Novelle, a collaborative open source writing tool software

Federico Gobbo
DICOM
21100, Insubria University
Varese, Italy
federico.gobbo@uninsubria.it

Michele Chinosi
DICOM
21100, Insubria University
Varese, Italy
michele.chinosi@gmail.com

Massimiliano Pepe
DICOM
21100, Insubria University
Varese, Italy
massimiliano.p@gmail.com

Abstract

In this paper we discuss the notions of hypertext, blog, wiki and cognitive mapping in order to find a solution to the main problems of processing text data stored in these forms. We propose the structure and architecture of Novelle as a new environment to compose texts. Its flexible model allows the collaboration for contents and a detailed description of ownership. Data are stored in a XML repository, so as to use the capabilities of this language. To develop quickly and efficiently we choose AJAX technology over the Ruby on Rails framework.

1 Introduction

Computational linguists are facing the explosion of new forms of writing as a mass phenomenon. Telling personal and collaborative stories throught web technologies is known under the etiquettes of 'blog' and 'wiki'. It therefore brings new challenges to the field of natural language processing. We are trying to address them by rendering explicitly the structure of these new forms of text in a way suitable for linguistic computation. In order to do so, we are building an open source writing tool software, called Novelle.

1.1 Hypertext as a New Writing Space

Bolter (1991) was the first scholar who stressed the impact of the digital revolution to the medium of writing. Terms as 'chapter', 'page' or 'footnote' simply become meaningless in the new texts, or they highly change their meaning. When Gutenberg invented the printing press and Aldo Manuzio invented the book as we know it, new forms of writings arose. For example, when books shouldn't be copied by hand any longer, authors took the advantage and start writing original books and evaluation – i.e. literary criticism – unlike in the previous times (Eisenstein, 1983). Nowadays the use of computers for writing has drammatically changed, expecially after their interconnection via the internet, since at least the foundation of the web (Berners-Lee, 1999). For example, a 'web page' is more similar to an infinite canvas than a written page (McCloud, 2001). Moreover, what seems to be lost is the relations, like the texture underpinning the text itself. From a positive point of view these new forms of writing may realize the postmodernist and decostructionist dreams of an 'opera aperta' (open work), as Eco would define it (1962). From a more pessimistic one, an author may feel to have lost power in this openness. Henceforth the collaborative traits of blogs and wikis (McNeill, 2005) emphasize annotation, comment, and strong editing. They give more power to readers, eventually filling the gap - the so-called active readers become authors as well. This situation could make new problems rise up: Who owns the text? Which role is suitable for authors? We have to analyse them before presenting the architecture of Novelle.

1.2 Known problems

It is certainly true that wikis and blogs are new forms of text. It is also true that we have already met these problems in the first form of purely digital texts – hypertexts. Now we are facing the same question during processing texts in blogs and wikis. We consider hypertexts as parents of blogs and wikis. Our aim is to use the analysis of hypertexts for interesting insights, useful for blogs and wikis too.

Following the example of Landow (1994), we will call the autonomous units of a hypertext *lexias* (from 'lexicon'), a word coined by Roland Barthes (1970). Consequently, a hypertext is a set of lexias. In hypertexts transitions from one lexia to another are not necessarily sequential, but navigational. The main problems of hypertexts, acknowledged since the beginning, have been traced as follows (Nelson, 1992):

- *The framing problem*, i.e. creating arbitrary closed contexts of very large document collections. When extracting sub-collections, some links may be cut off.
- *Comparing complex alternatives*, i.e. to get parallel or alternate versions of the same document in a simple and effective way, one of the main goal of Xanadu, the ultimate "global hypertext" dreamt by Nelson.
- *Typology of links*, i.e. when links become too many, we need a typology for links, avoiding confusion to the reader/author.
- *Version control*, as the system should keep track of the history of every document, saving differences and pointing out correspondencies.

We take from wikis the concept of document history and its consequences. We consider it as a good approximation of the 'version control' concept as shown above.

In wikis every document keeps track of its own history: *creating* a document means to start a history, *editing* a document to move ahead, *restoring* to move back onto the history timeline, *destroying* a document to stop the history itself. Moreover, a *sandbox* is a temporary view of a document itself - i.e. a sandbox can not cause a change in the history (Cunningham and Leuf, 2001). Figure 1 shows the model.

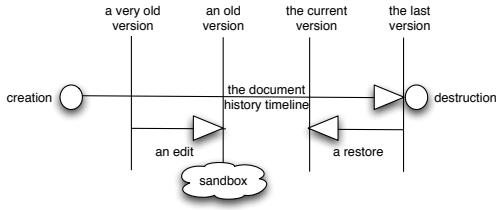

Figure 1: The document history model

History snapshots of the timeline may be considered as permanent views, i.e. views with a timestamp. Consequently, except in the case of sandboxes, every change in the document cannot be erased. This model will have a strong impact on the role of links and on the underpinning structure of Novelle itself.

2 The Structure of Novelle

Our aim is to create an open source hypertext modeling software, called Novelle. 'Novelle' is an Italian old-fashioned word meaning 'novels', and in German it means 'novel' too. It resembles the English word 'novel' and the French word 'nuovelle'. We believe that this name is clearly understable to every people educated in a European-based culture, and this is why we have chosen it.

The emphasis on narrativity takes into account the use of blogs as public diaries on the web, that is still the main current interpretation of this literary genre, or *metagenre* (McNeill, 2005). Furthermore we noticed that blogs and wikis are currently subjected to osmosis, because they have in common the underlying core technology. So blogs are a literary metagenre which started as authored personal diaries or journals. Now they try to collect themselves in so-called 'blogspheres'. On the other side, wikis started as collective works where each entry is not owned by a single author - e.g. Wikipedia (2005). Now personal wiki tools are arising for brainstorming and mind mapping. See Section 4 for further aspects.

2.1 The Problem of Ownership

The main difference between blogs and wikis is in the ownership of documents. Most blogs follow the *annotation model*, where a single lexia is central and the others are comments, sometimes in threads. Every lexia is authored and changes are minimal. People prefer commenting instead of editing. The paradigm is "write once, read many".

On the contrary, in wikis no lexia is authored and there is no hierarchy between lexias. In fact a document is still a set of lexias, but every document is only the set of historical versions of the document itself. Generally, people avoid commenting, preferring to edit each document. The paradigm is "write many, read many" (Cunningham and Leuf, 2001).

We believe that ownership has an important role and we do not want to force our users to take a

non-attributive copyright licence to their work. We consider the Creative Commons model as the most suitable one to let each author choose the rights to reserve (Lessig, 2004). Narrative writings or essays are creative works and they generally treat ownership as authorship, even for the most enthusiastic fellows of free culture (Stallman, 2001).

2.2 The Representation of Context

In the structure of Novelle we are trying to retain authorship and the core concept of document history of wikis through a typology of links, taking what we consider the best of the two worlds of blogs and wikis.

In Novelle each user owns his own lexias, and the relations between them, i.e. links. Furthermore authors are free to read and to link other users' lexias. In other words, each user does permit everyone to link its own lexias for free, at the condition that the others do the same. Every user may recall the link list on each element (e.g. a single word) of his lexias at every time, but he can not destroy them. Lexias may be commented by every user, but the author may retain for himself the right to edit. This decision has to be taken when a lexia is created.

If a user lets others edit some lexias, he has the right to retain or refuse the attribution when other users have edited it. In the first instance, the edited version simply moves ahead the document history. In the second one, the last user, who has edited the lexia, may claim the attribution for himself. The lexia will be marked as a derivative work from the original one, and a new document history timeline will start (see Figure 2). Authors may choose this right with the No-Deriv option of the Creative Commons licences (Lessig, 2004).

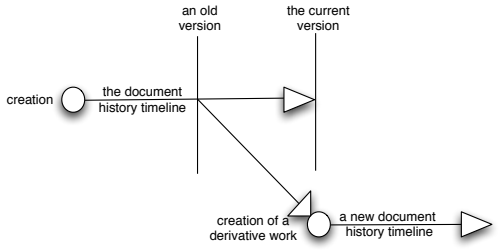

Figure 2: How to create derivative works

If nobody claims the document for himself, it will fall in the public domain. The set of lexias in the public domain will form a special document, owned by a special user, called Public Domain. If the author refuses the permission to create derivative works, i.e. to edit his own lexias, users still have the right to comment the author's work. So as to come to terms with this idea, we need a concept invented by Nelson (1992), i.e. *transclusion*.

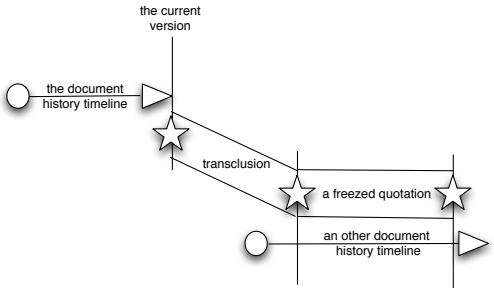

Figure 3: How transclusion works

Rather than copy-and-paste contents from a lexia, a user may recall a quotation of the author's lexia and write a comment in the surroundings. In doing so, the link list of the author's lexia will be updated with a special citation link marker, called *quotation link* (see later for details). Usually, the quotation will be 'frozen', as in the moment where it was transcluded (see Figure 3). Consequently the transclusion resembles a copied-and-pasted text chunk, but the link to the original document will always be consistent, i.e. neither it expires nor it returns an error. Otherwise the user who has transcluded the quotation may choose to keep updated the links to the original document. This choice has to be made when the transclusion is done.

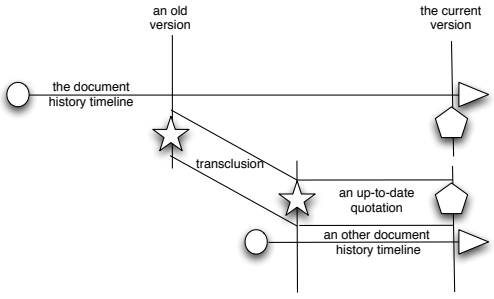

Figure 4: An up-to-date transclusion

If so, the transcluded quotation will update automatically, following the history timeline of the original document. For example, if the original

document changes topic from stars to pentagons, the quotation transcluded will change topic too (see Figure 4).

2.3 Contents and the Typology of Links

Following our model of ownership, there are at least two categories of links: shallow links and deep links. By *shallow links* we mean visual links occurring in a single canvas, usually owned by the same author. These will represent iconically the relationship space of lexias, as explained by McCloud, talking about web comics (2001). They are particularly useful when comparing parallel versions of the same text, e.g. digital variants (see Conclusions).

We consider a web page, or better a web canvas, as a *view of lexias*, i.e. a group of lexias and their relations visually shown with shallow links. A set of lexias is a *document*. Every author has the right to decide the relation type of a set of lexias, i.e. to form a document. A document can also be considered as a collection of history timelines, i.e. the set of related lexias and their versions. The set of documents is the *docuverse*, a word coined by Nelson (1992). We use the word docuverse, unlikely the original sense, with the meaning of a set of documents owned by a single author.

Every document can be viewed within a web canvas, but users may click on a deep link and so change view. With *deep links* we mean links which let the user change view, i.e. rearrange elements in the web canvas for revealing shallow links between lexias, belonging to the same document or not. Therefore a web canvas may show relations between views owned by different authors. We consider quotation links, i.e. links created by transclusion, as a special kind of deep links. Authors may create specific views adding labels to links. The set of labels will form a typology of links, customized by every user and even shared, on demand of users' desires.

With our typology of links, we aim to solve the framing problem as defined in Section 1.2. We want to model views as dynamic objects - the creation of context will be still arbitrary, but changes are very easily. We would also provide a user facility for choosing the right licence for every lexia, following the model of Creative Commons licences (Lessig, 2004).

3 The Architecture of Novelle

We have considered many hypotheses in order to choose a first layer architecture to save a repository. We used a multi-tier model based on XML. Our idea is based on merging together some of the most common design techniques used in blogs and wikis. Recently previous implementation techniques have been studied from their new aspects to find innovative web technologies. A basic scheme of Novelle architecture is presented in Figure 5. The first layer is the most important. It is based on

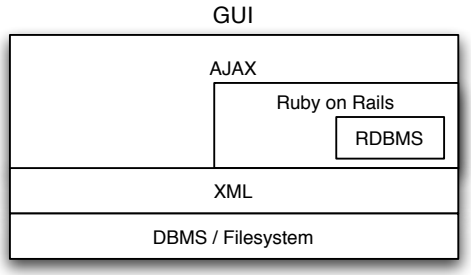

Figure 5: Novelle: multi-tier architecture

an infrastructure for storing effectively data repository in order to obtain the best performances. We have studied two alternatives for the repository. On one side we have different techniques to map XML trees onto a database management system. On the other side we may map XML trees directly on a filesystem – see below for details.

The second layer is represented by XML. Messages, data and metadata are exchanged between layers using the capability of this language. This allows to treat data and metadata on different level of abstraction.

The *Ruby on Rails* (2006) framework permits us to quickly develop web applications without rewriting common functions and classes.

We used the *Asyncronous Javascript And XML* (or AJAX) paradigm to create the graphical user interface. AJAX function lets the communication works asyncronously between a client and a server through a set of messages based on HTTP protocol and XML (Garrett, 2005).

3.1 XML repository

We chose to use XML trees to store together data, metadata, messages and their meanings because it has some benefits. The most important is storing

XML data. The other benefits of a native XML solution are: the storing without mapping your XML to some other data structure like objects, classes or tables; the neatness of the structure; the underlying technology from the abstract layer to the physical ones is based on a unique standard, widely accepted by the community. Data may be entered and retrieved as XML. Another advantage is flexibility, gained through the semi-structured nature of XML and the schema independent model used by most of native XML databases. This is especially valuable when you have very complex XML structures that would be difficult or impossible to map to a more structured database. At this time there are not XML databases so stable to be used into project of this kind.

Xindice (developed by Apache Group) proved better than others. Apache Xindice is a database designed from the ground up to store XML data or what is more commonly referred to as a native XML database. It stores short XML documents in collections with runtime generation of indexes. Unfortunately Xindice seems not to have been developed any more since April 2004.

Another native XML database, more usable and supported, is *eXist*. eXist is growing quickly and it implements some functionalities of Xindice. The standards support is not completed and some functions are currently being rewritten directly embedded into the software. After doing many tests on it, it reveals worse performances with respect to other platforms, even if it is more complete in comparison to the others.

Anyway our interest keeps focusing on them waiting for the first stable release effectively usable in Novelle. We have considered the possibility to map XML trees to relational or object-oriented database management systems that support XML. We can map directly an XML tree into a memory tree structure, made up with classes and objects, with object-oriented databases, as we can see in *Ozone* project (2006). The last stable version of Ozone was released in 2004. The main problem with Ozone - and with others OODBMS - is the overhead requested to the memory for storing a complex tree. On the other side, many RDBMS with XML support map directly an XML tree to an entity-relationship schema. In some instances XML trees are stored as Binary Large Object, or BLOB, into one big table. In other situations XML trees are parsed, splitted and finally stored in tables where attributes have the same names than XML nodes.

Ronald Bourret (2006) mantains and updates a very comprehensive list of native XML databases on his web site.

While we are waiting for a native XML database stable and useful for our project, we have decided to get inspiration from the common idea used in many blogs and wikis. Most of these architectures are used to store messages in a structure that is similar to a directory tree saved on filesystem. Often this idea is only developed to present to users messages organized in collection ordered by time (e.g. blogs), but all the platforms are based on RDBMS. We have found in our research only one other project in which messages are stored directly on filesystem: the *Gblog* project (Gblog, 2005). Nobody usually adopt this solution because the security of the web site is less strong. In order to represent messages archives the most common structure is the triple *../year/month/day/...* In our assumption, a message is a history. Therefore a structure of this kind works very well with our idea. We are going to build a filesystem time-based structure in which we can directly map our messages i.e. our histories. This structure is also a tree. We can write also an XML document that mantains an architecture scheme with some indexes to speed up queries. Moreover, we store with a message another XML document representing all the past history (i.e. the paths) of the message.

So as to sum up, every time a user stores a message, he has to save the first XML document with the message, then saves or updates a second XML document representing its past history and finally saves or updates a third XML message with filesystem directory tree. The overhead on bandwith and net speed of this solution does not let users notice significant differences, even though it is necessary to grant writing permissions to everyone on the entire repository. Having a native XML database will give the advantage of saving XML documents in a rapid, neat and indicized way, in order to be able to execute efficient queries on the repository.

3.2 eXtensible Markup Language

We chose XML as language and meta-language because we needed to be able to save messages with their meanings. Every lexia is saved with

some tags and attributes which describe its meaning. The possibility of storing separately data from their representations lets a system access more quickly to a data and extract the requested information. XML is a W3C standard and this makes our project ready to be changed and extended, as well as to be connected with other applications and services (XML, 2005). XML will be used to represent data, metadata, link typing, messages and paths map, and to exchange messages betweeen different layers.

3.3 Ruby on Rails

Ruby on Rails, or RoR, is a framework rich in extensions and libraries with licences suitable for our usage, in particular *XML Builder* and *gdiff/gpatch*. The first library offers a set of classes which allows to generate XML code in a simple way (Builder, 2006). Gdiff/gpatch library is an implementation of the *gdiff* protocol, that creates a patch from two files and then a new file from one of the previous files and the patch (Gdiff, 2005). Using this library we are going to be able to store the history and the last version in an easy way and saving space. Creating a document is therefore a sequence of patches. Storing works in the same way, that is executing a gdiff protocol and storing the new patch. Moving across the document history means retrieving a number of patch commands until you reach the desired version of the document.

Ruby on Rails does not support native XML databases at this time, therefore in our architecture there will be provisionally a relational DBMS dedicated to RoR, which had no problem with a filesystem repository.

3.4 Asyncronous Javascript And XML

AJAX is not a technology in itself but a term that refers to the use of a group of technologies together, in particular Javascript and XML. In other words AJAX is a web development technique for creating interactive web applications using a combination of XHTML and CSS, Document Object Model (or DOM), the *XMLHTTPRequest* object (Wikipedia, 2005).

AJAX paradigm has been recently defined, when someone has rediscovered a simple function originally developed by Microsoft as ActiveX control. This function, named *XMLHTTPRequest* lets clients ask servers for some particular data using asyncronous handshake. In this way users can continue using web application (typically filling web forms) while the client and the server exchange data and messages. Other developers have published a concurrent version of this function for other browsers than Internet Explorer, like Mozilla/Gecko, Opera and Safari. The web pages builded with this technology give the impression to have dynamic content. Important examples builded with AJAX paradigm are Gmail by Google, Writely, Kiko, Webnote, Meebo. Using AJAX to develop web applications and web services needs some attention. First of all Javascript must not be disabled in browsers. It is also necessary to pay attention to estimate correctly the time spent in exchanging messages between client and server so to exploit the good capabilities gained with AJAX, fully supported by and integrated in Ruby on Rails.

3.5 Access points

We are going to add to every view of Novelle a search engine that returns a list of meanings and a set of link between them. These links are represented in our project with images. Every image is itself a map that the user can surf and/or open to increase details level. When the user has found the message, he can access to it simply clicking on it. An user can comment or modify every lexia, if these actions are granted by the original author, as explained above. Users can create new links between lexias and they can describe what kind of link they intend to create through appropriate link type. These modifications are stored using the document history model of Novelle through following patch.

4 Related Works

The main source of Novelle are wikis and blogs. While wikis have spread from a detailed design (Cunningham and Leuf, 2001), unfortunately blogs have not been designed under a model. So we have tested and compared the most used tools available for blogging: Bloggers, WordPress, MovableType and LiveJournal.

Generally speaking, we find that the personal public diary metaphor behind blogs (McNeill, 2005) may bring to an unsatisfactory representation of the context. The only way to retrieve information is through a search engine or a calendar, i.e. the date of the ‘post’ – a lexia in the jargon of bloggers.

Moreover, we use some new web applications

to take and share notes or to browser everyone's bookmarks, e.g. del.icio.us. Mostly, these web applications oriented to writing give a strong emphasis on collaboration and sharing. This led us to rethink ownership and to use the Creative Commons model to design the contents of Novelle.

Finally, we noticed that personal wikis are used for storing cognitive maps of individuals and brainstorming. This use was already thought by the founders of wikis (Cunningham and Leuf, 2001), but it has not been widely explored in practics, as far as the authors know. However, this direction of work is not actually new - concept and mind mapping, the two main paradigms for cognitive maps, have been used for several years.

Concept mapping has been used at least in education for over thirty years, in particular at the Cornell University, where Piaget's ideas gave the roots to the *assimilation theory* by David Ausubel. Very briefly, concept maps show the relationships between concepts labelling both nodes and arcs. Every arc always has a definite direction, i.e. arcs are arrows (Novak, 1998).

In contrast, mind maps spread from a centre, with branches radiating out. Furthermore, mind maps, as thought and copyrighted by Tony Buzan, can label only nodes, not arcs. The resulting shape of mind maps is sometimes similar to neurons' (Buzan, 2000).

We have tested both concept and mind mapping software tools, available for free or in a trial period. In particular, CmapTools software is currently used at the Cornell University and it is free as a client. It may run on CmapServers, and it is a very good way to share the knowledge stored in cognitive maps. Unfortunately, it does not collect data in a format suitable for the web, and it does not permit to view concepts across cognitive maps owned by different users (Tergan, 2005). More, concept maps require a learning curve very high when started to be used, at least in our experience. On the contrary, mind maps are by far more intuitive.

There are a lot of mind mapping tools, which are clones of MindJet MindManager, the official software for Buzan's mind mapping. The mind mapping tool we were looking for should have had an open source licence, used a format for data storage suitable for the web, and been cross-platform. In fact, Freemind, as the closest approximation of our needs (Mueller, 2000), succeeded in running on the three major operating systems available, without sensible differences.

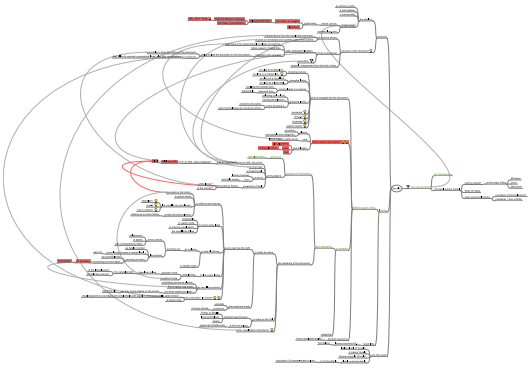

Figure 6: Our free mind map for Novelle

Even if we like the idea behind mind maps, we need to have a multiauthored environment, where arcs may be labeled. In other terms, the centre of the map should change according to the user's desire. That is why we thought about web canvas as document views. If we consider documents as free mind maps, the nodes will be lexias and the arcs will be links.

Apart from wikis, blogs, and cognitive mapping, we were also inspired by the experiences of early hypertext writing tools, in particular Intermedia and Storyspace. In fact, they were used expecially in academic writing with some success. Intermedia is no more developed and nobody of us had the opportunity to try it (Landow, 1994). Storyspace is currently distributed by Eastgate (2005), and we have used it for a time. However, in our opinion Storyspace is a product of its time and in fact it isn't a web application. Although it is possible to label links, it lacks a lot of features we need. Moreover, no hypertext writing tool available is released under an open source licence. We hope that Novelle will bridge this gap - we will choose the exact licence when our first public release is ready.

We are persuaded that there is no contradiction in collaborative mind mapping and academic writing. Maybe it is not by chance that Eastgate has also released a "personal content management assistant" (Eastgate, 2006). Our purpose is to bring back again collaborative writing and free brainstorming, as it should be.

5 Conclusions and Further Works

We are currently developing a prototype of Novelle. We argue that the model under Novelle would be an explicit representation of the context and a clear model for the contents. One of the main application of our software is natural language processing. We are going to test it expecially on digital variants of literary texts.

Acknoweledgements

We want to acknowledge Dr. Marco Benini, Dr. Alberto Trombetta for their deep insights and Gabriella Canciani for having reviewed the final draft of this paper.

References

Roland Barthes. 1970. *S/Z*. Editions du Seuil, Paris.

Tim Berners-Lee. 1999. *Weaving the Web*. Harper, San Francisco.

Jay David Bolter. 1991. *Writing Space: the Computer, Hypertext, and the History of Writing*. Erlbaum Associates, Hillsdale, N.J.

Ronald Bourret. 2006. *XML Database Products*. Url: http://www.rpbourret.com/xml/. Retrieved the 3^{rd} of January.

Builder library.for example 2006. *Project: Builder. Provide a simple way to create XML markup and data structures*. Url: http://builder.rubyforege.org/. Retrieved the 4^{th} of January.

Tony Buzan and Barry Buzan. 2000. *The Mind Map Book*. BBC Worldwide Limited, London.

Ward Cunningham and Bo Leuf. 2001. *The Wiki Way - Quick Collaboration on the Web*. Addison-Wesley, Boston.

Eastgate 2005. *Storyspace*. Url: http://www.eastgate.com/storyspace. Retrieved the 31^{st} of December.

Eastgate 2006. *Tinderbox*. Url: http://www.eastgate.com/tinderbox. Retrieved the 2^{nd} of January.

Umberto Eco. 1962. *Opera aperta*. Bompiani, Milan, Italy.

Elizabeth L. Eisenstein. 1983. *The Printing Revolution in Early Modern Europe*. Cambridge University Press, Cambridge, UK.

Jesse James Garrett. 2005. *Ajax: A New Approach to Web Applications*. Url: http://www.adaptivepath.com/publications/essays/ /archives/000385.php. Retrieved the 22^{nd} of December.

Gblog 2.0. 2005. *Gblog 2.0. Blog, reloaded*. Url: http://gblog.com/. Retrieved the 27^{th} of December.

Gdiff/Gpatch library. 2005. *Gdiff/Gpatch. An implementation of the W3C gdiff protocol*. Url: http://ruby.brian-schroeder.de/gdiff/. Retrieved the 28^{th} of December.

George P. Landow 1994. *Hypertext 2.0. The Convergence of Contemporary Critical Theory and Technology*. The Johns Hopkins University Press, Baltimore, Maryland.

Lawrence Lessig 2004. *Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity*. Penguin Books.

Scott McCloud. 2001. *Understanding Comics*. Paradox Press, NY.

Laurie McNeill. 2005. Genre Under Construction: The Diary on the Internet. *Language@Internet*, 1.

Joerg Mueller. 2000. *FreeMind*. Url: http://freemind.sourceforge.net. Retrieved the 31^{st} of December 2005.

Theodor Holm Nelson. 1992. *Literary Machines 90.0*. Muzzio, Padua, Italy.

Joseph Donald Novak. 1998. *Learning, Creating, and Using Knowledge: Concept Maps As Facilitative Tools in Schools and Corporations*. Lawrence Erlbaum Associates.

Ozone Database Project. 2006. *Ozone Database Project open initative*. Url: http://ozone-db.org/. Retrieved the 03^{rd} of January.

Ruby on Rails. 2006. *Ruby on Rails. Web developement that doesn't hurt*. Url: http://www.rubyonrails.org/. Retrieved the 03^{rd} of January.

Richard M. Stallman. 2001. *Free Software, Free Society: Selected Essays of Richard M. Stallman*. GNU Press, Cambridge, Massachusetts.

Sigmar-Olaf Tergan and Tanja Kellers. 2005. *Knowledge And Information Visualization: Searching for Synergies*. Springer, Berlin.

Dave Thomas and David Heinemeier Hansson. 2005. *Agile Web Development with Rails - A pragmatic guide*. Pragmatic Bookshelf.

Wikipedia. 2005. *Wikipedia. From Wikipedia, the free encyclopedia*. Url: http://en.wikipedia.org/wiki/Wikipedia. Retrieved the 31^{st} of December.

XML, eXtensible Markup Language. 2005. *Extensible Markup Language (XML)*. Url: http://www.w3.org/XML/. Retrieved the 27^{th} of December.