Javascript events. http://www.w3schools.com/js/. Technical report.

[12] Robert MacKay. Xquery. http://www.w3schools.com/xquery. Technical report.

[13] Jason Langstorf. Json, what is and how to use it. http://www.copterlabs.com/blog/json-what-it-is-how-it-works-how-to-use-it. Technical report.

[14] Ross Harmes and Dustin Diaz. Pro Javascript Design Patterns. 2008.

[15] Michael Kohlhase. An Open Markup Format for Mathematical Documents. August 2009.

[16] Constantin Jucovschi. Editing knowledge in large mathematical corpora. a case study with semantic latex (stex). 2010.