Designing MeanWriter, an HTML5 element-oriented WYSIWYM writing and reading tool for e-learning environments

University of Crete
Department of Computer Science

Kainourgiakis Giorgos
Master's thesis

April 15, 2014

University of Crete
Department of Computer Science

Designing MeanWriter, an HTML5 element-oriented WYSIWYM writing and reading tool for e-learning environments

Kainourgiakis Giorgos

A thesis submitted in partial fulfillment of the
requirements for the degree of
Master of Science

Author .....
Kainourgiakis Giorgos

Supervisor .....
Nikolaou Christos, Professor

Member .....
Spantidakis Ioannis, Professor

Member .....
Plexousakis Dimitrios, Professor

Approved by .....
Bilas Angelos, Professor, Chairman of the Graduate Studies Committee

Heraklion, April 2014

University of Crete
Department of Computer Science

Designing MeanWriter, an HTML5 element-oriented WYSIWYM writing and reading tool for e-learning environments

Kainourgiakis Giorgos

Abstract

The evolution of writing technology from early writing instruments to modern day computers, created a great impact in social organization, cultural transmission, learning, communication and language. Yet, early word processing systems exhibited issues in the visual correspondence between the author's display and the printed output, which was a critical problem in a paper dependent world. The rising of graphical user interfaces and the WYSIWYG (What You See Is What You Get) writing approach in the early 1970's restored this correspondence, allowing the user to visually format documents and produce fixed sized formats capable of printing. This approach is still dominant in word processing, desktop publishing as well as web authoring systems, despite the recent technological breakthroughs and the growing independence from paper. The rising of the Internet and mobile devices triggered the creation of WYSIWYM (What You See Is What You Mean) approach, as well as the use of reflowable content instead of fixed sized formats. WYSIWYM reduces the visual formating capabilities and encourages authors to semantically annotate the document's structure, providing an automatic visual feedback for every structural element. Moreover, the gradual penetration of Learning Management Systems in educational practice, raised the interest for learning theories and instruction with new technologies. In this thesis, we present the design and implementation of MeanWriter, an HTML5 WYSIWYM writing and reading tool that incorporates features deriving from writing research and modern learning theories. Apart from the simple interface, MeanWriter provides facilitations for planning, organizing and revising documents, as well as for the customization of the visual layout that satisfies the preferences and the visual abilities of the reader.

Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών

Σχεδιάζοντας τον MeanWriter, ένα HTML5 WYSIWYM εργαλείο σύνταξης και ανάγνωσης κειμένου για εκπαιδευτικά περιβάλλοντα

Καινουργιάκης Γιώργος

Περίληψη

Η εξέλιξη της τεχνολογίας της γραφής από τα πρώιμα εργαλεία στον σύγχρονο υπολογιστή, δημιούργησε σημαντικές αλλαγές στην κοινωνική οργάνωση, την μετάδοση του πολιτισμού, την μάθηση, την επικοινωνία και την γλώσσα. Παρόλα αυτά, τα πρώτα συστήματα επεξεργασίας κειμένου παρουσίαζαν προβλήματα την οπτική αντιστοιχία μεταξύ της οθόνης του συγγραφέα και της τελικής εκτύπωσης, γεγονός που αποτελούσε ένα σημαντικό πρόβλημα σε έναν κόσμο εξαρτημένο από το χαρτί. Η άνοδος των γραφικών περιβαλλόντων και η προσέγγιση WYSIWYG (What You See Is What You Get) στις αρχές της δεκαετίας του ′70, αποκατέστησαν την αντιστοιχία αυτή, επιτρέποντας στον χρήστη να μορφοποιεί τα έγγραφα του και να παράγει αρχεία κατάλληλα για εκτύπωση. Η προσέγγιση αυτή είναι ακόμη κυρίαρχη στην επεξεργασία κειμένου, στις ηλεκτρονικές εκδόσεις και στην δημιουργία ιστοσελίδων, παρόλες τις σύγχρονες τεχνολογικές καινοτομίες και την αυξανόμενη ανεξαρτησία από το χαρτί. Η άνοδος του διαδικτύου και των κινητών συσκευών αποτέλεσαν το έναυσμα για την δημιουργία της προσέγγισης WYSIWYM (What You See Is What You Mean), αλλά και για την χρήση δυναμικού περιεχομένου αντί των στατικών μορφών. Η προσέγγιση αυτή μειώνει τις δυνατότητες μορφοποίησης και ενθαρρύνει τους χρήστες να ορίσουν σημασιολογικά την δομή του κειμένου, ενώ προβάλλει μια αυτόματη οπτική αναπαράσταση για κάθε δομικό στοιχείο. Επίσης, η σταδιακή διείσδυση των ηλεκτρονικών συστημάτων μάθησης στην εκπαιδευτική πρακτική, αύξησαν το ενδιαφέρον για τις θεωρίες μάθησης και την διδακτικής με νέες τεχνολογίες. Στην εργασία αυτή, παρουσιάζουμε τον σχεδιασμό και την υλοποίηση του MeanWriter, ενός HTML5 WYSIWYM εργαλείου σύνταξης και ανάγνωσης κειμένου, που ενσωματώνει λειτουργίες που προέρχονται από την έρευνα του γραπτού λόγου και των σύγχρονων θεωριών μάθησης. Εκτός από την απλή διεπαφή του, ο MeanWriter παρέχει διευκολύνσεις για τον σχεδιασμό, την οργάνωση και την αναθεώρηση κειμένων, αλλά και την προσαρμογή της όψης του κειμένου στις προτιμήσεις και στις ικανότητες όρασης του αναγνώστη.

This document is dedicated to
my wife and daughter

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The evolution of writing instruments and materials from reeds and Mesopotamian clay tablets to modern day computers and writing applications was slow with very distant milestones. Throughout history, writing transformed social organization, cultural transmission, learning, communication and language, while breakthroughs in writing technology, such as paper and the printing press fundamentally changed the world. Traditionally, authors used writing instruments to inscribe on writing materials, which stored and displayed a static view of information in their surface. With the emergence of the computer, the storage of information separated from its display, which in early word processing systems disrupted the static visual correspondence between the information of the video screen to the printed output. This was one of the most common problems until the rising of graphical user interfaces and the "What You See Is What You Get" (WYSIWYG) writing approach. Researchers at Xerox-PARC developed this paradigm of interaction back in 1973, in order to restore the correspondence of the video screen to the printed output. Their solution involved the WIMP interface (Windows, Icons, Menus, and Pointer) and the WYSIWYG approach, which enabled authors to visually format documents.

Despite the fact that WIMP and WYSIWYG are still leading in contemporary writing applications, recent technological breakthroughs started to challenge these approaches. Mobile devices introduced new touch interfaces changing the rules of interaction, as well as revealed the shortcomings of fixed sized formats in varying screen sizes. The rising of Web 2.0 and the vision of the "web as a platform" delivered web applications, such as blogs and wikis, as well as many multimedia sharing services. Modern writing applications introduced the "What You See Is What You Mean" (WYSIWYM) writing approach, which enabled authors to easily markup the structural elements of documents, without adding any visual formating. Moreover, syncing cloud

services enabled the remote storage and manipulation of information, challenging traditional local storage techniques. At last, the distribution of content through the world wide web gradually reduced the dependence from paper as a storing and distributing media, and accordingly the need for document correspondence to printed outputs.

Another crucial issue of mainstream writing applications, such as word processors, HTML editors, or wikis, is the detachment from modern writing research and learning theories. Findings deriving from writing research underline the significance of planning, organizing and revising in the writing process, while they identify problems and propose strategies for effective writing. On the other hand, learning theories stress the importance of new instructional applications, involving simulations, videos, images and games in the learning process. These developments should inform writing applications' design, since they facilitate the document's creation and communicate its content.

1.1 Thesis objectives

The first objective of this thesis is to provide a historical review of the evolution of writing technology and identify different contemporary writing applications and approaches. Moreover, we use findings from learning theories and writing research, in order to inform our design and facilitate the processes of learning and writing. This research forms requirements that could address contemporary needs in learning, writing and reading, in a rapidly changing technological environment. The final objective of this thesis is the development of a web-based writing and reading tool, using emerging HTML5 technologies (HTML5, Javascript, CSS3 and Ajax) for easy integration in e-learning platforms and educational environments.

1.2 Thesis contribution

This thesis provides a new perspective in writing applications' design, by encouraging authors to explicitly markup document structure, while at the same time restricts the visual formating of the document, following the WYSIWYM approach. Instead of visual formating, this concept provides an automatic global visual presentation tailored to the preferences of the author or the reader, allowing the users to select fonts, font sizes, themes, line spacing, text alignment and the width of the document. This feature along with the capability of the user to highlight and annotate text, contribute to the reading dimension of the application. Moreover, this thesis introduces

the element-oriented approach, in which the application reveals only the relevant functionality to the working structural element of the document. This element could either be a header, a paragraph, a table, a multimedia element, a text selection or a text annotation.

This thesis also highlights the significance of writing research, by encouraging writers to plan, organize and revise their documents. We integrated capabilities, in which the user could define the reading audience and the literary genre of the document, as well as to set overall goals. The writer also could visualize and manipulate the document's structure, as well as to set subgoals for different sections. Finally, the user can easily embed multimedia and web applications to the document, in an effort to enhance the communicating message of the document.

1.3 Thesis organization

In Chapter 2, there is an extensive review of the history of writing technology, from antiquity until today. At first, we present the advent of writing in early human societies as a social problem-solving tool, affecting social organization. Subsequently there is a review of writing technology evolution, starting from early writing materials and instruments, such as clay, papyrus, bamboo, parchment and paper, until the emergence of document replication of the printing press and the quality documents of the typewriter. The rising of computers shifts our focus from writing materials and instruments to writing applications and information storage. The evolution of computers from mainframes to minicomputers and later from personal computers to mobile devices fundamentally changed the writing process, featuring different writing approaches and interfaces. Writing approaches including the "What You See Is What You Get" (WYSIWYG) approach and "What You See Is What You Mean" (WYSIWYM) approach, along with the rising of touch interfaces and the Internet shaped today's realm of writing applications. At last, there is an analysis of the types of writing applications today ranging from text editors and word processors to desktop-publishing and web authoring.

The next chapter, focuses on learning theories and instruction, as well as in writing research. We investigate learning theories and their instructional applications deriving from behaviorism, social cognitive theory, cognitivism, constructivism and various contemporary approaches. This review also introduces conclusions and models from the procedural approach in writing production and instruction, also known as writing research. We use these findings to inform our application's design, in order to conform with learning and writing research.

Chapter 4 introduces our design methodology as well as a set of rules to inform the application's design. Our design approach relies on iterative design, featuring the definition of requirements and a detailed analysis of functionality in first two stages. The stage of design determines how developers should implement this functionality followed from the stages of prototyping and evaluation, before the final stage of implementation. We also involve in the process design rules deriving from the approaches of user-centered design, universal design and multimedia learning.

In Chapter 5, we present the design process described in the previous chapter. In requirements analysis there is a overall system description, user categories, contexts of use, as well as functional and non-functional requirements. In the stage of analysis, we introduce the method of hierarchical task analysis with plans of execution, while in the next stage of design we introduce th basic design decisions for our application. We also provide our prototypes and the results of our evaluation. Finally, there is description of the implementation process and an overview of the technologies we used. In the last chapter, we present conclusions and suggestions for future research.

Chapter 2

Writing technology

2.1 Writing systems and functions of writing

The evolutionary history of the modern humans involved many chronologically distant and diverse milestones. Early hominids evolved from Australopithecus, to Homo erectus, later to Neanderthal and more recently to Homo sapiens sapiens in a time span of approximately 3 million years. Different evolutionary adaptations altered morphological as well as cognitive traits of early hominids, enabling them to preform complex tasks. The adoption of the upright position liberated hands and facilitating the early use of tools. The gradual development of frontal lobes and areas in the middle cortex enabled humans to synthesize, think abstract as well as speak. Language enabled humans to create and synthesize abstractions of reality and later expressing them with sounds and gestures. In the upper paleolithic era, early humans incised dots and lines in stones, while later they started to paint on caves human and animal body parts. These cave paintings became more realistic in the neolithic era and gradually started to vanish, since nomadic communities of hunters and gatherers created permanent agricultural settlements. At this point in history, the new social organization required division of labor and strict hierarchy (Martin, 1995, pp. 1-8).

Agriculture and animal domestication supported these early settlements. People stored the excessive production of goods for future needs of the community and for trading purposes. This new paradigm of organization required explicit ways to measure products as well as to preform complex transactions. For instance in ancient Mesopotamia, they employed incised seals to mark symbols on small clay objects, in order to engage in financial activities (Martin, 1995, p. 8). This was the first attempt to use graphical notation for these transactions, which also was the first attempt to

write. *Writing* was the visual representation of spoken language that included marks, symbols, pictures and later letters (Meggs and Purvis, 2011, p. 6). Coulmas defined writing "as the procedure of recording language, using a set of visual or tactile marks" (Coulmas, 2002, p .1). This set of marks or symbols, define the concept of a *writing system*. Many civilizations introduced a great variety of writing systems from antiquity until today, with different symbol functionality. Symbols either represented concepts, either syllables or phonemes either a combination of them. Scholars divide writing systems in four main categories based on symbols functionality in correlation to spoken language. These categories are briefly presented below (Burnaby, 1997, p .5) (Coulmas, 2002, p .35).

1. *Logographic writing systems*, include symbols that each one of them represent a meaningful linguistic form. This form express an idea or a concept and not a sound from a spoken language. These symbols commonly called ideographs instead of the most accurate term logographs and they usually formed from early drawings of natural objects named pictograms. Logographic writing systems were the first to emerge in writing history, including the obsolete ancient Sumerian writing system and the Chinese script, which Chinese use until today.

2. *Syllabic writing systems*, contain symbols that represent the sound of the syllable of the spoken language. This syllable consists of a single phoneme or a group of phonemes and one of the most common example of a syllabic writing system is Japanese kana.

3. *Alphabetic writing systems*, include symbols that represent the sound of one or sometimes more phonemes. These writing systems are essentially alphabets, for instance Latin, Greek or Cyrillic. Apart from visual alphabets, another form of an alphabetic writing system is Braille. This tactile alphabet contains tangible dots, which visually impaired people commonly use.

4. *Mixed writing systems*, derive from combinations of logographic, syllabic and alphabetic writing systems that fuse in varying proportions. An instance of mixed system is Egyptian hieroglyphs, which was a fusion of logographs as well as phonetic symbols.

Writing emerged in a transitional phase of human history and served a variety of functions. The most obvious property of writing was the mnemonic function. Writing was memory supportive to many human activities either for provisional or permanent purposes. Until then people recalled every knowledge or information from memory, having the result that information was fading and altering in time. Writing transmitted accurate information for great time periods, leading to an early accumulation of knowledge.

The most important aspect that writing affected was social organization and more specific the economy, the legal system and communication. Early accountants now were able to keep track of numerous and complex financial and commercial transactions. This led to an organized financial system with diverse participants that could easily buy, sell and exchange goods. Others transformed the social rules from oral tradition to explicit legal systems with laws and penalties. The emergence of the law regulated social behavior, in order to reduce conflict between people in early growing settlements. Furthermore, writing extended the communication range between human communities, since many early documents could travel great distances. The oral communication paradigm of speaker, message and listener transformed to writer, text and reader. The message was for the first time a tangible object that a reader could read in different places and different times. This new kind of communication facilitated commerce and exchange of knowledge between distant groups.

Writing also affected the process of cultural transmission in many levels, ranging from language to literature as well as technical methodology. With the emergence of writing, words started to express stable meanings, since the question transformed from "what the speaker means" to the "what the text means". It was the first time that language was capable to separate from the context of situation and speaker's intentions. Moreover, writing transformed oral poetry and story telling to literature. Many literate as well as performance arts are impossible without writing, including poetry, novels and drama. In general, writing supported the transmission of potentially every culture, since written word could support learning. The traditional apprenticeship between a teacher and a student, a speaker and a listener, incorporated reading as supporting tool. This fusion enabled the listener to autonomously learn and act without the physical presence of the speaker (Coulmas, 1989, pp. 11-15).

To illustrate the significance of writing, Coulmas states that "Complex civilizations cannot exist without writing. The invention of writing can be seen as a kind of social problem solving, and any writing system as a common solution of a number of related problems" (Coulmas, 1989, p. 15). In the next section there is a historical review of the evolution of writing materials and instruments from antiquity until the invention of the typewriter. This review attempts to clarify the needs that caused these innovative adaptations and to unveil patterns that could inform our design.

2.2 Writing materials and instruments

2.2.1 Antiquity

The word Mesopotamia derives from ancient Greek and refers to "the land between rivers". This region of modern day Iraq, consists of vast plains and two regional rivers, Tigris and Euphrates. Ancient Sumerian scribes used damp clay in their early writings, which was one of the most common commodities of the region. At first they created clay tablets while the clay was still malleable and then used a pointing tool to trace pictograms on their surface. This process was problematic since scribes traced clay with difficulty and after clay was dry symbols usually distorted. Years later the scripts evolved from pictograms to simple lines and shapes resembling small wedges, resulting scribes to use a rectangular ended reed to trace their writings, instead of a pointing tool. This writing system was the first to emerge and called *cuneiform writing*. When the clay tablets filled with wedged symbols, then were left to dry in the sun or baked in a special oven called kiln, in order to increase their durability. As a result archaeologists discovered a great amount of these tablets thousands of years later, organized in libraries and archives, with various contents, including financial transactions, history, literacy, dictionaries and more. Due to their cumbersome nature and weight, clay tablets mainly used as a memorization aid and not as a communication medium, for several millennia to come (Martin, 1995, pp. 43-45).

At this point a brief definition of a writing material and a writing instrument could be useful. *Writing material* is the physical medium that provides surfaces, on which authors inscribe symbols. The inscribing object of the writing material is called, *writing instrument*. In most cases the writing material defines the writing instrument and vise versa, or even the form of the actual writing system. In the case of cuneiform writing, scribes traced symbols in damp clay, which was the writing material, by a rectangular reed, which was the writing instrument.

While Sumerian scribes were using clay to inscribe their writings, in Egypt used papyrus, even from the first dynasty dated back to 3rd millennium BCE. Cyperus papyrus is an aquatic flowering plant, found in abundance at Nile's delta and was a very useful commodity in everyday life of ancient Egypt. It served many purposes, ranging from boat sails and cloths to sedge baskets and papyrus scrolls. The most famous use though was papyrus derived paper, which was cultivated and produced in Egypt, resulting a booming industry of ancient times (Martin, 1995, p. 45).

Ancient Egyptians cut the pith of the papyrus stalk in strips, placed them closely in rows on a damp wooden tablet and then left the resulting sheet to

dry in the sun. Afterwards they processed these sheets, in order to smooth them, and finally they glued them together to form a papyrus scroll. The resulting writing material was flexible, solid, light and capable to retain ink. In order to inscribe on papyrus, Egyptian scribes used a cut reed dipped in ink, in order to create their hieroglyphic script. This writing culture was transmitted throughout the ancient world, from Phoenicia and Greece, to the Roman empire years later. Papyrus was one of the most important materials in the history of writing, due to its durability, consistency and mobility. It's worth mentioning that a great part of ancient literature from Greece, Rome and Egypt, existing today preserved in papyrus scrolls (Martin, 1995, p. 46).

A different perspective on writing materials and instruments introduced in ancient China. From 1500 BCE Chinese scribes were using bamboo or wooden strips to inscribe writings, with a stick made of wood, reed or bamboo, dipped in varnish and later ink. If documents were to small, writers inscribe Chinese characters vertically to a single strip. For larger documents they threaded together many of those single strips and created bamboo slips. Those bamboo slips were the dominant writing material in ancient China until the invention of paper in 105 CE. Silk was another less popular writing material used in China, besides bamboo and wood. At first scribes used varnish with a stick to imprint symbols, but later developed a technique based on seals. Wooden or bamboo sticks were the only writing instruments in China for many centuries, until the invention of the hair writing brush. It was created three centuries before the invention of paper and constituted one of the most influential writing instruments improvement, that changed Chinese writing for the years to come (Chi-Chen, 1930).

Meanwhile in the rest of the world, different civilizations used a variety of writing materials and instruments, dating from the 5th century BCE. In India scribes were using palm tree leaves for writing and birch bark a material, which was very popular also in Russia. Wooden tablets and bark was common in Egypt, Cyprus and the Roman empire. The critical flaw though of the wooden tablet and wood in general was the fact that it received signs poorly and scribes couldn't reuse it easily. In order to address this issue, scribes covered the surface of the wooden tablet with wax and used a stylus to inscribe signs very efficiently. When the wooden tablet was full of symbols they heated up the layer of the wax, the wax melted and was ready for reuse. This technique was widely employed even in modern times in the West mostly for provisional writing (Martin, 1995, p. 41).

2.2.2 Middle ages

Between the 1st and 4th centuries CE, parchment appeared as the new dominant writing material in the West. Middle Age Europe wanted to disengage from papyrus, due to insufficient production and the disruption of communication and trade with Egypt. Europeans needed to produce a new writing material, capable to retain equal or better qualities in comparison with papyrus. This new papyrus rival was parchment. It was invented in the ancient Greek city of Pergamon, in the 3rd century BCE, four hundred years before its mainstream adoption in Europe. The raw materials used to produce parchment were animal skins mainly originating from goats, sheep and calves. After a sophisticated skin processing method, they produced a high quality leather capable of retaining ink on both sides. Parchment was much more durable and reliable material than papyrus, slightly heavier though, but yet tolerant. Other great qualities of parchment were the ability to be written on both sides and washed for reuse. Of course in order to create a long manuscript with parchment, it could cost the lives of a flock of animals and that was its major drawback. But yet the use of writing back then was very limited, therefore parchment gratified the needs in writing materials at the time (Martin, 1995, p. 51).

Paper though was the most innovative technological achievement of ancient writing technology and another groundbreaking writing material at the time. Introduced officially in China in 105 CE by Tshai Lun, paper produced from pressed moist fibers derived mainly from pulp of wood, rags or grass. When the mixture of pulp dried, it created a thin flexible sheet, which called paper. The new writing material had great qualities, such as flexibility, mobility and low cost, resulting its spread gradually all over the world nearly fifteen hundred years after its invention (Tsien, 1985, pp. 1-3). Scribes used wooden sticks, hair brushes in China and pens later in the West, as writing instruments to inscribe on paper.

Paper arrived in Europe from the Arab world in the 7th century CE and after a five centuries monopoly, Europeans started to produce it in the 12th century CE. It became though really popular in the 15th century CE with the emergence of the printing press. Before the printing revolution, Europeans used papyrus and parchment as writing materials and even ban paper in some regions, because of its fragile nature (Tsien, 1985, p. 5). Until then people were creating handwritten documents in the form of a scroll. Alternatively they packed many of the manuscripts in the form of a codex, which was essentially a book made of paper or parchment with handwritten content (Martin, 1995, p. 59). It's worth mentioning that before the emergence of printing press in Europe, document replication was possible only with handwriting.

2.2.3 Printing

Back to the 6th century CE though, forms of printing already appeared in China, such as the use of seals and stone inscriptions rubbings. In particular Chinese seals had a rectangular shape with a flat base, inscribed with Chinese characters in reverse. Scribes dipped the seals to ink and then stamped on paper the carved image of the symbol. The majority of those seals were small and limited, in contrast with wooden seals inscribed with even a hundred signs. This replication technique was called woodblock printing, since it was capable to produce long and complex texts with a single stamp (Tsien, 1985, p. 6). The other significant approach to document reproduction was the stone inscriptions rubbings. In China was common to carve script on stone with a hard writing instrument. Since this procedure was extremely time and effort consuming and stone was a very cumbersome medium, they invented the technique of stone rubbings. With this procedure they created document duplications by placing another material e.g. paper to the ink covered surface of the carved stone and then they were rubbing it with a brush. Stone inscription rubbing was another instance of early printing in ancient China (Tsien, 1985, p. 7).

Chinese also introduced the next evolutionary stage of printing around 1045 CE. Chinese alchemist Pi Sheng invented the first form of movable type. This technique was an improvement of the woodblock printing and occurred, due to the inconvenience in producing woodblock seals. The case was that every duplicate symbol in a woodblock document should be separately craved again. Pi Sheng created individual components inscribed with a single Chinese character resembling to tiny seals, which could be placed sequentially, inked and printed exactly like a woodblock. With this technique the replicator could reuse every movable part and create new documents. The major drawback though of this procedure was the painful retrieving process of the hundreds of Chinese characters, resulting the failure of movable type to replace woodblock printing as the dominant printing technique in China. Later in 1403 CE, Koreans invented metallic movable type, but also did not became popular (Meggs and Purvis, 2011, p. 45). The Chinese seal, woodblock and movable type were the pioneering deviant perspectives of writing instruments, because of their capability to produce a single symbol or matrices of symbols. These instruments though were only for the reproduction of manuscripts and not for authoring original documents. Despite the poor adoption of movable type in Asia, this technique will be a great success story later in Europe with the introduction of the printing press.

Centuries later in Europe the public demand for manuscripts was growing by the emerging literate middle class. The replication techniques of the bookmaking industry were one thousand years old and the duplication of a

single book required the effort of a scribe for months (Meggs and Purvis, 2011, p. 68).

Johannes Gutenberg addressed this issue in 1450 CE with his invention of the printing press, following a simple procedure. At first Gutenberg created a matrix of movable type metallic parts representing a fraction of the document. Then this metallic text soaked to ink and pressed with sufficient power to a sheet of paper, in order to leave an impression (Meggs and Purvis, 2011, pp. 72-73). This process was repeating for all part of the document, until the creation of a book. Later in the industrial revolution many innovations were integrated in to Gutenberg's invention leading to the mechanization of printing, such as the steam-power or the electric printing press, which transformed printing to a high-speed factory operation (Meggs and Purvis, 2011, p. 151).

2.2.4 Modern times

Meanwhile paper was the dominant writing material of the 19th and 20th century CE in almost every corner of the world. Ranging from passports and identity cards to banknotes, government archives and newspapers, paper invaded everyday life for communication, organization, accounting, exchanging, entertaining and informing (Martin, 1995, p. 463). This development initiated the rapid increase of literal personnel that could engage to service based professions, establishing writing in those sectors mandatory. As a consequence, the demand for improvements to writing materials and instruments rose, such as the paper quality as well as the usability of pens. Pens transformed from the reed pen of the antiquity to the goose quill pen, then to metal pen nib and finally to the fountain pen. These improvements were very significant but they didn't transform handwriting, in order to deliver same quality handwritten documents in comparison to the printed ones. The typewriter, one of the most astonishing milestones to writing instruments, eventually bridged this gap between printing and handwriting (Martin, 1995, p. 464).

Typewriter was the first personal writing instrument, which consisted of mechanical parts, including a keyboard, typebars and a cylinder. This device used the movable type concept to create ink impressions to a writing material, which was usually paper. When the typist was pressing a key, the type at the end of the bar stroked the paper on the cylinder, leaving an ink impression of letter or punctuation. The first major drawback of typewriters appeared, when the typist stroked the keys rapidly. Then neighboring typebars tented to jam, leading to often stops of the typing process. Typewriter manufactures addressed this issue, by changing the keyboard layout alignment from alphabetical to QUERTY (the first six letter of the top row),

in order to reduce the jamming possibility of two typebars (Martin, 1995, pp. 464-465). Another drawback of the first typewriter was the fact that the typist couldn't see the produced text, leading people to call this machine as the "blind writer" (Condoor, 2004).

The typewriter market grew steadily from the first commercial typewriter introduction in 1874, until the breakthrough of 1881, when the Young Women's Christian Association in the United States started offering typing courses and established typing as a great career opportunity for young women. The introduction of the Underwood No.5 typewriter solved the "blind writer" issue in 1985 and in 1920 the need for silent performance and portability created the electric typewriter. The IBM Selectric model solved the typebars jamming issue by replacing the typebars with a moving single sphere filled with reverse letter impressions, elevating typing speeds. The final improvement of the typewriter was an electronic version, which allowed the user to input, revise and afterwards print text through a small monitor. This was the last evolutionary stage of the typewriter and the first of the digital era starting with the emergence of the computer (Condoor, 2004). In the next section we will examine the evolution of writing in personal computers, various applications, such as text editors, word processors, desktop publishing and web authoring applications, as well as different writing approaches. Finally we will discuss the implications in writing derived from the World Wide Web and modern research issues, such as the Semantic web.

2.3 Information storage and writing applications

In the previous section, we examined the technological evolution of writing from antiquity until the rising of the electronic typewriter. So far the distinction between writing material and writing instrument was clear even in the case of the typewriter, which was the most complex personal writing instrument we investigated. With the emerging computer writing technology this clear distinction blurred, resulting an imperative need to find corresponding concepts to writing material and instrument, that could fit to the information age. These new concepts are essential to continue this evaluation of writing technology and identify important information, capable to inform our design. In order to search for a relevant concept of the writing material it's useful to inquire its fundamental properties. The writing material definition states, that is the physical medium providing a surface, on which authors could inscribe symbols. Apart from the inscription ability, this surface can also display and store written information, abilities which we have considered obvious so far. In the computer world though, information display and storage are two discrete processes, taking place in different

structural entities. For the purpose of this essay, we consider information storage as a research variable, in order to evaluate the writing procedure in the computer age. More specifically the term *information storage* is defined, as the type and location of the recording media used to store and retrieve information.

The other significant concept of writing technology, that has been used throughout this chapter, is the writing instrument. Essentially computers are writing instruments, since a user could create a digital document using them. Starting from the 1950's, computing devices had demonstrated a remarkable evolution in hardware, ranging from the 1960's mainframe to the modern smartphone. Similarly an corresponding development also occurred in software development tools and applications. Therefore from this point forward another indicator of writing technology to investigate are writing applications. By the term *writing applications* we define the computer programs or tools, used for document authoring purposes. To sum up, in this section we will examine the influence of the information age to the writing process, regarding the emergence of the computer. Also here the computer will not be examined as computational machine, but rather as a writing instrument. The following historical review will cover a period from 1950's until today, and will focus on information storage and writing applications.

2.3.1 Early computers, mainframes and minicomputers

The starting point of this review is the advent of commercial computing. After the development of ENIAC in the University of Pennsylvania, Eckert and Mauchly tried to transform an expensive scientific instrument, such as the 1950's computer, to a consumer product. Along with IBM corporation, which was their major rival at the time, they started shipping computers in early 1950's for commercial and military use (Ceruzzi, 2003, p. 14). First generation computers were enormous, with vacuum tube processors and created to encounter specific computational problems. These first computer manufactures developed this new technology, mainly to replace the "unit record equipment", which was very popular in business practice at the time. The "unit record" was an machinery installation of various punched card machines, used for business organization. A single punched card represented an autonomous encoded entity, e.g. a sales transaction, used for multiple purposes ranging from sorting and counting to tabulating and printing some desired sets of columns. The process involved, many decks of cards running though different machines producing the required result (Ceruzzi, 2003, pp. 15-16).

By the end of 1960's the new transistor mainframes used many different devices to store information sequentially, such as *punched cards* and *magnetic*

tapes. The concept of sequential information storage required data access in a predetermined ordered sequence. In contrast, IBM in 1957 had already invented a spinning disk, which could store and retrieve large amounts of data by using the random access technique. With *disk drive storage*, data could be accessed randomly and not sequentially, resulting random access to be a crucial milestone in computing history (Ceruzzi, 2003, p. 69). Until the mainstream adoption of random access techniques, program authors encoded computer programs on punched cards, creating the first instance of a digital document. At first, they used a keypunch machine to create a deck of punched cards, which every one of them corresponded to a single computer instruction. Afterwards they entered the cards sequentially to an intermediate computer, to transfer the program from cards to a reel of tape. Later, programmers mounted this reel of the tape to the tape drive of the mainframe for execution. They inserted data with the same procedure, while the mainframe sent the data process results also to tape. Then programmers took the results tape and mounted it again to the intermediate computer. A chain printer printed the results on fan folded paper for programmers to review and evaluate, in order to revise the program or not (Ceruzzi, 2003, pp. 73-74). This paradigm of editing was called *non interactive computerized editing*, since the author couldn't directly interact with the produced text, in this case the computer program (Meyrowitz and Van Dam, 1982a, p. 333).

Editing transformed from non interactive to interactive with the emergence of minicomputers and teletypes. Minicomputers expanded the application spectrum of computers by integrating various advancements of technology from electronics, solid-state physics and computer architecture. These new computers challenged directly the mainframe dominance by introducing the concept of an interactive computing device (Ceruzzi, 2003, p. 124). Interaction was the product of teletype, a device similar to a typewriter capable of printing computer output in a roll of paper. Furthermore, teletypes were sending encoded user input either in a form of electric signals or on punched *paper tape*, an approach similar to punched cards. Paper tape though was an advancement on data storage technology in comparison to punched cards, since computers read this information directly without human involvement (Ceruzzi, 2003, p. 133). Teletype technology introduced the first approach of an interactive editing application, which was the *line editor*. Users imported and stored information in a series of lines, in order to traverse and edit documents in line basis, generating user output at the teletype (Meyrowitz and Van Dam, 1982a, p. 344).

So far mainframes and minicomputers were executing sequentially a batch of jobs meeting the needs of a single user. The emerging technology of timesharing allowed computers to divide processing time and power to many

users making the computer accessible to broader audience (Ceruzzi, 2003, p. 154). Every user of a time-sharing system used a device called terminal, in order to interact with the main computer. At the beginning, early terminals integrated a keyboard and a printing mechanism, similar to teletypes. Later the improved glass teletypes integrated the video screen a groundbreaking technology at the time, but they offered limited viewing area and data entry capability. Smart terminals also employed a video screen as well as allowed the user to view and edit a full screen of text, improving the usability and functionality of the glass teletype (Ceruzzi, 2003, p. 250). The writing application of these terminals was the *full screen editor*, a great text editing innovation of its era. Full screen editors allowed users to view and edit the document in full screen with no line restrictions. In line editors, the user had to enter commands and text arguments to manipulate text, in contrast with screen editors that utilized a moving cursor to access any point of the document, in order to add, delete and modify text (Meyrowitz and Van Dam, 1982a, p. 335). The advent of the screen editor also marked the arrival of the *user interface*. Users interacted with the editor through a set of tools and techniques, which defined the concept of the user interface (Meyrowitz and Van Dam, 1982a, p. 323).

So far text editing referred essentially to programming and didn't concern ordinary office workers. Viatron and Wang Laboratories were the first companies, that tried to bridge the gap between typewriters and computers, in an early attempt to create the concept of office automation. Typists at the time were afraid by an accidental keystoke in a typewriter, that could waste a day's work. Wang's engineers addressed this issue by developing a user friendly text editor with convenient menus specially designed for computer terminals, marking the dawn of word processing (Ceruzzi, 2003, pp. 252-257). By the mid-1960's, the invention of the integrated circuit triggered the second generation of minicomputers. Video screens and graphical user interfaces were the emerging new technologies, changing computing for the coming years.

The most influential example of this era was the Xerox Alto, a newcomer second generation minicomputer. Xerox engineers developed the Alto in 1973 at Xerox-PARC, the famous research center of Xerox located in Palo Alto. The company's core business until then was the development of copiers, but advancements in computing technology shifted company's focus, in an effort to follow changes in the industry (Ceruzzi, 2003, p. 258). Xerox Alto included the computer mouse as an input and interaction device, since it was more usable in comparison to competing devices, such as the light pen or the joystick (Ceruzzi, 2003, p. 260). Alto's design also integrated a video screen, which displayed innovative visual elements, such as windows, icons and pull-down menus. This interaction approach of Windows, Icons, Menus

and Pointer (WIMP) was the first graphics user interface (GUI). These interaction advancements made possible with the introduction of the bit-mapped screen. This innovation displayed visual elements as a raster of pixels (picture elements), allowing the users to manipulate, scale and move graphical elements. Also the user could scale letters and fuse graphics with text on the screen, which until then was not possible in earlier alphanumeric CRT terminals (Ceruzzi, 2003, p. 261) - (Meyrowitz and Van Dam, 1982b, p. 370). Xerox Bravo was an interactive editor and formatter, that delivered all the above using the mouse as a pointing device. The author could visualize and modify for the first time the document's final layout, introducing the "What You See Is What You Get" (WYSIWYG) editing approach. This approach will dominate word processing systems for the coming years and affect until today the majority of writing applications. Xerox Alto despite the groundbreaking innovations was essentially a commercial failure, mainly due to elevated cost. Nevertheless these features of Xerox Alto will become mainstream some years later with the introduction of personal computers (Ceruzzi, 2003, p. 261) - (Meyrowitz and Van Dam, 1982a, p. 336).

2.3.2 Personal computers

As the density of semiconductor electronics increased, integrated circuits became significantly smaller and cheaper. These advancements resulted a great impact in the calculator industry. Until then office workers used mechanical calculators, which were very complex and very expensive. Wang Laboratories made the shift to electronic calculators first by introducing Wang Loci, a calculator featuring advanced functionality and reduced cost in comparison with its mechanical rivals. At first mechanical and then electronic calculators offered a directed functionality to arithmetic calculation, since accountants were the majority of their users. By 1970 electronic calculators were impressively smaller, cheaper and affordable to general pubic, but yet their focus to arithmetic computation was still very distant from a general purpose computer (Ceruzzi, 2003, p. 211-216).

The step forward to general purpose personal computing didn't derive from the calculator industry, but rather from the invention of the microprocessor. Microprocessors included a set of integrated circuits implementing the basic architecture of a general purpose computer. Moreover, they used software not hardware to diverse their functionality, in order to confront specific problems (Ceruzzi, 2003, pp. 217-218). In 1974 Intel announced the 8080 microprocessor and a year later the prototype of Altair was cover at Popular Electronics magazine. Altair was the first general purpose personal computer or microcomputer, with an orientation mainly to hobbyists rather than professionals. The front panel of Altair was full of switches, which users

employed to import data to registers and small lights displaying the binary representation of a command (Ceruzzi, 2003, pp. 226-227). But when a user switched off Altair it lost all of its data, which means that users couldn't store, reuse or share their work. A group of hobbyists provided a solution to this storage issue. They designed an interface, which could transform digital data to specific audio tones and then record them on cheap audio cassettes. This information storing technique was sequential, similar to mainframe magnetic tapes, very slow and was an obstacle to personal computer penetration to a broader audience (Ceruzzi, 2003, p. 231). IBM had invented the successor of the audio cassette already back from 1971, but it appeared in the personal computer market with a clone of Altair called IMSAI, featuring 8-inch floppy drives, a video screen and the Control Program/Monitor (CP/M) operating system (Ceruzzi, 2003, p. 240). The *floppy disk* was a random access storage medium, very fast compared to audio cassettes and an ideal information storage for the majority of early personal computers (Ceruzzi, 2003, p. 236).

In 1976 Michael Shrayer announced the Electric Pencil, the first word processor of the microcomputer era. News were spreading rapidly in computer hobbyists community and Electric Pencil became very popular in a short period of time. The program employed text as a continuous stream of characters, allowing the user to modify, insert or delete it in any part of the document. Furthermore, hyphenation was unnecessary since Electric Pencil formatted the text automatically in lines, materializing an early version of word wrap. Users could also view the document in the video screen and access any part of it by scrolling the cursor reverse and forward. The commands for editing and formatting were key-bindings of the Control key (^) and another key that corresponded in a specific command. Another functionality of Electric Pencil was text selection. Users could delineate blocks of text using two backslashes (\), one at the beginning and one at the end of the text selection. The operator then could easily copy, move or delete the selected text area. Users could also denote a page header and page numbering using the dollar symbol ($) in each page. Despite its useful functionality, Electric Pencil's major drawback was the absence of a standardized operating system capable to handle the communication between the program and the computer. Until then Michael Shrayer provided a different version for every hardware configuration, leading to Electric Pencil's 78 different versions. The penetration of CP/M operating system to the majority of early personal computers simplified this issue, making the porting process to a different hardware configuration easier. Shrayer announced the second version of Electric Pencil in 1978 only in eight versions, introducing features, such as paragraph indentation and text centering. Other additions of the second version were also character formatting features, including underline and boldface (Bergin, 2006a, p. 34).

The second wave of personal computing started in 1977 with the introduction of TRS-80 Model 1 and Commodore PET. They both featured keyboards, monitors and cassette players for economic storage. A different version of the BASIC programming language was available in a read-only memory chip along with a minimum operating system, ensuring the operation of the devices (Ceruzzi, 2003, pp. 263-264). Later that year Apple Computers introduced the Apple II with similar features and functionality, but with an optional addition of a 5 1/4 floppy disk drive. Floppy disk storage was ideal to the personal computer concept and as a result was the creation of an early commercial software market and channel of software distribution. Apple's II famous application was VisiCalc, an early spreadsheet application capable to compete expensive mainframe professional programs (Ceruzzi, 2003, pp. 266-268). In 1979 John Draper introduced a word processing system for Apple II called EasyWriter, featuring an early WYSIWYG ability to display text on the video screen identical to the printed page (Bergin, 2006a, p. 36). EasyWriter in combination with VisiCalc created booming sales by 1981 and established Apple II as a complete solution for businesses as well as ordinary users.

In order to confront Apple's success, IBM announced in August the legendary IBM Personal Computer. The IBM PC featured a keyboard, floppy disk drives and a monitor, that could display 25 lines of 80 characters, a notable improvement in comparison to Apple II especially for office applications (Ceruzzi, 2003, p. 268). IBM also shipped EasyWriter as the word processing system, but poor reviews of computer magazines at the time reduced the consumer demand. EasyWriter's commercial failure in the IBM PC software ecosystem did not had a negative impact for IBM PC, because of DisplayWriter. This application was IBM's in house word processing system alternative to EasyWriter, which IBM also shipped with every PC (Bergin, 2006a, p. 37). IBM provided also applications for accounting, CP/M and PC-DOS operating systems, games and a version of VisiCalc. Lotus Development introduced Lotus 1-2-3 in 1982. Lotus 1-2-3 was the critical spreadsheet application of the PC ecosystem, which assisted IBM to gradually overtook Apple's market share and establish IBM as the dominant vendor in personal computers industry (Ceruzzi, 2003, pp. 268-269). In 1983 IBM introduced the successor of the original IBM PC with the code name IBM PC XT. The most groundbreaking feature of XT was the 10 Megabyte *hard disk drive* for information storage (Ceruzzi, 2003, p. 298). Due to IBM's commercial success many companies started to produce IBM-compatible personal computers, which were essentially legal clones of the IBM PC. This emerging IBM-derived computer ecosystem led to a booming software and hardware commercial market, that challenged the traditional computing cultures of mainframes and minicomputers (Ceruzzi, 2003, pp. 277-279).

The most prominent word processing system of this era though was WordStar. MicroPro announced WordStar back in 1979, in order to provide a word processor for the CP/M operating system. WordStar included a usable Control-key user interface, designed to improve typing and editing speeds. The left hand of the user controlled the movement of the cursor, while the right hand controlled other functionality. WordStar was a WYSIWYG word processor in terms of the correspondence between the video screen document typesetting and the printed output, a feature that wasn't common in other word processing applications at the time. WordStar also integrated automatic word wrap and hyphenation, on screen page breaks and in-line user assistance (Bergin, 2006a, p. 39). Consumer demand for WordStar was excessive by 1984 transforming MicroPro to the largest software company worldwide with millions of dollars in sales (Bergin, 2006a, p. 40). WordStar was dominant in the CP/M word processing market. Micropro introduced also PC-DOS versions for the IBM PC with numerous improvements, such as spell checking, an un-erase feature and support for laser printers, but they didn't saw massive adoption (Bergin, 2006a, pp. 41-42). The emergence and wide acceptance of IBM PC and its clones transformed the computer industry, resulting the eventual decline of WordStar and the gradual market share takeover by new competitors, such as Microsoft Word and WordPerfect (Bergin, 2006a, p. 43).

The gradual transition from early microcomputers using the CP/M operating system to the emerging standard of the IBM PC, based on MicroSoft Disk Operating System (MS-DOS), appointed WordPerfect as the new major rival to WordStar supremacy. Apart from WordPerfect, various software vendors released over 200 word processing applications from 1982 to 1983 for the IBM PC, establishing the word processor market as highly competitive. By the introduction of WordPerfect 2.215 for the IBM PC, WordStar already held the 75 percent of the word processing market. WordPerfect gradually started to grab market share and by 1984 had already passed the million dollar sales mark (Bergin, 2006b, p. 50). WordPerfect's 3.0 printing support extended to over 200 printers, which was more than any other word processor at the time, a key factor for adoption of WordPerfect (Bergin, 2006b, p. 51). With the 4.1 release in 1985 WordPerfect incorporated a plethora of new features, including improved speller and thesaurus, automatic indexing and tables of contents and footnotes (Bergin, 2006b, p. 52).

At the time WordPerfect was starting to penetrate the word processing market, Apple was introducing Lisa in 1983. Lisa was the first personal computer with graphical user interface inspired by Xerox-PARC research a decade earlier. Xerox had already revealed Xerox Star the successor of Alto back in 1981 with a graphical user interface, but was very expensive and didn't become mainstream. Customers couldn't also afford the price tag

of Lisa, leading to another commercial failure this time for Apple (Ceruzzi, 2003, p. 273). All that changed with the Macintosh announcement in 1984. Apple Macintosh inherited many of Lisa's GUI features, introduced a mouse as a GUI pointing device, a new 3 1/2 inch floppy drive and (Ceruzzi, 2003, p. 275) MacWrite, the first true WYSIWYG word processor with a graphical user interface. Xerox Bravo was the inspiration of MacWrite's design, resulting WYSIWYG features, such as drop down menus, boldface, italics and underlined letters of various styles and sizes. Apple included MacWrite for free in Macintosh, but by the end of 1990 the Macintosh edition of Microsoft Word was the top selling word processing package in Macintosh platform and Microsoft was the number one publisher of Macintosh software (Bergin, 2006b, pp. 58-59). The great contribution though of Macintosh was the commercial introduction of graphical user interface to general public and established this type of interaction in personal computing industry until today (Ceruzzi, 2003, p. 275).

Until then, Macintosh used a raster monochrome video screen displaying graphics and fonts as an rectangular array of pixels (Meggs and Purvis, 2011, p. 531). Other raster output devices of the time, included dot-matrix, inject and later laser printers, that used the same pixel oriented concept in printing. The visual correspondence between the video screen and the printed output was the main concern of word processing systems, which were competing each other in order to deliver the most accurate result. A solution to this issue derived from Adobe Systems in 1982 and their PostScript page description language (Adobe Systems Inc., 1999, p. 11). PostScript described the final layout of a page, including the appearance of text, images as well as other graphical elements, regardless which output device displayed or printed the page (Adobe Systems Inc., 1999, p. 1). The fonts of PostScript language were graphical descriptions shaping the letter outline based on Bézier curves, instead of the arrays of pixels. This concept integrated also in graphical elements, leading to the dawn of electronic typography. Aldus developed PageMaker the first page-layout application in 1984 and introduced the first Macintosh version in 1985. This new application could integrate images, borders and headlines, change the type and size of fonts and manipulate text columns. PageMaker allowed the user to manipulate the layout of a page in the computer desktop similarly with the printing techniques of the time. This new method created the concept of *desktop publishing* and changed the publishing industry for the coming years (Meggs and Purvis, 2011, p. 531).

Meanwhile the dominant operating system for the IBM PC ecosystem was MS-DOS, which employed a command line interface instead of a graphical one. Already from 1982, many software vendors attempted to develop a GUI for the IBM PC, such as VisiOn an interface from VisiCalc develop-

ers, Top View from IBM and Graphics Environment Manager (GEM) from Digital Research, but they never saw commercial acceptance. Microsoft at the time was developing Interface Manager, a similar attempt to deliver a PC GUI, but sales and quality were also poor leaving MS-DOS and its command line interface the only feasible option for the IBM PC (Bergin, 2006b, p. 53) (Ceruzzi, 2003, p. 276). By 1987 WordPerfect was dominating the word processing market with a 30 percent share followed by the 16 percent of WordStar. IBM's DisplayWriter was holding a 13 percent and early versions of Microsoft Word gathered a 11 percent (Bergin, 2006b, p. 53). The next two years the market share of WordPerfect exploded to 60 percent generating $281 million dollars in revenue and by 1990 to 70 percent with $452 million (Bergin, 2006b, p. 54). Already from 1983, early MS-DOS versions of Microsoft Word were competing with WordPerfect, but never really challenged its market share (Bergin, 2006b, p. 58). The great market shift in graphical operating systems as well as in word processors started with the introduction of Microsoft Windows for the IBM PC. Microsoft announced Windows 1.0 in 1985 and Windows 2.0 in 1987. The next two years Microsoft Windows sold 2 million copies and became the most selling product of Microsoft. By the end of 1990, Microsoft released the breakthrough version of Windows 3.0 and transformed the computing industry for the years to come.

The decline of WordPerfect started when developers failed to deliver a version for the Windows 2.0 platform, due to busy schedules of WordPerfect 5.1 development cycle. The massive Windows 3.0 adoption initiated rapid changes in the operating systems market, leaving WordPerfect developers unprepared. They released WordPerfect 5.1 for Windows 39 months after the Windows 3.0 release, despite the fact that WordPerfect was still dominant to that day (Bergin, 2006b, p. 56). The new rival of WordPerfect was the Windows edition of Microsoft Word, which Microsoft integrated smoothly into the new Windows platform. Microsoft developers released Microsoft Word 1.0 in 1989 and exploited all the new capabilities of the Windows environment, in contrast to other competitors. Microsoft Word was an WYSIWYG word processing package with an interface and features, similar to MacWrite and the earlier Word edition for Macintosh. Later versions of Microsoft Word and Windows dominated the market and established Microsoft Word as the market leader and word processing standard of the IBM ecosystem (Bergin, 2006b, p. 59). The success of Microsoft Word didn't rely exclusively in its word processing capabilities and its graphical user interface, but rather to the productivity applications bundle with Excel spreadsheet and the PowerPoint presentation tool. In 1989, Microsoft announced Microsoft Office for Macintosh, a package including Word, Excel and PowerPoint and saw remarkable acceptance. Microsoft Office for Windows in 1990 was the following success, which by 1994 was holding the 90

percent of the market establishing Microsoft as the leading software company world wide (Bergin, 2006b, p. 60).

2.3.3 Workstations and the world wide web

IBM introduced the IBM PC back in 1981, creating a new computing paradigm. At the same period the parallel computing culture of workstations was rising. Workstations were very similar to personal computers in terms of architecture and design. They also integrated an inexpensive microprocessor and were much more affordable than minicomputers and mainframes at the time. The major difference between workstations and personal computers was the extended networking capability, for transferring data and for sharing devices, such as printers and plotters. The other different feature of workstations was the adoption of Unix as their operating system (Ceruzzi, 2003, p. 281). Bell Laboratories were at the time a research facility of AT&T. They developed the Unix operating system in the C programming language, instead of a processor specific assembly language. As a result Unix could run in any platform with a C compiler. In contrast to the majority of software vendors, AT&T supplied the source code for a nominal fee, allowing the user to modify and adapt Unix's functionality to its needs. Universities all over the United States exploited this quality of Unix, in order to modify and enhance its capabilities (Ceruzzi, 2003, pp. 282-283).

The penetration of Unix to academic institutions initiated a wave in writing applications development mainly for text editors, such as Vi and Emacs. Computer programmers used these text editors extensively for programming purposes. Yet academia required writing applications capable of typesetting as well as providing scientific notation for publishing books and scientific papers. One of the most popular typesetting systems with extensive use in academic community was LaTeX, due to mathematical and computer science notation. Leslie Lamport introduced Latex back in 1985, implementing a special version of Donalt Knuth's typesetting system called Tex, capable of executing LaTeX commands (Lamport, 1994, p. 5). These commands indicated the logical structure of the document by providing semantics in logical entities, including figures, tables, chapters, sections and paragraphs. As a result, Latex could create automatically table of contents, list of figures and tables as well as bibliographic references. In addition, LaTeX provided various document layout templates for books, articles, letters, reports and others (Lamport, 1994). Contrary to the visual design of the WYSIWYG approach, LaTeX was essentially a markup and typesetting language, that created a logical design of the document (Lamport, 1994, p. 7). The LaTeX approach allowed the author to focus on writing and not to worry about formating issues, since formating and typesetting were automatic (Lamport,

1994, p. 8). This paradigm is an instance of the "What You See Is What You Mean" (WYSIWYM) approach, a different writing perspective in comparison to WYSIWYG.

Back in early 1982, Vinod Khosla founded SUN Microsystems and at the same year announced the release of SUN's first workstation (Ceruzzi, 2003, pp. 281-282). Sun workstations integrated the Brerkley Unix as their operating system, a choice that generated booming sales and established SUN as a major competitor in the workstations market (Ceruzzi, 2003, p. 285). In 1987, SUN Microsystems introduced a RISC architecture processor called SPARC (Scalable Processor Architecture). SUN's engineers licenced the hardware design of SPARC to other companies in an effort to create a competing standard to personal computers and improve SUN's influence on the industy (Ceruzzi, 2003, pp. 289-290).

Ethernet was the networking technology of workstations, which was yet another groundbreaking invention of the Xerox Palo Alto Research Center in the 1970's. Ethernet transformed office environments by interconnecting workstations and later PCs into Local-Area Networks (LANs) (Ceruzzi, 2003, p. 291). The new concept of local-area networks allowed users to transfer files, share networks resources, exchange electronic messages, receive administration services and also connect to the Internet. As a descendant of Arpanet, Internet was a packet switching network that interconnected other networks worldwide using the TCP/IP protocol (Ceruzzi, 2003, p. 295). Yet the most non anticipated product of the Internet that took place in the most unexpected organization was the World Wide Web (WWW). Tim Berners-Lee invented the world wide web in the high-energy physics laboratory of CERN in 1990. Its fundamental notion was the organization of information in a hypertext form (Ceruzzi, 2003, p. 301). Robert Cailliau and Tim Berners-Lee in their proposal for world wide web stated that "Hypertext is a way to link and access information of various kinds as a web of nodes in which the user can browse at will" (Berners-Lee and Cailliau, 1990). In order to define the location of a hypertext they introduced the Universal Resource Identifier (URI). Using the URI the author of the hypertext could indicate an unique address of a document on the web. The communication layer between web nodes was possible with the HyperText Transfer Protocol (HTTP). HTTP's funtionality was to create requests and responses between the nodes of the web. Finally the last building block of the web was the HyperText Markup Language (HTML), a markup language that defined the structure and content of hypertext documents (Ceruzzi, 2003, p. 302).

An international standard by the name of Standard Generalized Markup Language (SGML) was the foundation of HTML. The basic purpose of SGML was the definition of a document's structure via markup notation, so a variety of devices could present the same content with different ways,

ranging from video screens to printers. SGML's goal was the division of document's structure and presentation in two distinct entities. Structure consisted of content and markup elements defining paragraphs, headings and lists. Presentation in the other hand, referred to the different ways a device could present this structure (Pfaffenberger et al., 2004, p. 5). HTML inherited this basic concept of SGML and with version 1.0 started to introduce basic HTML elements for structure. Mosaic 1.0 developers implemented the first HTML 1.0 specification, in an effort to create the first web browser. Tim Berners-Lee founded the World Wide Web Consortium (W3C) and released the second version of HTML, in order to standardize and evolve the HTML specification. HTML 2.0 allowed the user to include images in web pages as well as to submit information to a web page by using the HTML forms. The first real attempt though to separate structure from presentation was the 3.2 specification of HTML, that introduced the concept of Cascading Style Sheets (CSS). HTML authors used style sheets, to alter only the visual representation of a document without affecting the original HTML code. Web browsers at the time didn't support CSS efficiently enough, resulting a broader use of CSS not until the emergence of HTML 4.01 and CSS 2.0. Apart from HTML, another paradigm of markup language was the eXtensible Markup Language (XML). XML intended to meet the needs of the publishing industry by creating a flexible markup language that could fulfill any markup requirement. As a result, a new version of HTML derived from the XML concept called EXtensible HyperText Markup Language (XHTML), leading to a more well defined and structured markup language with strict syntax regarding to HTML elements and attributes (Pfaffenberger et al., 2004, pp. 7-8).

The emergence of HTML as the universal world wide web markup language triggered the development of tools that could author web pages based on the HTML specification. Text-oriented and WYSIWYG HTML editors were the main two categories of these applications. Simple text-oriented HTML authoring required from the user to code HTML, in order to create a web page without any assistance or code syntax highlighting, such as Windows Notepad or Linux's Vi. In contrast, smart text editors including Vim and Emacs, could provide syntax highlighting for various languages as well as auto-indenting for keeping documents structured. The last category though of these editors were the HTML-specific text editors providing additional features assistive to the author, in order to create HTML documents, including tools for entering tags or table creation wizards (Pfaffenberger et al., 2004, pp. 555-557). Finally, WYSIWYG HTML editors allowed the user to create a visual representation of document's final layout. Authors could create a HTML document using only with visual tools and not by coding. Microsoft Frontpage, NetObject fusion and especially Macromedia Dreamweaver revolutionized web publishing in the early years, implementing the WYSIWYG

approach in web authoring (Pfaffenberger et al., 2004, pp. 558-561). The gradual mainstream adoption of world wide web and Internet technologies altered also the process of publishing, by introducing the *remote storage* abstraction of web documents. Until then, authors stored their documents in local storage e.g. floppy or hard disks, sharing them only in a local-area network. In contrast, a web author could publish documents in a remote web server, distributing them on demand in a global audience with a single mouse click. With the emergence of world wide web, formating and typesetting of a web document were gradually becoming irrelevant. The reason was that computer screens displayed now the documents, instead of paper prints. The computer screen along with the world wide web gradually displayed paper from its dominant position for the first time in modern history.

In the meantime Unix was still very popular in academic institutions, but it was only available for workstations and not for personal computers. In 1991 Linus Torvalds, a twenty years old computer science student from Finland, started to code his own version of Unix for his IBM compatible personal computer (Ceruzzi, 2003, p. 333). This new operating system called Linux, spread throughout the Internet and became popular to computer enthusiasts all over the world. The code base of the project was open and many developers were started to contribute code or fixing bugs (Ceruzzi, 2003, p. 335). This open approach to software development became very popular later with Free Software Foundation[1] and the Open Source Initiative[2]. These institutions produced software and software licenses, which allowed users to execute, investigate, adapt and redistribute the source code of a computer program (Stallman, 2002, p. 3). This new perspective to software challenged Microsoft's supremacy to personal computer market for the years to come.

Apart from Linux, another emerging rival of Microsoft was SUN Microsystems and their new programming language called Java. In 1995 SUN Microsystems introduced Java, a new programming language capable to adapt to the new world wide web computing environment. Java saw massive adoption, due to its promise to be platform interdependent, meaning that a Java developer could run the same code to different operating systems and platforms. At the same time, Java offered animation and interactivity to web applications feeding a growing inkling from the press that this could reduce Microsoft's influence in personal computers market. Many thought that Microsoft's dominance was at stake not only because Java applications were platform independent, but also because developers could deliver them through the Internet as services. Despite the success of Java, history dis-

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

[2] `http://opensource.org/`

prove these predictions (Ceruzzi, 2003, pp. 324-325).

The acquisition of the German company StarDivision and their office suite in 1999, expanded the growing influence of SUN Microsystems in the computing industry. StarOffice was a full featured office suite with a WYSIWYG word processor, spreadsheet, database and presentation applications, intended to become an alternative office suite to the dominant Microsoft Office. One of the major differences between StarOffice and Microsoft Office was their file formats. Until then, Microsoft Office was using binary file formats to describe office documents, in contrast to StarOffice that was using XML-based formats. The marketing strategy of SUN Microsystems allowed the free purchase of StarOffice for personal and educational use. In contrast, SUN charged the same product seventy dollars for commercial use aiming corporate clients. The next step for SUN was the release of StarOffice source code with an open source license initiating the OpenOffice.org project. OpenOffice community released the first version of OpenOffice back in 2002. After its release, the Organization for the Advancement of Structured Information Standards (OASIS) made an effort to create an open standard based on the OpenOffice file format XML specification. The product of this attempt was the OASIS Open Document Format for Office Applications or shortly the Open Document Format (ODF). International Organization for Standardization (ISO) approved ODF as an international ISO standard in 2006, allowing its use from many other office applications. OpenOffice managed to grab some market share, but never challenged Microsoft Office in office productivity market (Ditch, 2007, pp. 11-12).

Until the emergence of ODF and XML-based formats, office applications used proprietary binary formats to encode human information to a machine readable binary form. The result was that corporations obligated users to use their applications, since were the only capable of reading the specific file format (Ditch, 2007, p. 4). This eventually led to a vendor lock-in of the market, generating industry standards and monopolies, such as Microsoft Office (Ditch, 2007, p. 2). The general shift towards open XML-based file formats and especially the ISO standardization of ODF, forced Microsoft to drop binary office formats and to introduce instead the XML Reference Schemas in 2003 and Office Open XML (OOXML) file formats in 2005. ISO standardized OOXML in 2007 and Microsoft integrated it in Microsoft Office 2007 as a prime format, describing word processor, spreadsheet and presentation information (Ditch, 2007, p. 13).

Back in 1993, Adobe used its in-house PostScript page description language to develop a non XML-based file format. Adobe's proprietary Portable Document Format (PDF) represents the exact visual layout of the document and gradually became an industry standard for non editable, but yet printable documents all over the web. ISO approved the 1.7 version of PDF as an in-

ternational ISO standard in 2008, allowing the PDF implementation to third parties. PDF is very popular even today mainly for exchanging fixed documents emphasizing in application and device independence (Ditch, 2007, pp. 13-14) - (Garrish, 2011, p. 7).

2.3.4 Smartphones, tablets and laptops

In recent years the computing industry delivered a plethora of computing devices, ranging from laptops and netbooks to tablets, smartphones and mp3-players. The early paradigm of personal computing, gradually transformed from cumbersome desktop personal computers to smaller and more manageable mobile devices. Advancements in microelectronics reduced even more the size of microprocessors and hard disk drives, resulting additional miniaturization. The introduction of *Solid State Disks (SSDs)* improved the technical characteristics of local storage, in comparison with magnetic hard disks, and dominated mobile devices. Tablets and smartphones challenged the current perception also in GUI design by introducing touch interfaces. This design innovation diverged from the classic WIMP interface, since it replaced keyboards and mouses with finger taps and gestures on the device's touch screen. These new environments combined with the reduced size of touch screens initiated a new generation of applications and operating systems, capable to integrate to this new hardware.

The smartphone industry emerged from the merger of early Personal Data Assistant (PDAs) and mobile phone industries. At mid 1990s, early PDAs provided limited hardware capability as well as applications functionality. At the same period, mobile phones were transforming from mobile telephone devices to mobile computing devices by integrating CPUs and LCD screens to their design. Mobile phones started to provide basic applications for contacts management and text messaging, but they were far from real computer functionality. The first success story though didn't arrive until 1999, when Research In Motion (RIM) unveiled BlackBerry. At first, BlackBerry started as a two-way pager, but rapidly became one of the most popular mobile computing devices. The ability of BlackBerry to send and receive e-mail with its QUERTY keyboard created a wave of massive adoption by businesses. As a response to BlackBerry's success, mobile phone manufactures, including Nokia, Ericsson, Panasonic, and Samsung collaborated in an effort to create an operating system for their devices. The Symbian operating system gradually dominated the early smartphone market by holding the 65%, but later Ericsson, Panasonic and Samsung abandoned the project and left it entirely to Nokia (Hall and Anderson, 2009, pp. 64-65).

The great shift of the market though occurred in 2007, when Apple unveiled the iPhone and gradually established itself as the major rival in the

smartphones industry. The usable touch interface and the stylish design of iPhone became instantly a desirable product for consumers, establishing Apple as the smartphone market leader. The penetration of smartphones in the consumers market was gradually increasing the number of Internet searches conducted with a smartphone. This attracted the attention of Google, which desired to be present as a search engine in the mobile space. Google's first move was the acquisition of Android in 2005, a company that was developing software and an operating system for mobile devices (Hall and Anderson, 2009, pp. 66-67). The next step was the formation of the Open Handset Alliance in 2007, a cooperation of companies, which promised to transform Android to an open source mobile operating system accessible to any mobile manufacturer (Open handset alliance, 2007). At the time, the leading desktop software company hadn't deliver a rival product, that could compete with Apple and Google in the smartphone market. Microsoft responded to competition not until 2010 with the introduction of the Windows phone 7 series in an effort make a dynamic entrance in the smartphones market. The "live tiles" concept was the core idea of the Windows phone GUI, a dynamic way to access applications and also provide real-time content to the user, contrary to static icons. Microsoft also integrated a hardware button for Bing, Microsoft's in-house search engine, in order to enhance its position in the mobile search engine market (Microsoft Corporation, 2010).

Although computer manufacturers introduced tablet computers already from the early 1980s, they didn't saw commercial acceptance until the introduction of Apple's iPad in 2010. Apple engineers adapted the operating system of iPhone (iOS), in order to fit in a tablet environment capable of supporting a larger touch screen and new applications. Consumer demand for iPad was remarkable, establishing tablets as a new emerging technological market. Apple's success drove many hardware vendors, including Samsung, HP and Motorola, to release devices running the Android operating system, in order to be present in this new market (Milanesi, 2011). The list of competitors grew in 2012 with the release of the Microsoft Surface, a tablet computer featuring the touch interface of Windows 8 (Microsoft Corporation, 2012a).

In the desktop and laptop market though, Microsoft is still the dominant player in operating systems sales (Gartner Inc., 2011). In 2012, Microsoft released Windows 8 the new version of the Windows operating system. Windows 8 supports touch functionality in a new start screen interface based on the live tiles concept as well as the classic Windows WIMP interface (Microsoft Corporation, 2012c). Apple on the other hand represents a small, but significant amount of the overall desktop operating systems market share (Gartner Inc., 2011) with iMacs and MacBooks, their desktop and laptop computers respectively. Apple develops as well as distributes the operating system of MacBooks and iMacs called OSX, numbering its ninth edition co-

denamed Mountain Lion in 2012 (Apple Inc., 2012). Rivals including Google and Canonical also claim a share in this market, apart from the traditional operating systems vendors. Google launched the Linux-based Chrome OS back in 2009 in an effort to appear in the desktop operating systems market, after the major consumer acceptance of the Android operating system in mobile devices. Chrome OS is materializing a fresh concept in operating systems design, since most of the user experience takes place on the web. In contrast to the traditional user desktop, Chrome OS uses the integrated Google Chrome browser for accessing web applications as well as for configuration of the operating system itself (Google Inc., 2009). The market shift to desktop Linux-based operating systems, started with the commercial success of Android and later with Chrome OS. Already from 2004, a small company named Canonical intended to produce a popular Linux-based operating system. In 2004 Canonical introduced Ubuntu, a Linux-based user-friendly desktop operating system directed to the general public and not to proficient computer users (Canonical Ltd., 2006). As Ubuntu was gaining acceptance and followers around the world, in 2013 Canonical unveiled its vision for the "Unique convergence across all four form factors: a phone can provide tablet, TV and PC interfaces when docked to the appropriate screen, keyboard or remote" (Canonical Ltd., 2013). This promise highlights the range of the transformation that mobile computing and digitization of devices contribute to applications design and maybe a glimpse about the future of computing.

2.4 Contemporary trends in writing applications

Today's technology trends drive major vendors to gradually adopt these new hardware and interaction innovations, resulting a transition to a new computing paradigm. Inevitably the processes of writing, reading and communicating are in a transformation phase and directly linked to these changes. We will investigate four main types of applications, ranging from text editing and word processing to desktop publishing and web authoring, in order to describe the state of today's writing applications. Furthermore we will examine phone, tablet, desktop, native and web applications, in order to unveil informative trends in writing applications design.

Apart from hardware and interaction techniques, there are also advancements in information storage with file hosting services. One of the most popular file hosting services is Dropbox. Two MIT graduates founded it back in 2007, in order to provide remote storage as well as sharing and synchronization services, counting now over 50 million users. Synchronization refers to the feature of having the same files in multiple devices using

a special Dropbox folder. Dropbox servers update file changes through an Internet connection, providing file consistency in all devices. Dropbox also supports single and multiuser file revision control and recovery of deleted files, features that are very useful in collaborated writing. The file hosting market numbers many leading companies, operating usually by the marketing term "cloud services", including Apple's iCloud, Microsoft's SkyDrive and Google's Drive (Wang et al., 2012, pp. 1-2).

2.4.1 Text editing

All these changes affect also the design and features of writing applications that cope to fit in this new environment. The first category to investigate are text editors. Computer users use contemporary text editors mainly for code development, configuration files editing and provisional writing like notes. The inability of the produced plain text files to support tables, figures and graphics constitutes text editors an unreliable solution for document authoring. On the other hand their simple and fast interface is sufficient for their purposes, making text editors essential for every operating system or device. Many flavors of text editors exist today in the market for smartphones, tablets and desktop computers. Some popular text editors for the Windows platform are Notepad and Notepad++, for Linux are gEdit, Vim and Emacs and for OS X, the TextEdit.

2.4.2 Word processing

On the other hand office suites are transforming rapidly in the last years, mainly due to the rising of touch interfaces. The need to edit documents, spreadsheets or presentations on smartphones and tablets, shifted the design blueprints creating new rivals to the market. In the Android environment office suites, such as Office Suite Professional and Quickoffice are the most popular contenders, followed by Pages for iPad and iPhone and versions of Microsoft Office in Windows phone and Surface. These word processors follow the WYSIWYG approach and have the ability to edit documents, spreadsheets and presentations, with simple interfaces capable to fit in different screen sizes and orientations.

Apart from mobile applications, Microsoft Word is still dominating the desktop word processing market (Figure 2.1). In late 2012, Microsoft announced the new version of Microsoft Office 2013, delivering many new features with emphasis on touch functionality. Moreover the user has the capability to use a stylus, in order to create, erase and color content, take notes and convert handwritten content to text. Another innovative feature of Microsoft Word

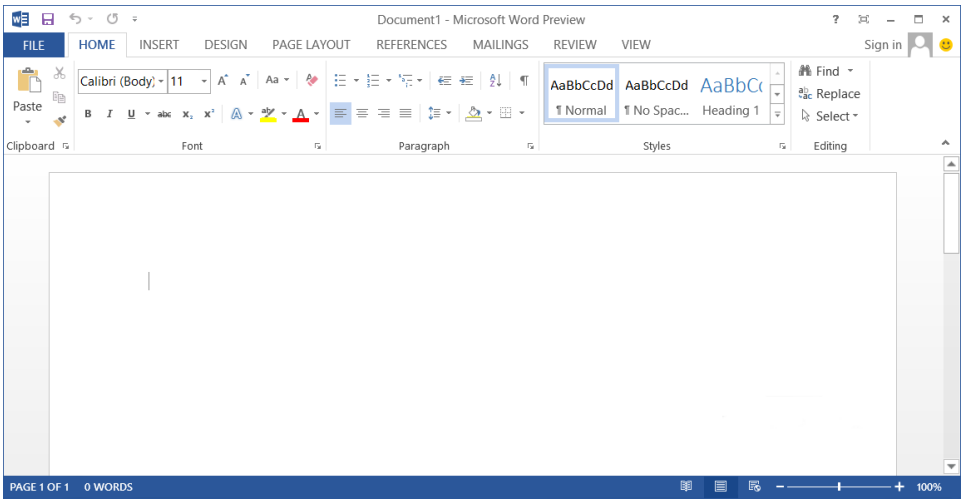

Figure 2.1: User interface of Microsoft Word 2013

is the reading mode. The user can view the document with different layouts, choose colors and navigate easily though touch the text. Moreover, Microsoft Office 2013 became social delivering communicating features through Skype and Yammer, a private social network for businesses. Microsoft Office also uses SkyDrive as a syncing cloud service, for sharing documents in multiple devices. These shared documents are also available off-line and they are syncing when the device reconnects to the Internet. Office is also available as a service with Office 360, a web service providing Microsoft Office 2013 applications though the Internet based on a user subscription (Microsoft Corporation, 2012b).

Office 360 is Microsoft's response to web based application services, such as Google Docs and Zoho. On-line office services provide a reliable alternative to native office applications funded mainly through subscriptions or advertisements, featuring full or basic functionality. Assuming a broadband Internet connection an on-line office application exhibits significant benefits, including easy sharing, collaborating and documents access from everywhere. These services use open or industry standard formats, providing interoperability with other office applications (Ditch, 2007, pp. 29-30). Google Drive[3] integrated the on-line office suite of Google, providing word processing (Figure 2.2), spreadsheets, drawings, presentations and forms (Google Inc., 2012). In addition, Google Drive also features versions for mobile and tablet environments, maintaining a consistency with their desktop versions.

[3]the syncing cloud service of Google

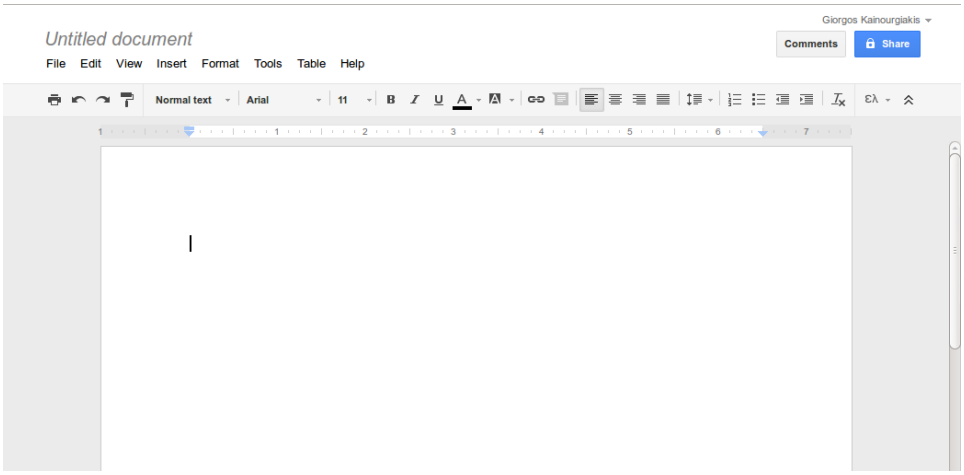

Figure 2.2: User interface of Google Docs word processor

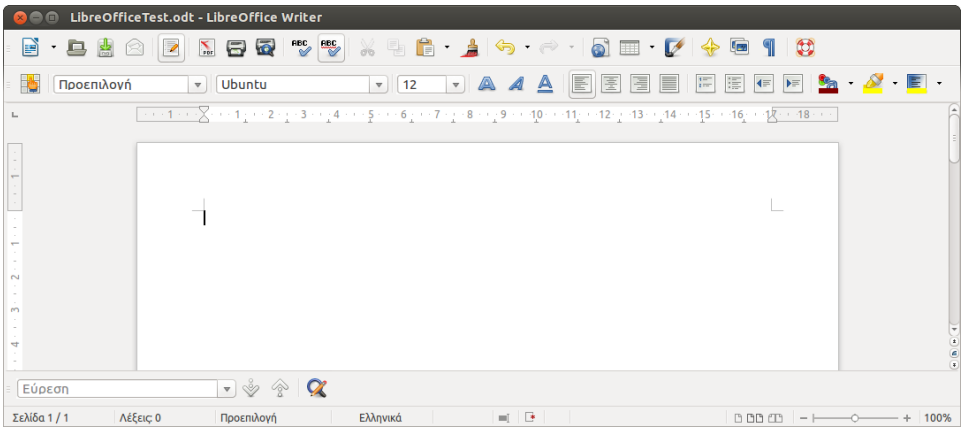

Figure 2.3: User interface of LibreOffice Writer

The open source rival of Microsoft Office was for years the OpenOffice.org suite. Two years though after the acquisition of Sun Microsystems by Oracle in 2010, Oracle donated the code of OpenOffice to Apache Software Foundation initiating the Apache OpenOffice Project. Apache foundation released the first version of the new Apache OpenOffice 3.4 in May of 2012. Claiming over a 100 million users the Apache OpenOffice is still a great open alternative to Microsoft Office, following simple design guidelines targeting mainly to desktop use (Apache OpenOffice Project, 2012). Internal processes though in the OpenOffice community led to the creation of The Document Foundation in 2012. OpenOffice community members formed this non-profit organization committed to deliver a re-branded fork of the OpenOffice suite called LibreOffice (The Document Foundation, 2013a). The open source community welcomed LibreOffice with enthusiasm, resulting many Linux distributions to drop Apache OpenOffice and integrate LibreOffice as their default office suite. Some of the supporters of The Document Foundation include leading companies, such as Google, RedHat, Canonical and Novel as well as organizations like the Free Software Foundation and the Open Source Initiative (The Document Foundation, 2013b). The word processor of LibreOffice named LibreOffice Writer (Figure 2.3) follows at the time the design of OpenOffice aiming as well to desktop and corporate users.

2.4.3 Desktop publishing and e-books

While word processing applications address educational, government and corporate needs in modern societies, they are insufficient for the printing industry. Publication and graphic design corporations use desktop publishing applications for creating printed matter, such as newspapers, magazines, brochures and books. These tools provide a WYSIWYG experience with high quality page-layout, typesetting, images and tables as well as color models ideal for the printing process (Meggs and Purvis, 2011, p. 531). Today's desktop publishing application produce graphics, drawings and publish content even on the web, with numerous features that assist the author. Adobe InDesign and QuarkXPress are the leading WYSIWYG desktop publishing software and industry standards (Adobe Systems Inc., 2012).

Apart from the WYSIWYG approach of modern desktop publishing systems, Latex expresses a WYSIWYM approach in document typesetting and publishing, which is still very popular in the academic community. The majority of LaTeX oriented text editors, including Kile and Texmaker, assist the author to create LaTeX source code (Figure 2.4). In contrast to textual WYSIWYM editors, the Lyx editor follows a different approach. According to Lyx website "LyX is a document processor that encourages an approach to writing based on the structure of your documents (WYSIWYM) and not

```
\documentclass[english]{article}
\usepackage[T1]{fontenc}
\usepackage[latin9]{inputenc}
\makeatletter
\providecommand{\LyX}{L\kern-.1667em\lower.25em\hbox{Y}\kern-.125emX\@}
\makeatother
\usepackage{babel}
\begin{document}
  \title{What is Lyx?}
  \maketitle
  \LyX{} is a document processor that encourages an approach to writing based on the structure of your documents
(WYSIWYM) and not simply their appearance (WYSIWYG)
\end{document}
```

Figure 2.4: LaTeX source code presented in a text editor

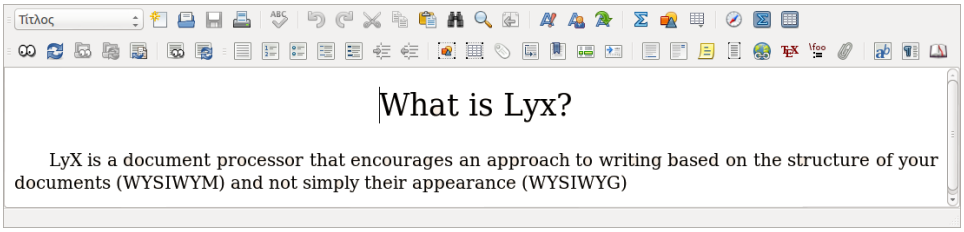

Figure 2.5: Visual representation of the document in Lyx

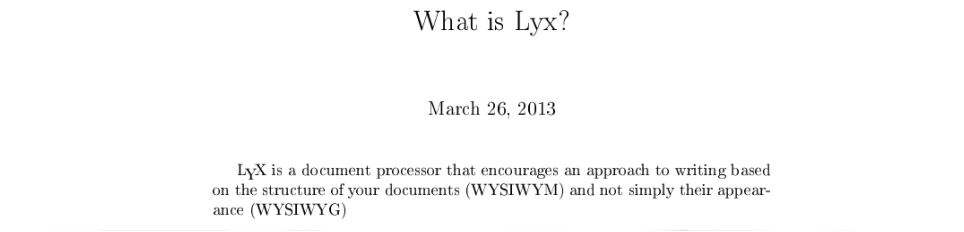

Figure 2.6: Final produced PDF file from Lyx

simply their appearance (WYSIWYG)". The Lyx approach mandates the user to markup document structure and at the same time provides a visual representation (Figure 2.5) that not necessarily correspond to the final printed output (Figure 2.6) (Lyx website, 2013). Usually LaTeX produces a file in PDF or DVI format ready for printing, despite that many users prefer to distribute the document from the Internet and not actually print it.

The increasingly public acceptance of the world wide web as the main distribution channel of information, transformed the publishing industry rapidly and established electronic as well as web publishing, as the most prominent competitors to printed matter. The concept of electronic publishing refers to the authoring and distribution process of electronic documents, such as e-books and electronic articles. A general definition of the e-book (Gibson and Gibb, 2011, pp. 306-307) states that "An e-book is a digital object with textual and/or other content, which arises as a result of integrating the familiar concept of a book with features that can be provided in an electronic environment". The digital form of e-books include open and proprietary file formats, such as EPUB, HTML, PDF, TXT, LIT, RTF and Mobipocket (Lee et al., 2002, pp. 228). E-book reading devices created the variety in e-book formats, since manufactures promoted their own proprietary formats in dedicated e-book reading devices. Today users can read e-book in desktop and laptop computers, tablets and smartphones with an e-book reading application (Gibson and Gibb, 2011, pp. 306-307).

The most popular e-book formats include Adobe's PDF and open standard EPUB, representing WYSIWYG and WYSIWYM approaches respectively. Adobe's PDF is a fixed format representing exact typesetting and positioning of elements in the document, in contrast to EPUB that relies in semantics and the concept of the *reflowable content* (Garrish, 2011, p. 7). The EPUB3 implementation is using many web standards, for instance XHTML5 for markup, CSS3 for visual representation, Scalable Vector Graphics (SVG) for graphics and Javascript for scripting (Garrish, 2011, p. 3). The result is that EPUB e-books can provide reflowable content to any screen or device as well as support for multimedia. Finally, users with disabilities can access the content more easily, since the document can adapt to user's reading preferences.

2.4.4 Web authoring and HTML editors

Apart from electronic publishing, world wide web provides also publishing capabilities oriented to web pages, instead of electronic articles and e-books. Recent advancements in world wide web offer new opportunities in web authoring and publishing, mainly due to the emergence of Web 2.0. In 2004, Web 2.0 introduced a fresh view about the future of the web, the idea of the

"web as a platform" (Anderson, 2007, pp. 5-6). This concept is focusing in six core ideas, including user generated content for individual and cooperate production, the generation of great amounts of data, collaboration, openness and utilization of the network effect (Anderson, 2007, pp. 14-26). Paul Anderson explains that "The Network Effect is a general economic term used to describe the increase in value to the existing users of a service in which there is some form of interaction with others, as more and more people start to use it" (Anderson, 2007, p. 20). The derived services from the "web as a platform" philosophy include multimedia sharing, content syndication, podcasting, content tagging and bookmarking, blogs and wikis. These services are briefly presented below (Anderson, 2007, pp. 7-11).

1. *Multimedia sharing* refers to multimedia storage and sharing services, for instance YouTube for videos and Flickr for photographs. Users can create and share multimedia, contributing to the idea of user generated content.

2. *Content syndication* is the process of gathering information from websites, e.g. a new story's title and synopsis. The RSS format organized this sequential information, forming a RSS feed, which informs users about updated content of a website.

3. *Podcasting* refers to creation and distribution of audio recordings, for instance interviews, lectures or talks, usually in MP3 format that could be played in computers or mobile devices.

4. *Content tagging services* allow users to describe with keywords (tags) a digital object, including websites, videos, texts or photographs, while *social bookmarking services* permit users to store and share bookmarks and favorites remotely.

5. *Blogs* or *Weblogs* are simple websites hosting brief texts which called posts, arranged chronologically in a journal style. Writers can also tag posts with keywords and allow readers to comment on each blog entry.

6. *Wikis* are websites oriented on user collaboration. Wiki users edit the website with simple on-line editors, in order to produce group work. Wikis also use a version control system that allows editors to restore previous versions of a webpage.

Rising web technologies and open standards were the driving forces of Web 2.0 implementation and web applications development. These applications emerged to confront annoyance and time waste of page refreshing and reloading, after the user's interaction with traditional dynamic or static websites.

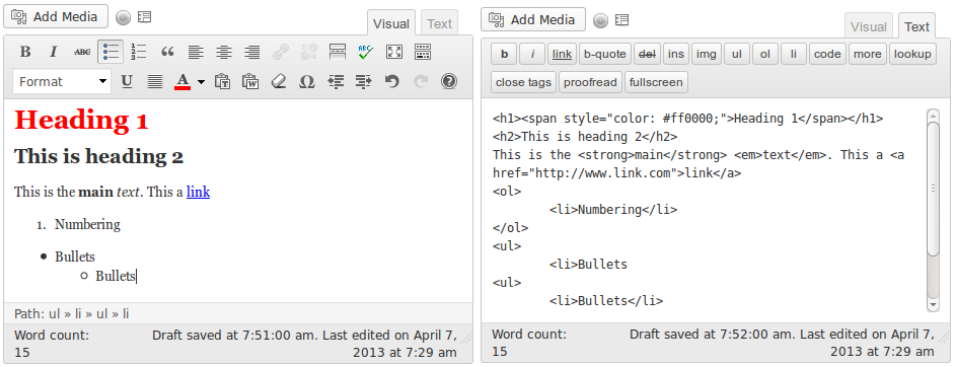

Figure 2.7: Visual and text mode of the WYSIWYG WordPress HTML editor

At first, web-servers of dynamic websites process user requests with server-side scripting languages, including PHP, ASP, Python, Ruby etc. Afterwards they send an HTML page as a response to user's request, causing the reload of the whole page on the client. For long the industry was looking for a solution to page reloading problem. The adoption of Asynchronous Javascript and XML (AJAX) technologies finally addressed the reloading issue and delivered a better experience to the website user. These technologies exchange small amounts of data with the web-server and update only fractions of the page, creating the impression to the user of a responsive interface similar to a native desktop application. AJAX include technologies, such as HTML, XHTML and CSS for markup and presentation, Document Object Model (DOM) for document dynamic control, XML and XSLT for data interchange and manipulation, XMLHttpRequest for asynchronous data retrieval from the server and Javascript for client scripting (Anderson, 2007, pp. 27-28).

Until the rising of Web 2.0, web authors used only native desktop applications, such as simple text editors, HTML-oriented editors and WYSIWYG HTML editors like Adobe's Dreamweaver. Today's authors also use web-based applications, since a great amount of the writing process takes place in the web. Blogs represent the most accessible personal mean for authoring as well as publishing content, attracting millions of users worldwide. These users utilize blog provider services to write their blogs, facilitating personal, educational or even journalistic subjects (Anderson, 2007, p. 15). The majority of blog providers, including Google's Blogger and WordPress.com, use simple web-based WYSIWYG HTML editors for post editing, providing basic functionality smoothly integrated to the service. For instance WordPress uses the open source HTML editor TinyMCE for page and post editing, which utilizes two editing modes shown in Figure 2.7. The Visual mode is a

Figure 2.8: Wikipedia editor featuring wiki markup and final preview

WYSIWYG interface similar to word processors for formating and editing. It supports media upload, character formating, bullets and numbering, block quote and text alignments. The editor also features links, spell checker, outdent and indent, font colors, undo, Microsoft Word and plain text pasting options and style drop-down menu for text structure markup. In contrast, the Text mode allows the user to edit directly the produced HTML code. The user can add a limited set of HTML markup notation or style with CSS following following the WYSIWYM approach. Text formating and style is added inline with CSS style commands, resulting a single HTML file containing HTML blended with CSS (WordPress Support, 2013).

Wikis on the other hand represent a web authoring tool focused on user collaboration. The most prominent success story of wiki software so far is Wikipedia. Thousands of users worldwide edit Wikipedia using MediaWiki software in an the collaborative effort to deliver an on-line free encyclopedia (Anderson, 2007, p. 8). Wikis usually follow a textual WYSIWYM approach based on a simple wiki markup language, contrary to the WYSIWYG approach of blogs. Wikipedia editors write wiki markup in a web-based text editor (Figure 2.8) featuring basic functionality, including character formating, links, file embedment, citing references, structure markup, bullets and numbering, tables and picture galleries. Wikipedia's on-line editor also provides special characters, wiki markup help, a citations tool as well as math and logic capabilities (Wikimedia Foundation, 2013). MediaWiki transforms wiki markup to XHTML 1.0 transitional as an output format. According to the developers of MediaWiki the use of XHTML as a writing format presents a variety of benefits. The most important is the easy integration of other XML namespaces, including The Mathematical Markup Language (MathML) for mathematical notation and SVG for vector images. Another benefit is the easy transformation of XHTML to other formats as well as that XHTML is the most up-to-date version of HTML (MediaWiki Development, 2013). Besides MediaWiki, Walter Ditch also presents recent discussions for

the possibility to use XHTML as a universal document format for government and industrial use. Some of the arguments that endorse XHTML use, include usability of hyperlinks in modern working environments, easy browser viewing and long-term document preservation (Ditch, 2007, p. 31).

Content Management Systems (CMS) are general-use web authoring applications, exhibiting a great variety in designs and functionality. They allow the user to create, publish, manage and edit web content (Laleci et al., 2010, p. 1). Blogs and wikis are also types of CMSs focused in specific orientation and purpose. Today, many organizations and industries extensively use CMSs even to preform critical business operations, such as financial transactions (Maass, 2012, p. 111-112). An extended variety of CMS software exist on the market today, with different licensing models and industry focuses, ranging from tourism and government to publishing and medicine (Maass, 2012, p. 122-123).

Popular CMSs platforms use different HTML editors to manipulate content, including the open source projects Drupal and Joomla. Editors such as TinyMCE[4], CKEditor[5], NicEdit[6], JCKeditor[7] and HTMLBox[8] follow the classic WYSIWYG approach. On the other hand editors, such as MarkItUp[9], BUEditor[10] and wmd[11] use a textual form of the WYSIWYM approach, ranging from simple HTML to BBCode or even wiki markup. BBCode is a lightweight markup language that offers a limited set of tags in square brackets ([ ]). Many websites including forums and blogs use it mainly for messaging purposes (Drupal website, 2013) (Joomla website, 2013).

WYMeditor is a WYSIWYM editor that uses a hybrid approach of WYSIWYG and WYSIWYM similar to the Lyx approach. This editor mandates the user to markup structure and at the same time provides a visual representation of the document. The produced format is a strict version of XHTML, compliant with W3C XHTML specification without visual information, such as font styles and weights, colors and borders. The concept is that other applications should provide a visual representation, while WYMeditor provides only structure and content. The design of WYMeditor is filling the gap between the WYSIWYG and the textual WYSIWYM approach, in HTML editors. According to project developers regular CMS authors often misuse WYSIWYG editors, leading to the reduction of code quality and overall visual representation. At the other hand textual WYSI-

[4] www.tinymce.com
[5] www.ckeditor.com
[6] www.nicedit.com
[7] www.joomlackeditor.com
[8] www.remiya.com/htmlbox/
[9] www.markitup.jaysalvat.com
[10] www.drupal.org/project/bueditor
[11] www.code.google.com/p/wmd/

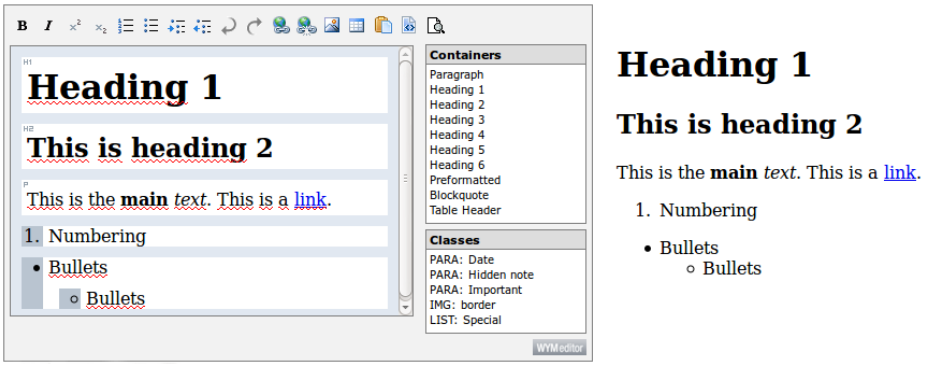

Figure 2.9: WYMeditor interface and final preview

WYM editors based on HTML, BBCode or wiki markups are very complex and confuse the users, resulting difficulties in the authoring process. The WYMeditor approach is based on a concept "to leave details of the document's visual layout, and to concentrate on its structure and meaning, while trying to give the user as much comfort as possible (at least as WYSIWYG editors)" (WYMeditor website, 2013a).

The WYMeditor interface is shown in Figure 2.9, featuring basic functionality, such as character formating, bullets and numbering, indent and outdent. Also it provides undo, redo, insert image and table, links, paste from Word, show HTML and preview. Besides the main toolbar, there are options on the right side of the editor called containers. With containers the author can define structural semantics of the document, including paragraph, headings, quotations block and preformated text. Some HTML classes are also available for paragraphs, such as date, note, important and borders for images. WYMeditor also provides a visual representation of each container used, by visually distinguishing each container from each other with a white background and a distinctive tag, such as P, H1, H2 etc. Adding and removing containers as well as bullets and numbering, is possible by the Enter and Backspace keys respectively, while Backspace is also used to delete images (WYMeditor website, 2013b).

Users insert tables from the toolbar, providing number of rows and columns, table's caption and summary. The user can manipulate and edit the table with a simple tool shown in Figure 2.10. The tool provides the capability to add columns left and right from the working cell or to delete the working cell's current column. The same concept applies respectively to the row of the working cell. The interface of the tool consist from a left and right arrow followed by an x-mark in between. Accordingly it provides an up and down arrow with an identical x-mark for rows. Another feature of tables is

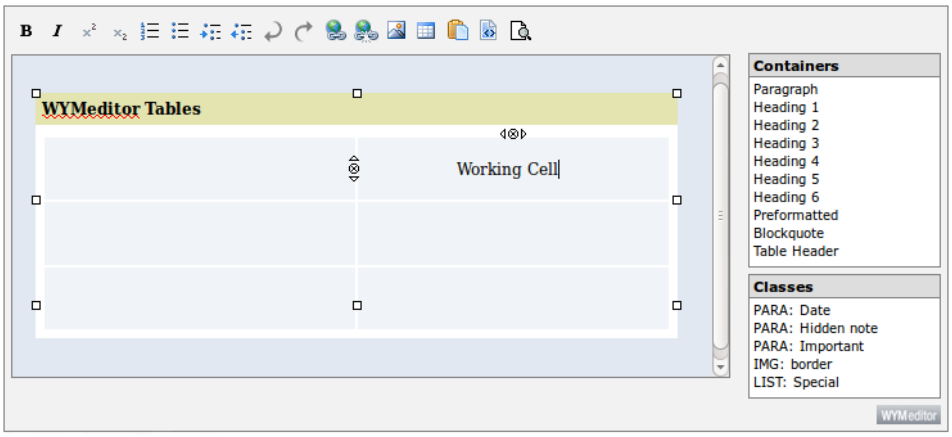

Figure 2.10: WYMeditor's working cell table editing

that WYMeditor allows the user to resize the table for authoring comfort, although tables do not save visual presentation information (WYMeditor website, 2013b).

The WYSIWYG and WYSIWYM fusion approach of WYMeditor could provide answers to the ongoing discussion about writing approaches and document formats suitable for the Internet age and device diversity. WYSIWYG critics argue that web WYSIWYG writing applications produce repetitive and unnecessary markup, in order to provide a full featured WYSIWYG writing environment. This phenomenon increases storage and transmission costs and causes delays in download and browser rendering times. Inefficient markup is consuming time and financial resources of the service provider and this is eventually passes on to the end user. Spiesser and Kitchen (Spiesser and Kitchen, 2004) proposed some techniques to optimize automatically generated HTML code by WYSIWYG authoring tools. Their approach included removal of HTML with no effect, proprietary tags and whitespaces. They also employed style classes to transform in-line CSS styles to header CSS styles, for tags with the same style. Using dynamic programming they reduced the size of the document about 33%, demonstrating the magnitude of the inefficiency of the HTML code Microsoft Word and Frontpage produced (Spiesser and Kitchen, 2004, p .335). In contrast, hybrid and textual WYSIWYM writing applications are producing more efficient HTML code, since they provide a reduced set of visual presentations capabilities and they allow the user to control the produced code.

Moreover, the use of a WYSIWYG writing application requires from the author to design and produce the final layout of the document. This presupposes at least some elementary abilities in visual design, typography and

skills in using a specific writing application. Authors consume time and effort to this procedure distracting them from the creation of content, which is the essence of the writing process (Lamport, 1994, p. 8) (Sauer, 2006, p. 3). The division of a document's structure from presentation is a promise made back from the emergence of SGML. LaTeX utilized this concept for years, but it became popular now with the rise of wikis. Wiki markups allow authors to provide only semantic information and content without visual layout, since presentation is already defined by the wiki software.

In addition, the rising diversity of computing devices constituted the WYSIWYG approach even more problematic. Many different devices with varying screen sizes, computing capabilities and operating systems flooded the market causing problems with readability, accessibility and homogeneous design. The result was the need for visual optimization of a document for every possible configuration. Fixed page formats such as PDF, OOXML and ODF sacrifice visual layout in favor of readability, since mobile devices display them as reflowable content. In this case the WYSIWYG approach partially loses its purpose, since the author can not deliver the original document visual layout to the end reader. Using the WYSIWYM approach, the same document could be presented with various visual layouts, since it's semantically defined (Sauer, 2006, p. 3).

In contrast, there is much criticism of the textual WYSIWYM approach and specifically for wiki markup languages. This could also apply to other more complex markup languages, including HTML or LaTeX. Christoph Sauer (Sauer, 2006, p. 2) identifies four wiki markup shortcomings regarding the user's initial wiki markup confusion, the inability to modify the layout, the absence of document overview and finally the numerous different wiki markups. In general, wiki authors before they write in wikis they should first learn a wiki markup language, since documents with wiki markup and content usually puzzle them. Moreover, the textual WYSIWYM approach removes the formating ability from users causing loss of visual production freedom, which is a major drawback in comparison to the dominant WYSIWYG approach. Also the direct absence of document overview and the fusion between wiki markup and content, results to reduced readability that causes difficulties in the writing process. Finally, the problem of multiple wiki markups, referred to as the "wiki markup mess", raises barriers in broader wiki adoption as a writing application. Despite these shortcomings, a wikis usability study (Désilets et al., 2005) though conducted to 4th grade children indicated that wikis are generally usable. The usability issues concerned the creation and management of links and images, accounting the 49% of all problems and the 79% of the catastrophic problems.

Many wiki engines and editors use numerous approaches to address the usability issues of wikis. WikiWizard is an editor developed for JSPWiki,

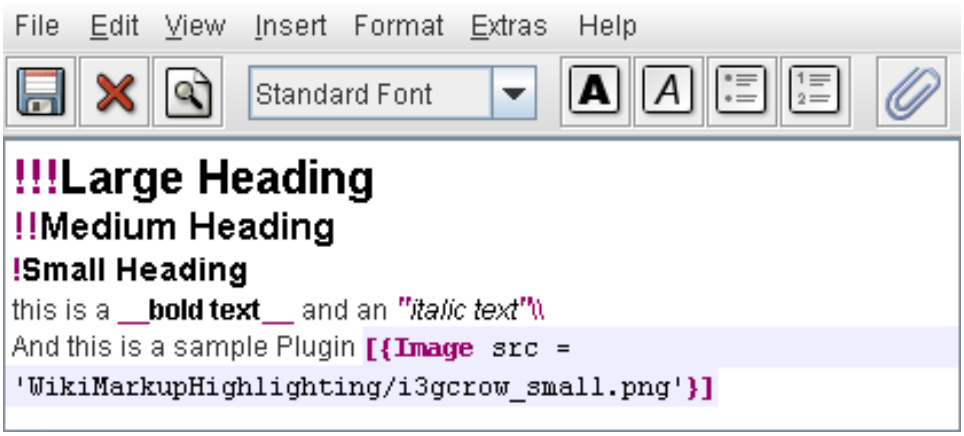

Figure 2.11: WikiWizard interface with visual feedback and semantic highlighting

that uses wiki syntax highlighting and visual feedback shown in Figure 2.11 (WikiWizard website, 2013). Designers intended to create an editor, with familiar user interface using menus and toolbars, to provide instant visual feedback by formating fonts to headers, bold, italics etc, without hiding the highlighted wiki markup (Sauer, 2006, pp. 4-5). This approach is another fusion of the WYSIWYG and WYSIWYM approaches, emphasizing wiki markup instead of visual formating.

Apart from general purpose CMSs, educational institutions employ web-based information systems called Learning Management Systems (LMS). LMSs are also referred to as e-Learning platforms, e-Learning environments, distributed learning systems, course management systems and instructional management systems. Their purpose is to create a virtual learning environment and integrate a wide range of pedagogical and course administration tools to assist the learning procedure (Coates et al., 2005, p. 19-20). LMS use is widespread in higher education institutions, including open source and proprietary solutions. In 2011, the market leader was the proprietary Blackboard LMS representing a 60% of the market, followed by the open source project Moodle with 19%, while others such as Sakai and Desire2Learn represented 7% of the market each (Grajek et al., 2012, pp. 19-20).

Blackboard uses a WYSIWYG HTML editor based on TinyMCE, delivering a classic WYSIWYG experience through a simplified or an advanced user interface. The editor also integrates WIRIS a new MathML equation editor for mathematical notation (Figure 2.12). WIRIS is an HTML5/Javascript application providing an simple interface for basic mathematical operations, matrix calculus, calculus and series, logic and set theory as well as units

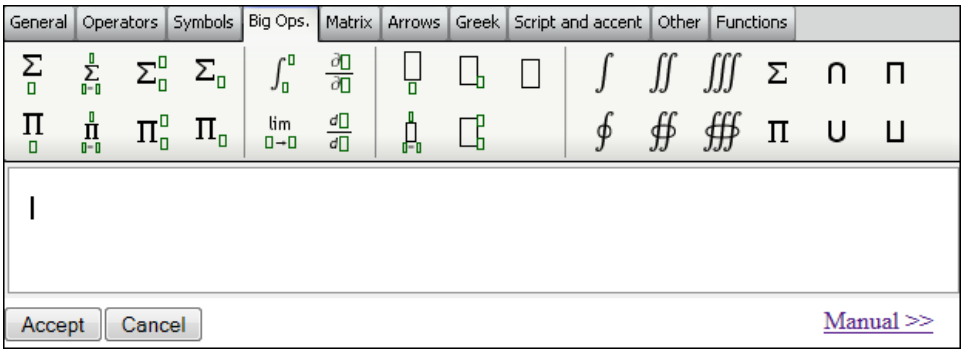

Figure 2.12: WIRIS, the equation editor of Blackboard LMS

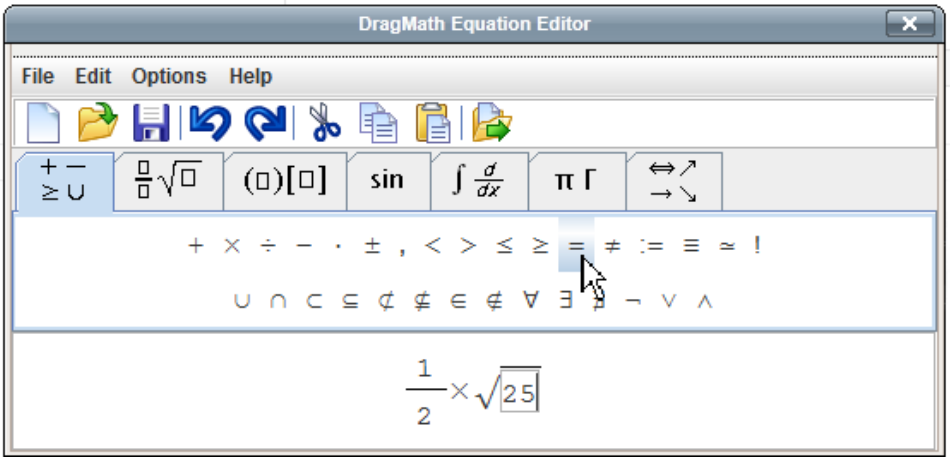

Figure 2.13: DragMath, the equation editor of Moodle LMS

and greek alphabet. Mathematical formulas and equations embed directly to the HTML document as an editable MathML form. Another functionality of Blackboard content editor is the capability of adding mashups. This feature allows the integration of content from third party Internet services, such as Flickr Photos, SlideShare presentations, YouTube videos and content from NBC. Finally, the edit CSS function provides a compact interface for configuring the overall visual layout of the page, regarding text, backgrounds, paragraphs, boxes, borders, lists and positioning (Blackboard Help, 2013).

The open source Moodle platform is the major competitor of Blackboard, which also using a version of TinyMCE as the main WYSIWYG HTML editor. The user has the ability to configure the functionality of the editor by adding or removing editor plugins, in order to create a desirable tool-

bar configuration. The overall functionality and behavior of Moodle's editor is similar to a classic WYSIWYG HTML editor (Moodle 2.4 Documentation, 2013b). The point, in which Moodle HTML editor 2.4 diverges from Blackboard's content manager, is mainly in the area of mathematical formulas. Moodle uses an open source Java equation editor called DragMath, which supports a wide range of mathematical syntaxes, including MathML, LaTeX, Maple and Maxima. DragMath's mathematical interface is similar to WIRIS, although it features also drop-down menus and an additional editing toolbar, besides mathematical symbols (Figure 2.13) (Moodle 2.4 Documentation, 2013a).

2.4.5 Semantic web and semantic annotation tools

A Scientific American article introduced the term *Semantic Web*, already back from 2001. Tim Berners-Lee described his vision about a universal and decentralized extension of the existing world wide web, providing structure and meaning to web pages. The gist of semantic web is the creation of an environment based on meaningful content for humans as well as computers. In this environment software agents could execute complex tasks for users, by roaming from page to page extracting useful information and enhancing the cooperation between them (Berners-Lee et al., 2001).

In order to built the semantic web, W3C developed a number of standards, including the Resource Description Framework (RDF), RDF Schema (RDFS), the Web Ontology Language (OWL), a query language for RDF called SPARQL and the currently working Rule Interchange Format (RIF) (Hitzler et al., 2011, p. 14). The basic building block of semantic web is the XML based RDF. This framework describes data by a set of triples called statements in a form of "Subject - Predicate - Object" forming a directed labeled graph. Subjects and objects (classes) represent the nodes of this graph, while the edges between them represent their relations (properties). RDF Schema and OWL provide formal representations of these properties and classes, also called *ontologies*, in order to facilitate further connectivity to other nodes or graphs (Polleres, 2010, p. 2). The names of the nodes and edges of these graphs are different, in order to distinguish from each other. RDF uses Uniform Resource Identifiers (URIs) for naming, a generalization of Uniform Resource Locators (URLs) used for web addressing. The result is that every node and edge are corresponding to a single name similar to a web address. Ontology languages such as OWL strictly define these node and edges with an ontology (Hitzler et al., 2011, p. 21). Finally, the SPARQL standard allows queries in RDF data similar to SQL in relational data, providing query capabilities. W3C is working on future editions of SPARQL, in order to extent its capabilities and deliver many more requested database

features to the public (Polleres, 2010, p. 3).

Despite W3C's effort to define and develop semantic web technologies, web developers often criticize the semantic web concept for being too ambitious and complex. They also argue that is not addressing issues of the industry as well as applications development. Microformats represent an alternative approach to semantic web standards, in order to provide semantic information into a web page. They assign attribute values of existing HTML elements, in order to embed semantic data to the document. The simplicity of the microformats approach trades off with limitations in the application area and extensibility, in comparison with the general purpose semantic web standards. W3C responded to microformats with the development of the RDF in attributes (RDFa) standard, in an effort to merge RDF-based data into XHTML documents (Hitzler et al., 2011, p. 14). Nevertheless, with the emergence of HTML5 another semantic contender appeared, attempting to reduce the distance between the constrains of microformats and the abstractions of RDFa. Microdata use special additional attributes that can describe semantic entities, such as `itemscope` and `itemprop`. Again the major advantage of this method is simplicity. The absence though of a specified schema generates compatibility issues, since authors can describe data semantically without an ontology (Mayer and Guinard, 2011, p. 5).

Nevertheless the semantic web is growing recently and the need to share data as well as ontologies is rising. One of the major drivers of the semantic web is the scientific community, since many fields share and integrate data from other fields. Using an unified working platform based on ontologies (e.g. for biology), scientists from other fields (e.g. medicine), can use the produced data and semantics for their own research. Nigel Shadbolt, in order to emphasize the utility of the semantic web in scientific research, states that "The need to understand systems across ranges of scale and distribution is evident everywhere in science and presents a pressing requirement for data and information integration" (Shadbolt et al., 2006, pp. 96-97).

Semantic annotation is the process that produces semantic metadata, assigning semantic information to data resources. Researchers divide semantic annotation in three categories in terms of user involvement, including manual, semi-automatic and fully-automatic approaches (Oren et al., 2006, p. 1). Manual semantic annotation applications provide assisting visual tools for user-provided annotation, browsing, semantic information reading and web page navigation. Other more simple applications provide just a basic infrastructure to manually markup semantics into documents. In contrast, semi-automatic and automatic approaches focus on creating semantic metadata automatically. They employ natural language processing, document structure analysis and machine learning approaches, either implemented with training sets or with human supervision. These annotation

approaches though are partially successful until today (Laclavik et al., 2012, pp. 1000-1002). Furthermore in terms of formality, annotations also include informal, formal and ontological annotations. The most basic form is informal annotations, which usually include only human readable raw text. On the other hand, formal annotations use a machine-understandable formal language to describe themselves e.g. a markup language, while ontological annotations use machine-readable formal languages based in an specific ontology (Oren et al., 2006, p. 4).

Collaborative writers frequently use document annotations in the writing process as well as in reading for highlighting purposes. For instance, Annotea is a tool that focuses on manual document annotation. It marks-up comments and externally assign them to corresponding fragments of the document. A different example is Ontomat , a manual annotation tool which focuses in external ontological object annotation (Oren et al., 2006, p. 7). Although the majority of these applications assume that annotated documents exist in HTML/XML form, tools such as the SemanticWord assist Microsoft Word users to markup semantic content in Word files using a integrated graphical user interface (Uren et al., 2006, p. 15). Other approaches to semantic annotation include semantic wikis, semantic blogs and CMSs as well as image and video annotation tools. Semantic wikis use wiki markup to annotate semantic content and provide annotations to a web page. Wikis either allow the user to annotate content in the same page or store annotations separately to a URI. Some tools reuse existing ontologies through URIs and namespaces definitions, while others use annotations from internal wiki pages. In addition, several blogs incorporate semantic information, such as addressbook entries, bibliographic data, subjects and general post information (Oren et al., 2006, pp. 7-8). In the CMSs industry, the growing need to annotate and later sustain metadata in changing documents, challenges the scientific community. Researchers are working to develop reliable solutions for large content repositories, meeting the needs of the industry and the scientific community (Laleci et al., 2010). Finally, the concept of semantic web extents the boundaries of document annotation, attempting to include multimedia annotation as well. Many image and video annotation tools work in this direction based on varying approaches, representing another challenging research issue (Dasiopoulou et al., 2011).

RDFa Content Editor (RDFaCE) is a WYSIWYG web-based HTML editor derived from TinyMCE, providing either manual or semi-automatic RDFa and microdata annotations. The main innovation of RDFaCE is the use of four content views, including the classic WYSIWYG, a WYSIWYM view of the annotated data, an RDF triples view (Figure 2.14) and a source code view. The WYSIWYM view extents the classic WYSIWYG interface of the editor, with different color highlights of the annotated objects. The user can

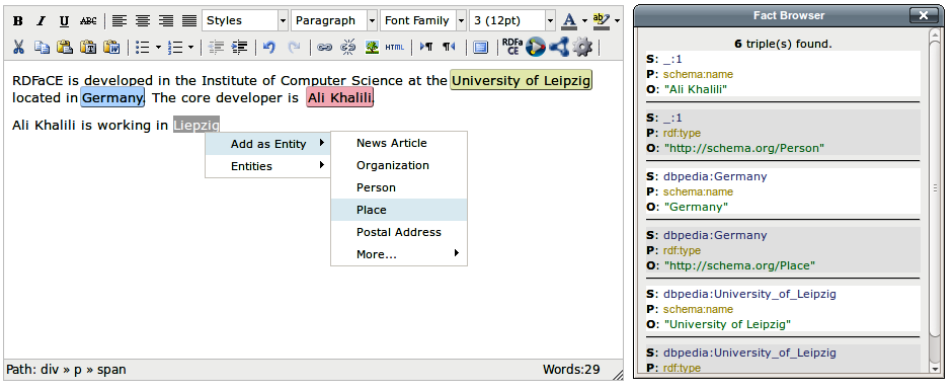

Figure 2.14: RDFaCE editor views, WYSIWYG and WYSIWYM with RDF triples view

modify information of the entity with a double click on the annotated area, while pointing with the mouse RDFaCE reveals the main category of the entity e.g. person. Moreover the user can create new manual annotations with a single right click on a text selection, choosing the desirable category from an entity menu. The third view consists of RDF triple browser called "Fact browser", providing information about the created RDF triples in a "Subject - Predicate - Object" form. Finally, the source code view provides the produced HTML code of the document with the integration of the RDFa annotations. Apart from manual annotation, RDFaCE supports also semi-automatic annotation using Natural Language Processing (NLP) web APIs. These services extract information from the text and later provide automatic suggestions for entities and relations, supporting the annotation process (Khalili et al., 2012).

2.5 Conclusions

The gist of this review is to unveil patterns of writing culture and the technological evolution of writing as well as to provide the state of the art of modern writing applications and information storage. In the first section we present patterns of writing culture significance referring to social organization, cultural transmission, cultural innovation sharing and adoption as well as to diversity and importance of writing systems. In the second section, we illustrate properties of successful writing materials and instruments, in order to investigate if the modern computer could meet them as a writing instrument. These properties include adequate production, economic viability, portability, storage capability, durability and usability. In the last

section we define the basic blueprints of our application's design referring to the type of the application, development tools that will use, supported operating systems, our writing and interface approach as well as the produced file format.

2.5.1 Writing culture

The first emerging pattern appears in *social organization.* As we saw, writing emerged as a cultural transmission and social organization tool, when human societies transformed from hunter-gatherers to farmers. Written word gradually redefined domains including the economy, the legal system and the communication, in order to fit in the new paradigm of the rural society. Equivalent organizational needs existed also in modern societies along with requirements for computation and automation. Ranging from the unit record to mechanical calculators and early computers to office suites, the need for accounting and business organization was evident. Writing combined with computers addressed these issues with word processors, spreadsheets and later databases, demonstrating the continuous need for social organization from the antiquity until today. As a conclusion, our writing application should conform with the requirements of the current productivity paradigm and provide innovative solutions to organizational issues. For example the easy creation of corporate documents, presentations, charts or spreadsheet calculations are features that could integrate to our design.

Even from antiquity, people employed writing for *cultural transmission*, since it could transmit knowledge without the physical presence of a speaker. The transmitted knowledge was also consistent in the passing of time contrary to the oral paradigm. Document reproduction techniques appeared, in order to extent the life span of a document and at the same time to expand its reading audience. Stone inscription rubbings, seals and woodblock printing in China and later the movable type and the printing press in Europe were the printing techniques that transformed writing to a mainstream culture. Document reproduction and distribution became important issues in the emerging literate society, perfecting the printing techniques as well as creating a variety of distribution channels. News stands, bookstores and libraries distributed printed matter creating a vibrant publishing industry. Computers and the Internet though altered the traditional paradigm of publishing, creating electronic as well as web forms of publishing. Internet became the distribution channel of the new publishing industry, providing reduced costs, efficiency and high speeds. Electronic and web documents are replacing gradually the printed matter today, demonstrating that our writing application should follow this concept. Web authoring technologies provide interaction and collaboration in platforms, such as wikis, CMSs, LMSs and

blogs, requiring our design to integrate to these platforms. The most appropriate form of writing application that could meet these requirements is the web-based HTML editor. In addition, our educational focus adds more requirements to the design, ranging from scientific notation to drawing tools, animations, games and videos. Web content is capable to integrate various other web applications that could assist the learning procedure. The resulting design should be usable for students and teachers as well as writers and readers.

In this review, we saw that major technological achievements, such as paper, printing or even writing itself required many years or even centuries to spread across the world. This is demonstrating the slow process of *sharing cultural innovations* between different social groups. Causes of this phenomenon in antiquity were the absence of communication between different cultures and the unwillingness to share innovations with others, in order to retain an relative advantage e.g. preserve an economic interest related to a specific innovation. Instances of this, were the monopolies of papyrus and paper in Europe, from Ancient Egyptians and later from Arabs. The producers didn't share the manufacturing techniques of paper and papyrus production with Europeans, in order to maintain the dominant position in writing materials market.

In modern times, the main barriers of sharing cultural innovations were quite similar, including corporate strategies and profits. Software and hardware vendors produced and sold a product, aiming to create an industry standard and dominate the market. This was evident from the emergence of the first computer in 1950's until today. Hardware companies created a new computer based in a emerging technology, while software companies created operating systems and applications for the new device. The typical success story for the hardware industry was the creation of the IBM PC and the IBM compatible ecosystem, while for software industry was Microsoft applications and operating systems. These practices led to market lock-ins from vendors, discouraging competition, interoperability and innovation. Recently the open standards adoption, including ODF, OOXML, PDF or HTML5 technologies, from major vendors changed the paradigm of developing software at least in the file format level and created a more friendly corporate environment to different approaches. Our application will integrate open development tools as well as open standard formats for the produced documents.

Moreover in the early software market, dominant word processing applications were dependent of the dominant operating system. Electric pencil was leading in computers without a standardized operating systems, due to many versions. WordStar and later WordPerfect dominated respectively the CP/M and MS-DOS ecosystems, while Microsoft Word led in the Win-

dows environment. In touch devices identical patterns emerged with new writing applications for the Android and iOS ecosystems. After the emergence though of Web 2.0 and the concept of the "web as a platform" a great amount of the writing process now takes place on the web. Wikis, CMSs, blogs and web office applications, such as Google Drive and Zoho, in combination with remote information storage and syncing services can provide a viable alternative to native writing applications that depend on operating systems and native development tools. This is another argument to create an operating system agnostic application based on web development technologies, such as HTML5 and Ajax.

Apart from cultural sharing, social groups regularly exhibit reluctance to *adopt innovation* by insisting to traditional practices. In this case the innovator shares the innovation, but the users do not adopt it. An example of this was the transition from parchment to paper as the dominant writing material in Middle age Europe. Many regions of Europe banned paper use, due to its fragile nature, having the result that paper became popular not until the printing revolution. Another example was the QWERTY keyboard layout use in the IBM Selectric and the electronic typewriter. Designers selected to install a QWERTY keyboard layout despite that they designed the typewriters in order to avoid jamming, the reason that QWERTY created initially. Dvorak already from 1932 had introduced, after years of studies, the "Dvorak" keyboard layout that improved typing speeds, accuracy and reduced fatigue. The poor adoption though by the millions typists who already were proficient in QWERTY made the "Dvorak" layout a marketing failure (Condoor, 2004). In addition we saw that the need for correspondence between the document layout of the computer screen and the printed output created the WYSIWYG approach. Many writing applications still use interfaces based on this approach, even though there is no correspondence to printed output, such as web-based HTML editors or word processing systems for tablets and smartphones. Applications provide familiar interfaces to the user even though they do not provide the original functionality. Users tend to confuse from different writing approaches and unfamiliar interfaces, a fact that was evident in wiki software use. To conclude, our design should take into account the dominant approaches in writing, in order to create a smooth transition to new approaches. One of these features could be the visual feedback of the document to the writer, delivering an approach similar to Lyx or WYMeditor.

Back in ancient China, early printing based on seals showed slow adoption rates, due to the logographic nature of the Chinese script. Yet later in the West, movable type a similar technique to seals demonstrated major success, when Europeans used it with alphabets. This could underline the design significance to comprehend the diversity of writing systems and create writing

instruments and materials specifically designed to meet each writing system's requirements. To illustrate the writing brush was ideal to inscribe Chinese script on paper and the rectangular reed to carve cuneiform script on clay and not vise versa. The western paradigm of writing is a fraction of the world's writing systems, therefore the design process of a contemporary computer based authoring tool is affected of many aspects in order to be usable by different writing cultures. These aspects can be the type of the writing system such as logographic, syllabic, alphabetic or mixed, text orientation (left-to-right, right-to-left, top-to-bottom or bottom-to-top) and input methods. Due to the plethora of writing systems and our incapability to investigate all, our writing application will be designed for western left-to-right top-to-bottom alphabets.

2.5.2 Writing materials and instruments

In this section, we investigate properties of successful writing technology throughout history. The first emerging quality of writing materials and instruments is *adequate production*. The reduced production of quality papyrus in Egypt was the most evident example of writing material scarcity. Papyrus during antiquity was sufficient for the limited writing needs of the ancient world and insufficient for the rising needs in the Middle ages. Egyptians produced papyrus only from a aquatic plant growing mostly in Nile's delta. Paper in the other hand could be made from a variety of abundant raw materials, including mainly pulp of wood, rags or even grass. Paper gradually dominated the world as a writing material, due to its adequate production combined with the *economic viability* of its raw materials in most regions. Parchment was also abundant in most parts of the world, but it wasn't economic viable, due to elevated cost in livestock. Parchment use required the death of many animals, in order to create a single book. Economic viability in terms of writing materials was a key aspect, which advanced over the years by reducing production costs.

On the other hand writing instruments such as pens, reeds and brushes were mostly affordable and easy to produce. The only two writing instruments that were very complex to manufacture as well as expensive, were the printing press and the typewriter. However, the printing press reduced the cost of document reproduction vastly, while the typewriter elevated document quality to the level of printed documents. These features made their manufacturing cost and their complexity a beneficial trade-off, even though these instruments weren't abundant nor economic viable in most regions of the world. The case of the computer though is the most complex of all. In this review we presupposed computer use along with other technologies, such as power grids and telecommunications infrastructure. Power supply and

Internet connections are essential in our perspective, while they are evident mostly in developed nations and not in developing ones. This makes the cost and the abundance of computers irrelevant in these countries, since even they could afford computers they couldn't use them. The penetration of computers and Internet is very poor worldwide, but with increasing trends even in developing nations (Chinn and Fairlie, 2010). At the same time, mobile phones and mobile phones Internet access exhibit a great penetration, even in the most poor countries of Africa (Stork et al., 2012). This means that computers are getting more accessible and affordable in every part of the world, either in a conventional or in a mobile form. These findings also exhibit a promising future of computing devices as writing instruments, even though manufacturers do not produce them everywhere and a great majority of the world population still can not afford them.

The next significant advancement of writing technology was the property of *portability*. From clay and wooden tablets to bamboo slips, papyrus and later paper the evolutionary course of writing materials was towards to increased portability. The reduction of weight and volume combined with increased information density was very useful for communication as well as storage. The same trend applied in later also in information storage ranging from punched cards and magnetic tapes to floppy, hard disk and SSD drives. Modern devices can store and retrieve enormous amounts of information, while the Internet and remote storage provide access to information instantly from anywhere with a single mouse click. Writing instruments on the other hand were generally portable except the typewriter and early computers. Many manufacturers applied improvements to the original typewriter design to reduce its volume and weight, but without significant success. In contrast, computer manufacturers transformed computers from enormous electrical machines to laptops, tablets and smartphones. As a result, modern technology continued the trend of antiquity in terms of portability, revolutionizing further computer hardware, information storage as well as communication.

Other indicators that shaped writing technology during history was *durability* and *document reproduction*. Although writing materials evolved from more durable materials such as clay, wood, bamboo and parchment to less durable like papyrus and paper, the need for document preservation was evident from antiquity until today. At the same time, scribes also preferred writing materials that they could easily inscribe without damage. In most cases though, durable materials were usually cumbersome, expensive or difficult to inscribe on. Parchment for instance was much more resistant compared to papyrus and paper, but it wasn't economical viable or simple to produce. Physical document durability was gradually diminishing, since documents could last in time through reproduction. Destructible materials like paper dominated everyday life mostly due their capability to easily re-

produce. On the other hand, writing instruments were mostly consumables, until the emergence of metallic pens. The printing press and the typewriter were also durable, due to metallic components, but they needed also effort for maintenance. The modern computer follows a different life cycle, since the hardware is wearable and at the same time software updates could make the computer also unusable. Electronic documents though could potentially live indefinitely in storage devices, since they could reproduce very easily with a single mouse click.

Usability is a major issue of writing technology that generally involves the reduction of effort from the writer and the reader. In writing materials, usability translates to easy manipulation of the material, easy and clear surface inscription, increase of document quality as well as information density. The evolutionary course of writing materials in terms of usability started from cumbersome and difficult to inscribe materials, such as clay tablets, wood and bamboo, to flexible ones including papyrus, parchment and paper. Craving stone, clay or wax soon transformed to writing in these surfaces that were also capable to retain ink. Paper and parchment though were also capable to be inscribed on both sides, an ability that doubled the information density leading to the reduction of document volume and cost.

On the other hand, the evolution of writing instruments from reed sticks to writing brushes and goose quill pens to fountain pens was a race for writing usability. These instruments evolved to retain as much ink as possible, to reduce user's writing effort as well as to produce a clear readable document. Later, the typewriter's dominance highlighted the need for high quality readability, equivalent to printed documents. Moreover, another example of usability was the electric and electronic typewriters, which further reduced the typist effort in stroking keys and check for errors. The computer though revolutionized the process of writing in every aspect of usability, error handling and productivity. Computer users for the first time could manipulate the document's layout, integrate figures, easily correct errors, store and sent electronic documents thought computer networks. The evolution of writing technology gradually reduced writing and reading effort, while at the same time increased the *inner complexity* of writing instruments. From wooden and reed sticks to metallic pens and from typewriters to modern computer's applications, inner complexity showed remarkable increase, while required a great effort of cooperation from many different stakeholders including software developers to hardware manufacturers.

2.5.3 Writing applications and storage

In this review we investigated various types of writing applications, covering text editing, word processing, desktop publishing, web authoring as well as

the process of semantic annotation. Text editors exist today for specific operations mainly relating to code development, editing configuration files as well as taking notes. Word processors and desktop publishing applications are still focusing on printable digital documents as well as the WYSIWYG approach and WIMP interfaces. This means that they could not easily fit to emerging web technologies, touch interfaces and mobile devices. Recent developments in e-books and electronic publishing industry indicate a shift from fixed formats, such as PDF, OOXML and ODF, to reflowable content formats like HTML and EPUB. Moreover, the rising of Web 2.0 radically transformed web publishing and user collaboration, indicating the importance of the web-based HTML editors. These applications could easily integrate to various web applications and at the same time could produce reflowable content also viewable from mobile and touch devices. Our educational and scientific focus mandates that our application should be able to integrate smoothly to LMSs, CMSs, wikis and blogs providing a great user experience. Finally, the penetration of Semantic web in scientific practice indicates that modern writing applications should be aware of semantic or document annotation, by providing useful interfaces for manual annotation. This is another argument to create a web-based HTML editor, providing touch functionality as well as mobile usability. At the same time our writing application will provide manual interfaces for document and semantic annotation.

The decision to develop a web-based HTML editor relies to the integration capability to other web applications, such as wikis, CMSs, LMSs and blogs. This means that our application should not be dependent from server-side scripts and run entirely on the client, which in this case is the browser. The family of HTML5 technologies, including HTML5, CSS3, Javascript and various Javascript libraries is the most modern and open platform for web development today. Other rich Internet applications technologies, such as Adobe Flash, Java or Silverlight, are not open and require browser plugins, in order to execute. In addition, our application could also run as a native application in a browser and store the produced documents locally as well as remotely. The file format of the produced documents will be a version of XHTML, since it's ready for the web and could integrate SVGs, videos, images, MathML and mashups, while at the same time produces reflowable web content that could transform to any desired environment. Moreover, the web nature of our editor is allowing our application to potentially execute in every operating system with a modern web-browser.

The most critical decision for our design though is the writing approach that we will follow. In this review, we presented WYSIWYG, WYSIWYM and hybrid approaches as well as their advantages and drawbacks. Our deriving conclusion is that neither the dominant WYSIWYG approach nor the

textual WYSIWYM approach are sufficient for simple users or modern computing environments. Mobile devices, touch interfaces and the modern web require reflowable content as well as a clear division between the document's structure and presentation. At the same time ordinary users need a simple and usable interface capable to fit in diverse hardware and complicated textual markup languages.

The document should conform to the reader's and not the writer's preferences regarding the visual layout. This problem exists even from antiquity, since the writer or the printer of the document shaped its visual layout regardless the needs and preferences of the reader. It's obvious that before the emergence of the computer, the reader couldn't modify the visual layout, but today with modern technology this is possible as well as essential. Unfortunately, the dominant word processing systems are still focused on paper and fixed layout formats, making a paradigm shift difficult. Even in web-based HTML editors, WYSIWYG inspired interfaces and usability are still dominant even though there is no correspondence to printed output nor to a overall web page visual layout. The only WYSIWYG aspect of the document is the correspondence with character formating, including bold, italics, font sizes and colors. In our approach the reader will define the character formating, even in the case of boldface or italics, since the writer will markup it up as emphasis, allowing the reader to choose the desired presentation. Character formating options will be reduced allowing the writer to focus on the authoring procedure and not to worry about formating. This concept is very popular in textual WYSIWYM tools like Latex and wikis, which reduce the formating freedom of the author. The writer, which is also a reader, will denote his visual preferences during the writing process. Our editor will force the author to provide structural semantics and at the same time will provide a visual representation of the document, implementing a hybrid approach similar to WYMeditor approach. Finally, our goal is provide other reading facilities, such as document annotations, bookmarks and notes, capable to assist the reading process. In the Table 2.1 we provide a summary of the main design guidelines for our HTML editor.

Application type	Client-side web-based HTML editor
Information storage	Remote and local
Development tools	HTML5, CSS3, Javascript
File format	XHTML with scientific notation and semantic annotation
Operating system support	All
Writing approach	Hybrid, similar to WYMeditor
Interface	Touch and WIMP
Devices supported	Mobile, tablet, desktop
Semantic annotation support	Yes
Reading support	Document annotation, notes, bookmarks, personal styling

Table 2.1: Writing applications blueprints

Chapter 3

Learning and writing

3.1 Cognitive science

Already from antiquity, early Greek philosophers attempted to unravel the mystery of the human mind. Plato and Aristotle in an effort to understand the nature of knowledge made early assumptions about thought and memory (Anderson, 2009, p. 4). The coming years, *philosophy* was the only investigating discipline of human cognition, proposing many theories about the body and mind nature and interconnection, consciousness, knowledge acquisition and free will. Particularly, the knowledge acquisition problem was a long and fierce philosophical debate that lasted for centuries. The problem referred to the contribution of biology versus experience to any human observable traits. Nativists proposed that a significant amount of knowledge is innate to humans and encoded in biological traits, while empiricists argued that people acquire knowledge only through interaction with the environment. John Locke the founder of empiricism, viewed the mind as a blank slate with no innate information, which experience eventually fill in (Friedenberg and Silverman, 2006, pp. 29-63). At the same time, physical sciences including biology, chemistry and physics were advancing remarkably using the scientific method as their core practice, instead of philosophical speculation. In contrary, human cognition research will not be a subject of scientific analysis until the end 19th century (Anderson, 2009, pp. 4-6).

The scientific study of the human mind and behavior started in 1879 by Wilhelm Wundt in his laboratory in Leipzig, Germany. *Psychology* was the new emerging scientific discipline of cognition, using the experimental method of introspection. The underlying concept was that self-observation could unravel cognitive functions and ultimately the origin of thought (Anderson, 2009, p. 6). Wundt was the founder of voluntarism one of the many schools

of thought that followed in the history of psychology. In the coming years, structuralism attempted to unveil the structural elements of the mind, while functionalism focused to the functions that the mind can preform. Later, the Gestalt movement introduced a holistic view of the mind contrary to the reductionist approach of the structuralists. At that time, Sigmund Freud founded the premises of the psychodynamic approach and psychoanalysis, trying to divide cognitive functions to distinct competing entities (Friedenberg and Silverman, 2006, pp. 65-91). The inner complexity of the human brain though, as well as the failure of introspection to deliver cognitive models gave rise to the the movement of behaviorism, which gradually dominated America by 1920. Behaviorists viewed the human brain as a black box and claimed that psychology should study exclusively the external behavior and stimuli and not inner cognitive functions. This perspective dominated America for forty years, until the 1960's (Anderson, 2009, pp. 7-8).

At that time, the advent of computers created a huge impact in the scientific community and especially in cognitive research. The new discipline of computer science and mostly the field of *Artificial Intelligence (AI)* demonstrated that information processing machines can solve problems as well as act intelligently. This development initiated a fresh view of the human mind as an modular information processor. Moreover in 1950's, Noam Chomsky's work in *linguistics* introduced new complexities in language research that the dominant behaviorist perspective could not sufficiently address. The result was that in this period from 1950 to 1970, which scholars refer to as cognitive revolution, the influence of behaviorism gradually declined giving birth to a new perspective in psychology. In 1967, Ulric Neisser published his book *Cognitive Psychology* introducing this new approach with emphasis in information processing and cognitive modules, including perception, attention, language, memory and thought (Anderson, 2009, pp. 9-10). The success of the cognitive approach is evident today, mainly because other subfields of psychology integrated and use cognitive theories in their domains, including social, developmental, clinical and educational psychology (Friedenberg and Silverman, 2006, p. 95).

In recent years, another approach to human cognition is advancing rapidly. *Neuroscience* started as a subfield of biology and studied the structure and function of the nervous system. Soon, the need to unveil the underlying neural mechanisms of cognitive processes, resulted the field of cognitive neuroscience. The methodology of cognitive neuroscience is fundamentally different in comparison to traditional cognitive research, mainly due to brain imaging techniques, the study of brain damages and electrical stimulation of the brain. Techniques including Computerized Axial Tomography (CAT), Positron Emission Tomography (PET), Magnetic Resonance Imaging (MRI) and functional Magnetic Resonance Imaging (fMRI) provide visual represen-

tations of the brain activity during cognitive tasks. This allows researchers to identify for every cognitive process the activated brain regions. Scientists also use this technique to study patients with brain damages, such as strokes and head traumas. Variations in patients' behavior and cognitive ability associate competencies and observable traits to involved brain regions. Others study intended brain lesions in animals with the same purpose. The last technique involves the electrical stimulation of the neurons of a specific brain region and the observation of the behavioral outcome (Friedenberg and Silverman, 2006, pp. 163-169).

Already from 1976, the Cognitive Science Society with the publication of the journal Cognitive Science founded a new field in human cognition research. Cognitive science is the interdisciplinary collaborative effort of philosophy, psychology, artificial intelligence, linguistics and neuroscience to provide answers regarding the human mind (Anderson, 2009, p. 10). During the evolution of cognitive research from philosophy to neuroscience, every major school of thought proposed a theory for human learning. Philosophical approaches including nativism and empiricism, and early psychological views such as structuralism and functionalism provided early models of human learning and the foundations of modern learning theories (Schunk, 2011, pp. 4-10). Schunk defines learning as "an enduring change in behavior, or in the capacity to behave in a given fashion, which results from practice or other forms of experience" (Schunk, 2011, p. 3), while Ormrod summarizes learning "as a long-term change in mental representations or associations as a result of experience" (Ormrod, 2011, p. 4).

The primary purpose of learning research is to describe how learning occurs in humans and animals, and secondarily how educators can utilize these findings in order to implement educational policies and improved methods of instruction. Historically, the fields of learning and instruction did not overlap, since they mainly employed people with different interests. Learning researchers were commonly psychologists, in contrast with instructors that usually had backgrounds in education. Today, learning research and educational practice are closely communicating, providing each other useful feedback for mutual revisions (Schunk, 2011, pp. 18-21).

Writing research is a relatively new domain in science, started in the early 1970's. In a short period of time, the study of writing attracted many scholars from various fields, creating an impressive literature. This field investigates the act of writing, including the underlying cognitive processes (Kellogg, 1999, pp. vii-viii) as well as a procedural analysis. In their pioneering work, Bereiter and Scardamalia define writing as "the composing of texts intended to be read by people not present", while they describe writing as both natural and problematic, in order to highlight its complex cognitive nature (Bereiter and Scardamalia, 1987, pp. 4-6). From another

perspective, Kellogg views writing as the communication of thought between people, through consensual symbols in a common basis of reference (Kellogg, 1999, pp. 9-10).

3.2 Behaviorism

The first learning theory to introduce derives from the *behaviorist* approach. Thorndike, Pavlov, Guthrie and Skinner were the leading researchers that proposed learning theories based on the behaviorist view. Their core ideas include that the internal cognitive processes were unnecessary to explain the human mind, as well as learning is the associative process of stimuli and responses. The most influential learning theory of the behaviorist school of thought was B.F. Skinner's theory of operant conditioning. According to this theory a desirable behavior to a initial stimulus is more likely to occur in the future, if the consequence of the behavior is reinforcing. Contrary, if the consequence is punishing, this behavior is less likely to occur in the future (Schunk, 2011, pp. 114-115).

Behaviorists also introduced concepts related to reinforcement and punishment, such as shaping, chaining, extinction and reinforcing schedules. The concept of shaping refers to the gradual modification of an existing behavior using reinforcement in a series of slightly different tasks (known as successive approximations), aiming to a desirable behavior. Educators use shaping, in order to create complex single step behaviors, contrary to complex sequential behaviors, in which they use chaining. In the chaining process, reinforcement occurs incrementally starting in one response, then two responses in a row, until the reinforcement of the whole sequence. Teachers use shaping and chaining in order to create new behaviors in students and extinction to eliminate problematic behaviors. The process of extinction refers to the gradual decrease of the frequency of a behavior that no longer leads to reinforcement. Desirable behaviors also extinguish, resulting educators to consistently reinforce these behaviors, either continuously or intermittently. In the case of intermittent reinforcement, behaviorists used reinforcement schedules to consistently reinforce either to create new or maintain existing behaviors (Ormrod, 2011, pp. 63-66).

Learning theorists and educators still use behaviorist principles, due to the positive results that are showing in the learning process, despite their obsolete theoretical foundation (Schunk, 2011, pp. 114-115). They employ techniques of reinforcement and punishment especially in classroom management, in order to transform undesirable behaviors. Moreover, many widely used instructional innovations have their roots in behaviorism, including applied behavior analysis, instructional objectives, programmed instruction,

computer-assisted instruction and mastery learning (Ormrod, 2011, p. 78).

1. *Reinforcement and punishment* are the foundations of the behaviorist view. According to B.F. Skinner one of the major issues in Western paradigm of education is that educators teach skills that will not be immediate useful, postponing their rewarding consequence in the future. The absence of an immediate reward forces teachers to resort to artificial reinforcers that awarded also inconsistently compared to the occurrence of the behavior, such as grades, free time, stickers or praise. In order to deal with this problem, Skinner suggested to incorporate punishments in educational practice, in order to threaten students for not learning. These punishments included grades reduction, criticism or even ridicule. Reinforcement and punishment rise many concerns regarding their application in educational practice. Reinforcement totally ignores the cognitive elements that interfering with learning, may encourage reward driven behaviors such as cheating, is counterproductive in complex and creative tasks, can undermine intrinsic rewards and doesn't prepare students not only for success but also for failure. Punishments suppress behaviors with no guaranty that these behaviors will not occur in the future or in the absence of the punisher. Moreover, punishment can have negative psychological effects on students, while it can improve behavior in a specific context and at the same time may create problems in another (Ormrod, 2011, pp. 78-92).

2. *Applied behavior analysis* is the systematic approach that intervenes a in problematic behavior. At first, analysts identify and quantify behaviors in observable measurable terms. Later, they identify conditions of the environment that could contribute to the problematic behavior and specify measurable target behaviors. Then, they form, implement and monitor a clear intervention plan, which they modify if necessary. Finally, if the intervention is successful usually they promote and use this solution in similar situations to others (Ormrod, 2011, pp. 92-99).

3. *Instructional objectives* refer to a description of skills and knowledge that students should acquire by the end of the teaching process. At first, behaviorists used the method of behavioral objectives, an early version of instructional objectives. According to this method, educators defined the measurable desirable behavior that students should perform at the end of instruction. They also defined the conditions under which the students should exhibit this behavior and a criterion for evaluating the performance of this behavior. Critics argued that behavioral objectives focus on lower-level instead of higher-level skills and are insufficient for more complex thinking and learning. Contrary,

instructional objectives when describe a more abstract form of educational goals could benefit both teachers and students. Teachers could become more efficient, since they can plan and easily communicate with others specific educational goals and their's approach effectiveness, while students can focus better on the most important educational material. Instructional objectives play a key role in educational practice, since educators use it extensively in curriculum design as well as in the evaluation of the instructional programs (Ormrod, 2011, pp. 99-104).

4. *Programmed Instruction (PI)* is a technique that addresses the timing inconsistency issue, between the time of the desirable behavior and the time of reinforcement. In order to deal with this problem, Skinner built a teaching machine using a box with a display window and a roll of paper printed with educational material. Learners viewed the material successively in discrete segments or frames, answering questions, viewing the answers and continue to the next frame. This method employs behavioral concepts, including active responding, shaping, immediate reinforcement and respects individual differences in learning pace. PI employed a linear program, which means that all students viewed the same frames in the same order. *Computer-Assisted Instruction (CAI)* uses a similar concept with PI, but its realized as computer software and employs a branching program instead of a linear. This means that advanced students can move on the material, while students with difficulties continue with remedial instructional frames. CAI also enriched the learning process by integrating animations, videos and images. Today, educators instead of CAI, they use terms such as Computer-Based Instruction (CBI) or Computer-Assisted Learning (CAL) incorporating simulations, intelligent tutors, tools and games (Ormrod, 2011, pp. 104-106).

5. *Mastery learning* is an instructional method in which learners first master one subject and only then proceed to the next. Behaviorists believed that learners can acquire complex behaviors in a suitable environment given adequate time and support. Shaping is the basic technique of mastery learning that usually includes a sequence of small discrete units with gradual increase of difficulty. When learners complete a unit, they demonstrate their mastery in a solid measurable criterion, such as test grades. Failing students engage in further remedial activities in their own pace, while passing students working in enrichment activities or even tutoring students. Researchers concluded that mastery learning is more efficient than traditional instruction, often leads to higher achievement and learners can retain information for a longer period of time. This is true though, if the educational

objective is to learn specific skills and not for analyzing and complex problem solving (Ormrod, 2011, pp. 106-109).

Besides the instructional innovations and applications of the behaviorist approach, our focus is essentially in the process of writing production and instruction. In this domain, the behaviorist approach totally ignored the structural elements of the writing procedure, as well as the social aspects of the author's environment, emphasizing mainly to the final product. Writing instruction was focusing separately in vocabulary, spelling and grammar, while educators considered evident and automatic the final synthesis of those skills in the form of a general writing skill. The role of instructors was to evaluate the final text mainly in terms of grammatical and spelling mistakes, while their involvement in the process of writing was nonexistent. The instructor was typically the only audience of the writing content and usually use it as an examination technique. Many researchers criticized the behavioral approach in writing, mainly due to the total detachment from cognitive and social elements as well as the inefficiency in the instructional process (Spantidakis, 2010, pp. 22-24).

3.3 Social cognitive theory

In 1950's, many scholars started to challenge the foundations of the behaviorist approach as well as the derivative conditioning theories of learning. Albert Bandura began his research in the influences of social behavior, back in the early 1950's. He believed that conditioning theories didn't provide a solid theoretical ground capable to explain deviant and social behaviors. The main idea of his research was that humans could acquire new behaviors, by simply observing others to preform them. In this learning process, observers didn't have to preform the behaviors themselves, while reinforcement was irrelevant. *Social cognitive theory* challenged the key assumptions of the behaviorist approach including reinforcement and the learner's active involvement, stressing the belief that learning also occurs in the social environment. Bandura proposed that people could acquire knowledge and skills, as well as beliefs and attitudes, by merely observing social affairs (Schunk, 2011, pp. 118-119).

Social cognitive theory incorporates concepts including models, modeling and motivation. The concept of a model in social cognitive theory mainly refers to a certain behavior or skill, while modeling is the changing process of behaviors, cognition or emotions, though the observation and imitation of different models. Bandura identified three main functions in the modeling process, such as response facilitation, inhibition/disinhibition and ob-

servational learning. Schunk defines that "response facilitation refers to the modeled actions that serve as social prompts for observers to behave accordingly". Inhibition occurs when a model exhibits punishing consequences that leads the observers not to preform similarly, while disinhibition involves the acceptance of prohibited behaviors, due to the absence of negative effects. Finally, observational learning through modeling is the change in behavior that was unlikely to occur without it, even in the case of a highly motivated observer. Observational learning requires the observer's attention in order to perceive the model, retention of the model's symbolic form in memory, production of the behavior based on the personal mental representation, and motivation, which is the calculation of the model's importance based on rewarding consequences (Schunk, 2011, pp. 123-145)

Social cognitive theory distinguishes motivation of the learner in core structural elements, including self-efficacy, goal setting, outcome expectations and values. Self-efficacy or efficacy expectations refers to personal confidence in learning and preforming competencies, in certain activities. This means that a person with high levels of self-efficacy is aware of what is capable of doing. These people usually have a vibrant involvement in learning activities, in contrast with students with low self-efficacy that may avoid educational tasks (Schunk, 2011, pp. 146-147). Goals are the objectives of the learning procedure in terms of performance, and a key concept to preserve motivation for long periods of time without external intervention. At first, the commitment to certain goals is essential, as well as the subsequent involvement in a learning task. Along the way, students compare their performance to a initial goal, raising self-efficacy levels and sustain motivation, if these evaluations are positive. In addition, social cognitive theory examines the motivation of the learner in comparison with the consequences of the model. Learners use their experiences and observations to evaluate the possible outcomes, retaining afterwards long-term engagement in the learning process, in order to reproduce these desired consequences. Learners calculate the value of a model, based on their personal perception about its importance and usefulness, contributing also to motivation. People usually learn behaviors that exhibit a consistency with these values (Schunk, 2011, pp. 138-145).

Important instructional applications deriving from the principles of social cognitive theory include techniques, such as models, self-efficacy, worked examples, as well as tutoring and mentoring.

1. *Models* refer to behaviors that students observe and imitate. Teacher models are important not only for their facilitation of the learning process, but also for the improvement of students' self-efficacy. Teachers that explain concepts and skills, provide student's self-efficacy infor-

mation and encourage students to engage in learning activities have increased odds to positively influence them. In addition, its very important for teachers to ensure the consistency between words and actions. Research shows that students find hard to follow their instructions otherwise. Accordingly, peer models can also exhibit positive effects in observing students, mainly when using many of them that correspond to different levels of skill (Schunk, 2011, p. 157).

2. *Self-efficacy* is a very important aspect of the learning process. Apart from teaching knowledge and skills, the role of teachers relies also in building self-efficacy on students. Although, students require teacher's assistance in the learning process, periods of independent student practice are also essential for improving self-efficacy. Researchers highlight the importance of self-efficacy in student development and in academic achievement (Schunk, 2011, p. 157).

3. *Worked examples* are graphical and procedural illustrations of a step by step problem solving scenario. Instructors create models that represent the approach of an experienced problem solver in a particular issue. This demonstration incorporates the explanatory narration of the instructor, in order to clear the puzzling aspects and simplify the problem. At first, learners study the worked examples trying to identify strategies and skills that apply to a wider range of problems, and then attempt to solve similar problems themselves. Educators use this method extensively in science and mathematics, as well as in interactive computer-based learning environments (Schunk, 2011, p. 158).

4. *Tutoring and mentoring* are instructional methods based on the principles of social cognitive theory. Tutoring typically involves the instruction of a single person in a specific subject, while mentoring refers to the instruction of many in a educational environment. Tutoring generally is more apprenticeship-oriented focused on content instruction, while mentoring involves guidance and facilitation of the learning process. Research found that the method of mentoring leads to educational and motivational advantages in both mentors and students (Schunk, 2011, pp. 158-159).

Evidence of empirical research supports social cognitive theory in a variety of contexts, participants and settings. The initial acquisition of an early approximate skill based on model observation, combined with subsequent practice and remedial feedback from an instructor could improve learning as well as motivation (Schunk, 2011, pp. 160-161). In the writing instruction domain, these principles of social cognitive theory provide a theoretical ground for the ancient instructional approach of mentor texts. Students read

a text from an experienced author, analyze its meaning with guidance form the instructor and work on vocabulary and grammar exercises. Afterwards, the instructor requests an essay with a similar subject encouraging students to imitate and use elements of the mentor text in their writings. Using this method, students could unveil structural and grammatical elements as well as writing styles, advancing their writing skills, if they could avoiding plain imitation (Spantidakis, 2010, pp. 22-24)

3.4 Cognitivism

The cognitive revolution in 1950's and 1960's gradually dethroned the prevailing behaviorist approach. Cognitive psychology became the dominant theoretical perspective in the study of human mind with great influence in learning theories. Psychologists and learning researchers all over the world increasingly incorporated cognitive elements in their theories, and by 1970's the majority of them had accepted and expanded the cognitive perspective. The most important precursors of cognitivism were Edward Tolman's theory of purposive behaviorism, Gestalt psychology and the field of verbal learning (Ormrod, 2011, p. 141). Although, cognitive researchers often disagree over the boundaries and divisions of different cognitive approaches, contemporary cognitivism includes a wide range of theories, including information processing theory, constructivism and emerging contextual theories. The dominant cognitive approach in learning is a group of theories known as *information processing theory*, an attempt to examine the underline cognitive processes involved in the processing of information by humans (Ormrod, 2011, pp. 152-156).

The prevailing model in information processing theory is the two-store memory model. This model explains the cognitive processes of perception, attention, short-term memory as well as functions of long-term memory, including storage, retrieval and forgetting. According to this model, a stimulus input triggers one or more of the human senses, activating the relevant sensory register. This register retains the input for an instance in a sensory form, while the system initiates the procedure that assigns meaning to the trigger. This associative process of stimulus to prior familiar information is known as perception. The next step involves, the transmission of information from the sensory register to the short-term memory, which is a working memory with limited storage capacity and duration. Then, the permanent memory, which is the long-term memory, retrieves and integrates information to the working memory, in order to recognize already known patterns (Schunk, 2011, pp. 165-168). Control processes coordinate the procedure and the information streams throughout the system, allowing the encoding and storage of

information in long-term memory. This overall procedure applies to a single and not multiple input stimuli. In the case of multiple instant stimuli, the process of attention acts as a filter and bounds the number of inputs that the system could simultaneously handle. This means that humans could process only a fraction of the triggering information at the same time (Schunk, 2011, pp. 224-226).

Educators increasingly apply the deriving principles of information processing theory in educational settings. Instructional applications using the cognitive approach, include advance organizers, conditions of learning, and cognitive load.

1. *Advance organizers* is a instructional technique based on the assumption of hierarchical organization of knowledge. According to this method, instructors provide a general description of the main ideas of the material, in order to connect new knowledge with prior information. Teachers provide broad statements in the beginning of the instructional process, directing the attention of learners on primary concepts and relationships. Advance organizers could be expository or comparative. Expository organizers introduce new useful information for the lesson comprehension, including concept definitions and generalizations of general principles, while comparative organizers focus on drawing analogies with prior relevant knowledge. Organizers could also be graphical maps with explanatory text for concepts and relations. According to research, the use of organizers promotes learning, while in the map form they could easily integrate with educational technology (Schunk, 2011, pp. 218-219).

2. *Conditions of learning* refer to the settings, in which learning takes place. In this theory, Robert Gagné identified the types of learning outcomes and the conditions that promote instruction. Learning outcomes include intellectual skills, verbal information, cognitive strategies, motor skills, and attitudes. Intellectual skills refer to procedural knowledge that includes concepts, rules and processes, while verbal information is declarative knowledge and incorporates facts and events. Cognitive strategies are control processes, ranging from attention to problem solving, while motor skills refer to physical movement competencies of the body. Finally, attitudes involve inner views that reflect personality characteristics, such as fairness and honesty. Instructors can arrange the external conditions that could facilitate learning, since they represent a supportive learning environment, in contrast with internal conditions that represent the cognitive processing capabilities of the student and stored knowledge in the long-term memory (Schunk, 2011, pp. 219-221)

3. *Cognitive load* refers to the processing effort of the working memory regarding a task. Similarly to the limitations of attention, the processing capacity of working memory is also narrow, due to processing times and the involvement of multiple cognitive processes. Cognitive load theory informs instruction designers about these limitations, in order to avoid cognitive overload of students in learning activities. Cognitive load could either be intrinsic or extrinsic. Intrinsic cognitive load refers to the invariable inner complexities and cognitive demands of the learning material, while extrinsic involves the quality of the instructor's presentation and the required learning tasks (Schunk, 2011, pp. 223-224).

As the cognitive approach revolutionized the perception of the human mind, the fields of learning and writing started to transform. In early 1970's, the pioneering works of Emig, Graves and Elbow essentially founded the domain of writing research and the procedural approach in writing production. Emig investigated the procedures that students used during their engagement in writing extended texts, while Graves examined students' writing subjects, styles, purposes and processes, reaching to useful conclusions at the time. Elbow on the other hand, based on personal experience and the study of professional authors, proposed that writing is the repetitive two-stage process of editing and revising. He argued that these two stages should not interfere, allowing the author to write spontaneously without considerations, until the revising stage. Elbow's perspective initiated a trend in writing research, which is known today as *free writing*. Common beliefs of early writing research was the absence of ready solutions in the writing process, as well as the author's difficulty to conclude to the final content. The majority of writers do not hold a clear view on the final content, rather they constantly restructure their knowledge, in order to reach to a specific perspective (Spantidakis, 2010, pp. 34-36).

During the 1980's, writing researchers started to incorporate cognitive elements in their theories, as well as to propose early models of writing production. The most prominent writing production model of the cognitive era was the model of Flower and Hayes, published in 1981 (Flower and Hayes, 1981) (Spantidakis, 2010, pp. 37-38). According to this model, writing production and comprehension is a problem-solving procedure, consisting of the writing processes of planning, translating, reviewing and monitoring (Figure 3.1)[1] (Spantidakis, 2010, pp. 43-44).

During planning the author constructs an internal representation of knowledge, by generating and organizing ideas and by setting writing goals. The author recalls all the relevant information from the long-term memory, re-

[1] "http://commons.wikimedia.org/wiki/File:FlowerHayesModel.png"

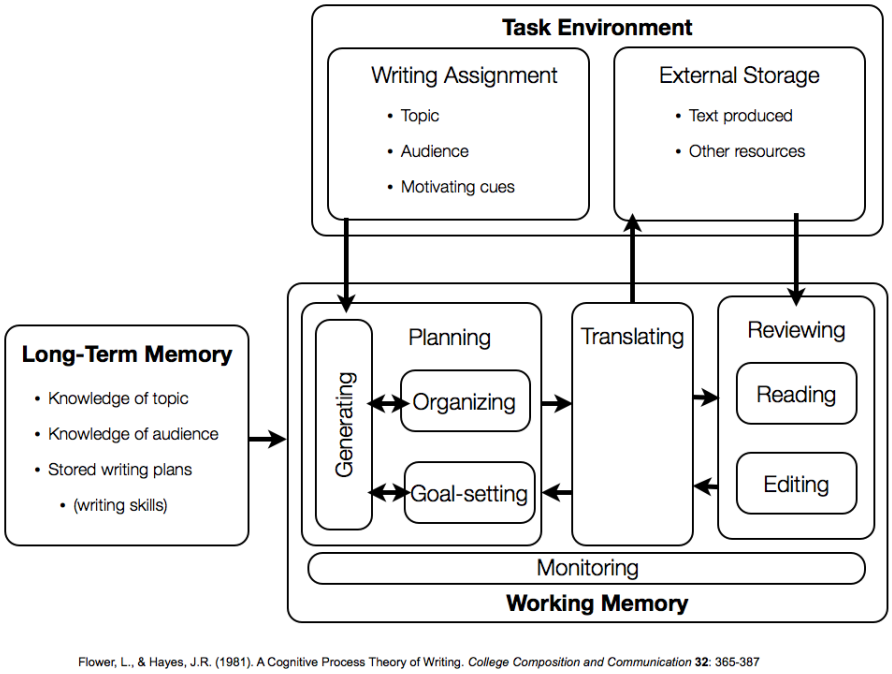

Figure 3.1: Structure of the cognitive model of Flower and Hayes

garding the writing assignment's topic, audience and objectives, and organizes them in a meaningful structure. The organizing process is consistent with the writing goals, that the writer sets, in order to communicate with a specific audience. The process of translating transforms the writing plan into visible language, creating text. During reviewing the author reads, evaluates and revises the produced text, in order to meet the writing goals. Monitoring regulate the transition between processes and evaluate the progress of every single process (Flower and Hayes, 1981, pp. 369-375). Contributions of the Flower and Hayes model include a new perception of writing as a dynamical, complex, multi-procedural, problem-solving activity, in which the author plans, sets goals, organizes, writes and revises during the writing production (Spantidakis, 2010, p. 46). Despite the model's significance, critics stressed the distancing of the model from the sociocultural environment of the author, arguing for the inclusion of the social context in writing production. Moreover, they questioned the validity of the model regarding the unexperienced writers, contenting that the model describes mostly the processes of experienced authors. These considerations gradually altered the procedural nature of writing production models, including developmental and socio-cognitive elements (Spantidakis, 2010, p. 38).

3.5 Constructivism

The rising of the constructivist movement in philosophy and psychology derived new approaches in learning theories and writing research. Jean Piaget's *cognitive development* research and the *sociocultural* approach of Lev Vygotsky introduced theories about the construction of human knowledge (Schunk, 2011, p. 229). Piaget argued that children individually construct complex representations of knowledge and expand their cognitive capacities with age, interacting with the physical and social environment. On the other hand, Lev Vygotsky stressed the significance of social affairs and cultural contexts in children's development, founding the premises of a sociocultural perspective in learning and psychology (Ormrod, 2011, pp. 289).

Piaget identified four critical elements regarding cognitive development, including biology, interaction with the physical and social worlds and equilibration. Equilibration is a central concept in this theory, representing an internal mechanism that resolves cognitive conflicts namely inconsistencies between cognitive representations and environmental observations. When a cognitive conflict occurs the process of equilibration triggers either the component process of assimilation or accommodation, in order to maintain balance. Assimilation refers to the inclusion of a new environmental information to the existing cognitive structure, while accommodation involves

the restructure of internal representations to adapt to observable reality. This procedure represents the core learning mechanism of the theory of cognitive development. Another main idea is the division of children's cognitive development in stages. Piaget identified behavioral patterns involving the children's abilities in relation to age, in order to form his cognitive development stages. These stages include sensorimotor, preoperational, concrete operational and formal operational, accordingly to ages from birth to two, two to seven, seven to eleven and eleven to adulthood (Schunk, 2011, pp. 236-240).

Sociocultural theory emphasizes the role of social interaction and cultural-historical aspects in human development. Vygotsky claimed that interpersonal relations between people facilitate cognitive growth, under the critical influence of a cultural-historical context. He also acknowledged the importance of biological factors in cognitive development, resulting his interest in disabled children (Schunk, 2011, pp. 242-243). Sociocultural theory supports the existence of higher cognitive functions in humans, which people promote with social interaction as well as physical and cognitive tools in a process known as internalization. Physical tools are cultural innovations, such as computers, knifes or cars, while cognitive tools include symbolic or mental entities, such as writing systems or language. Internalization is process in which social experience transforms to inner cognitive structure and promotes learning and development. Learners often adapt their knowledge to fit their own needs in a process known as appropriation. Another core idea in sociocultural theory is that challenging activities improve cognitive development. According to Vygotsky, challenging tasks exist in the the zone of proximal development (ZPD), which Ormrod defines as "the range of tasks that children can't yet perform independently but can perform with the help and guidance of others". When children master learning tasks, they accordingly change their ZPD, in order to engage in more challenging activities in the future. Vygotsky also highlighted the role of play in cognitive development, arguing that it represents a important foundation for the transition from childhood to the adult world (Ormrod, 2011, pp. 313-318).

Many sociocultural theories derived from the Vygotsky's approach including activity theory and situated cognition. Activity theory describes how the actions of individuals or groups relate to the environment, in which they occur. According to this theory actions and environment consist a single entity known as activity system, which represents the unit of analysis in this theory. This system is interconnected and incorporates the people involved (subjects), their goals (objects/motives) and tools, the social relations between them (community), the system's conventions (rules/norms) and the different roles of the participants (division of labor). The last concept of activity systems is outcome, which refers to the changes that the system

produces (Spantidakis, 2010, pp. 116-121). Situated cognition or situated learning is a constructivist theory, which emphasizes the importance of the relations between people and situations, in learning and development as well as belief systems and knowledge. This approach contrasts with information processing theory, which stresses the importance of cognitive processes, in order to explore cognitive development in specific domains, such as science, literacy and mathematics (Schunk, 2011, p. 233).

The approach of constructivism to instruction design involves the creation of educational environments according to constructivist principles. The core idea is to provide inspiring experiences that promote learning. Instructors usually challenge learners with problem-solving activities, highlighting the primary concepts of the subject. They regard students' perspectives in instruction planning and adapt the curriculum, in order to reduce the gap between the students' beliefs and the perspective of the subject. Student assessment takes place during instruction, involving the teacher's observations and the student's work and understanding. The American Psychological Association proposed a set of constructivist, learner-centered guidelines for educational planning, that highlight the importance of the approach. Deriving instructional applications include discovery and peer-assisted learning, discussions and debates, inquiry and reflective teaching, as well as cognitive apprenticeship (Schunk, 2011, pp. 261-265).

1. *Discovery learning* focus on the formulation and evaluation of hypotheses based on collected information. During this problem-solving procedure, students investigate, synthesize and evaluate, in order to transform observations to general rules and principles. The involvement of the instructor is minimal, and includes the formulation of a discovery situation, as well as discrete student guidance. Critics argued that this method exhibits inferior results to guided instruction, resulting the emergence of a hybrid approach featuring guidance and discovery learning known as guided discovery (Schunk, 2011, pp. 266-268).

2. *Inquiry teaching* refers to the dialectical approach similar to discovery learning, based on the Socratic method. The instructor discusses with a student by posing repetitive questions, challenging the student to reason, formulate principles and apply them in relevant cases. Teachers employ this method in individual tutoring as well as in small groups (Schunk, 2011, pp. 266-268).

3. *Peer-assisted learning* is a general division of instructional methods that utilize learner peers for educational purposes. These methods include peer tutoring, reciprocal teaching and cooperative learning. Peer-tutoring promotes class participation from the tutor as well as the tutee, facilitates questions and cooperation, that could lead to

better results in comparison to traditional instruction. Moreover, the main idea of cooperative learning involves the engagement of a small group in completing a learning task, in order to promote cooperation between students (Schunk, 2011, pp. 269-271). Finally, reciprocal teaching requires an initial modeling of the learning activities, and later the sequential switch of the instructor's role between the teacher and a small group of students (Schunk, 2011, p. 246).

4. *Discussions and debates* refer to active classroom dialogue between students, highlighting varying perspectives of a topic. Usually, the instructor is the moderator and the promoter of the discussion, while the subject typically involves complex and controversial elements. On the other hand, debates involve student groups that disagree over a topic, presenting arguments to promote their points of view. Debates are not spontaneous and require preparation and group collaboration (Schunk, 2011, p. 271).

5. *Reflective teaching* is a method that considers students' backgrounds, motivation, and cognitive processes, as well as learning contexts and the instructor's self awareness, in instructional planning. The main idea of reflective teaching is that there is no single instructional approach suitable for every student. Therefore, instructional design should be sensitive in contextual changes as well as constantly adapt to the new context in a process known as fluid planning. This process requires active teachers with adequate personal knowledge and problem solving skills, capable of observing, evaluating and planning, in order to produce instructional decisions that fit in the contextual variations of the learning process (Schunk, 2011, pp. 271-274).

6. *Cognitive apprenticeship* is a instructional method that reflects the principles of situated learning and attempts to synthesize traditional apprenticeship and schooling. Traditional apprenticeship consists of the phases of modeling, scaffolding, fading and coaching. During modeling, an apprentice observes the demonstration of an expert preforming a task, in visible discrete stages. Scaffolding comes next and refers to the expert's support, adjustments and corrections, in the early preforming attempts of the apprentice. The gradual reduction of this support and the progressive independence of the apprentice is the process of fading. Coaching is evident during the whole process of apprenticeship involving monitoring, evaluating, supporting and encouraging the learning procedure. The problem with the educational application of traditional apprenticeship is that in traditional practices tasks are visible with evident significance and a clear matching with a skill, since they situate mainly in the workspace. In contrast, educational skills are not visible, with unknown significance, that students should

> employ in a variety of situations. The challenge of cognitive apprenticeship is to inform educators, in order to make mental tasks visible to students, to highlight their significance in authentic contexts, as well as to generalize the use skills mainly in novel situations (Collins et al., 1991).

Despite the significance of constructivism in learning and instruction, the implementation of constructivist principles in real educational environments is rather puzzling. School systems, standardized curricula and testing, unprepared teachers and students, even non supportive parents, create obstacles to the incorporation of constructivism in the classroom. Besides problems, educators today value the premises of this approach for learner-center instruction, deep concept perception and learning through experience, attempting to incorporate even more constructivist principles in the learning process (Schunk, 2011, pp. 265-266).

The significance of constructivism is evident also in writing research. Criticism to the early cognitive model of Flower and Hayes, for not including social influences and processes of unexperienced authors, resulted new approaches including developmental and socio-cognitive writing models, as well as sociocultural theories of writing instruction. In 1987, Bereiter and Scardamalia introduced the writing models of *knowledge telling* and *knowledge transforming*, in an effort to describe the writing processes of immature and proficient authors respectively. The primary difficulties of unexperienced writers, typically regard the writing code, as well as the generation of the writing content. The content production of oral language is radically different from written language, since speech involves the contribution of a conversation partner in "thinking of what to say, in staying on topic, in producing an intelligible whole, and in making choices appropriate to an audience not immediately present". According to the knowledge telling model, beginning writers initially identify the topic of the writing assignment and the literary form. These identifiers act as cues in information retrieval from the long-term memory, without ensuring thought relevance or consistency to a writing plan. In the next step, writers test if the retrieved knowledge is appropriate for inclusion to the content, based on simplistic rules. If so, they write down the mental representation of the content in notes or text, initiating the next cycle of content production (Bereiter and Scardamalia, 1987, pp. 6-10).

On the other hand, the knowledge transforming model describes the writing processes of proficient authors. According to this model, problem analysis and goal-setting involve the content and rhetorical problem spaces. The first deals with content generation issues by retrieving and processing knowledge, as well as the author's beliefs and worldview. The second faces goal

achieving problems by evaluating and fitting the composition in terms of the target audience, grammar, textual coherence and meaning. These problem spaces interconnect with each other, since emerging issues often regard both problem spaces. When the user reaches to a final mental representation of the content continues in the knowledge telling procedure, in order to produce text (Bereiter and Scardamalia, 1987, pp. 10-12) (Spantidakis, 2010, pp. 48-49). Bereiter and Scardamalia also introduced the significance of *procedural facilitation* in writing production. They defined them as "ways to ease the executive burden of writing in some particular respect, without providing any substantive help such as suggestions of content or form" (Bereiter and Scardamalia, 1987, pp. 10-12). Research shows that writing in a supportive and encouraging environment with procedural facilitators helps unexperienced writers to develop their skills and writing abilities. Writing researchers characterized the models of Bereiter and Scardamalia as developmental, because they describe the involving writing processes of beginning writers as they become proficient (Spantidakis, 2010, p. 51).

In the 1990's, Flower and Hayes proposed two different socio-cognitive writing models, in order to respond to 1981 critics. Flower introduced the construction of negotiated meaning model, in 1994. According to this model, writing production is a meaning construction procedure, between the writer's and the reader's mental representations. This interaction occurs in a context, involving sociocultural issues, language and discourse conventions. This model stressed the significance of the author's involvement, in writing production and reading comprehension, as a promoter of the writing process (Spantidakis, 2010, pp. 51-53). In 1996, Hayes also proposed a socio-cognitive writing model featuring two primary parts, the task environment and the individual (Figure 3.2)[2]. The task environment includes the social environment consisting of the audience and the collaborators of the text, as well as the physical environment which incorporates the text so far and composing medium. The individual involves motivation and emotions, the working memory, the cognitive processes of writing production as well as the long-term memory. The working memory is divided in a phonological memory, a visuospatial sketchpad and semantic memory, while the long-term memory involves knowledge about the topic, the audience, the language and the literary genre. Hayes replaced the process of planning with reflection, reviewing with text interpretation and translation with text production (Spantidakis, 2010, pp. 53-56).

Sociocultural theories investigated the writing process in terms of writing production and instruction. Although they didn't introduced procedural models of writing, they stressed the importance of cognitive apprenticeship,

[2] http://www.louterpromoveren.nl/schrijven/free-writing-gratis-en-dus-goed/attachment/hayes/

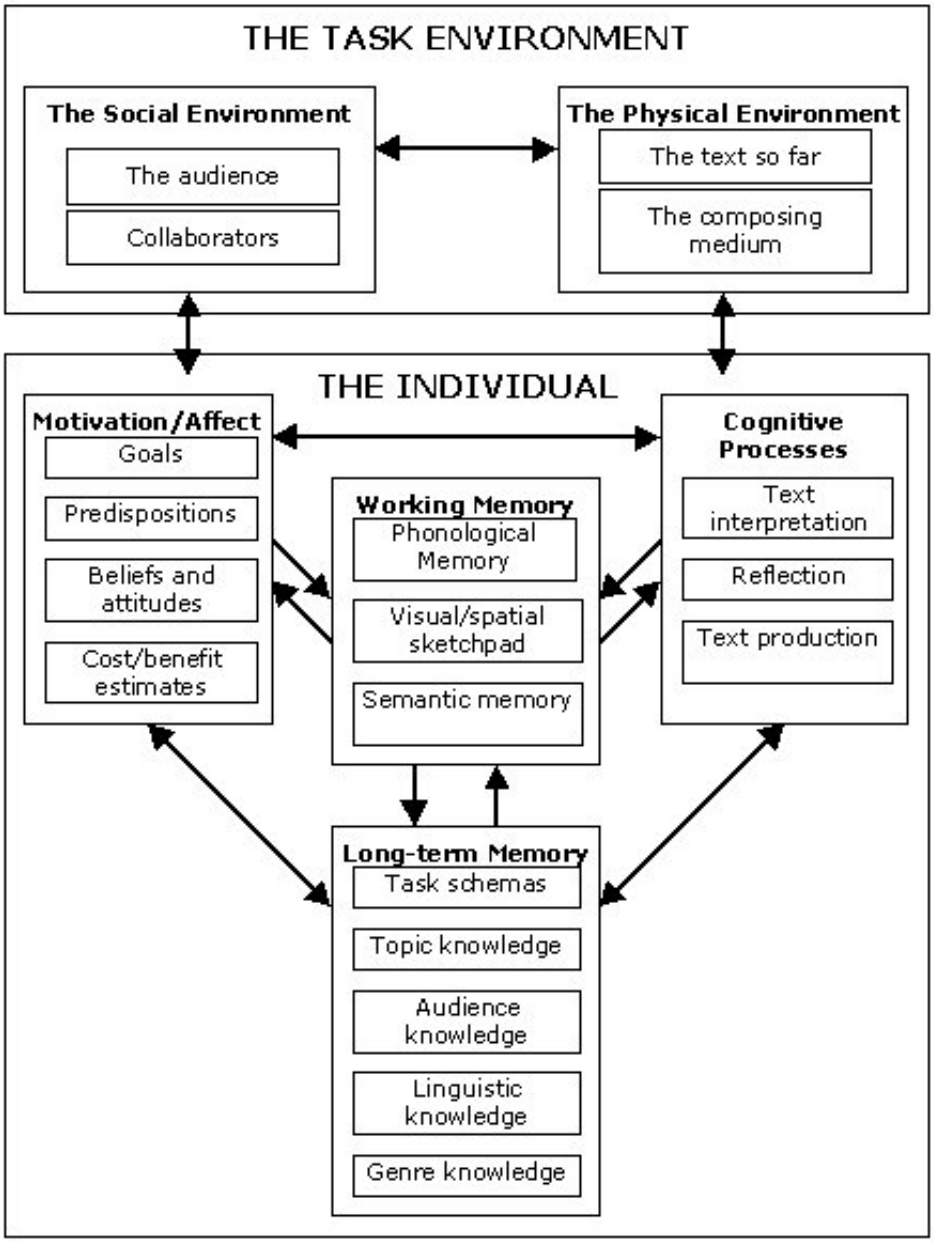

Figure 3.2: Hayes's framework for understanding cognition and affect in writing.

procedural facilitators and tools, as well as communities of practice in the teaching of writing. The implementation of cognitive apprenticeship for the development of writing competence, requires the active involvement and guidance of the instructor in every phase of the writing process, including content generation, structure planning, goal-setting and reviewing. The role of the instructor is to provide a cooperative environment, in which students could develop strategies, define the structural elements and literate genres, as well as compose and review texts. The involvement of the instructor gradually fades as the students become more competent. Research reveals that the method of cognitive apprenticeship in writing instruction shows positive results in the development of writing abilities of novice authors (Spantidakis, 2010, pp. 126-130).

On the other hand, procedural facilitators promote the processes of planning, monitoring and revising, by reminding strategies, procedural stages and tools in the writing process. They acting as cues, contributing to the reduction of the cognitive load of the author, in order to to ease the writing process. Moreover the involvement of cultural tools can promote writing. These tools refer either to physical, cognitive or linguistic elements, including writing systems, diagrams, writing instruments, graphic organizers, grammar and spell checkers, text structures, or any other device capable to promote the writing procedure (MacArthur et al., 2008, p. 211). The third pedagogical principle for the instruction of writing refers to communities of practice. Interaction and socialization in these literacy communities, facilitates linguistic skills such as speaking, reading and writing, while allows valuable feedback from peers and instructors. Communities of practice allow participants to spread their knowledge, express confusions and doubts, while they share values, beliefs, genres and social conventions. Research shows that participation in these communities allow members to regulate their own learning, while they promote higher order thinking (MacArthur et al., 2008, p. 214).

3.6 Contemporary approaches

Modern day developments in complex forms of learning and instruction, include the study of cognitive learning processes, motivation, self-regulation strategies, educational technology and neuroscience of learning (Schunk, 2011). Although researchers debate on which cognitive processes contribute to learning, instructors generally recognize the significance of conditional knowledge and metacognition, concept learning, problem solving and transfer (Schunk, 2011, p. 279). Scholars typically divide human knowledge in declarative, representing concepts, facts and events, as well as in procedural

for rules and processes. A third type is conditional knowledge that represents the competency to decide the situation and the reason, in which someone uses declarative and procedural skills. Conditional knowledge allows students to combine knowledge with procedures, in accomplishing educational tasks. A core concept in this process is metacognition or the intentional control of the mental processes. Metacognition is a process of "cognition about cognition" and involves knowledge and monitoring activities. At first, learners analyze which skills, resources and strategies should employ in performing a task, and secondly, they use metacognitive monitoring processes, in order to ensure a successful result (Schunk, 2011, pp. 284-287).

Another cognitive learning process is concept learning. In every educational setting or everyday life, people confront concepts. Concepts refer to cognitive representations of groups, involving physical objects or abstractions with similar attributes. The process of concept learning involves the creation of mental representations capable of grouping objects and abstractions by identifying similarities. Prevailing theories suggest that people classify concepts based on if-then rules, or on generalized images that define critical attributes, known as prototypes (Schunk, 2011, pp. 292-294). Problem solving is also another crucial cognitive learning process, with wide range of application in various domains, but mostly in science and mathematics. Schunk defines problem solving as "people's efforts to achieve a goal for which they do not have an automatic solution". This procedure involves the problem solver's expertise also known as initial state, states representing a final goal or including subgoals, and the cognitive operations that modify these states and eventually solve the problem. Typical problem solving strategies include generate and test methods, brainstorming, monitoring and reducing the differences between the current state and the final goal, and drawing analogies with familiar problematic situations (Schunk, 2011, pp. 299-317). The last strategy refers to analogical reasoning competencies and reflects the processes of transfer. According to Schunk, "Transfer refers to knowledge being applied in new ways, in new situations, or in familiar situations with different content". Transfer involves the activation of the memory and cross-referencing mechanisms, in order to link prior knowledge with new situations, and plays a key role in instruction (Schunk, 2011, pp. 317-322).

Apart from the cognitive learning processes, motivation and self-regulation appear to be very important to learning. Motivation is the process, which incites and maintains actions and behaviors that demonstrate goal orientation. Many theories stressed various ideas about motivation, ranging from goal-setting and fulfilling expectations to the need of achievement and control. Motivational strategies are critical for instruction, since they promote the active involvement of students in educational settings and activities (Schunk, 2011, p. 397). Self-regulation refers to the coordinative processes that con-

trol cognition, actions and emotions for goal-accomplishing purposes. In the learning process, useful self-regulatory behaviors include goal-setting and evaluating progress, expecting desirable outcomes, monitoring and adapting strategies, as well as maintain positive emotions. The role of instructors is to demonstrate self-regulation skills and strategies, in order to promote self-regulation in students (Schunk, 2011, pp. 441-443).

Advancements in information technology have radically transformed instruction as well as educational technology. In recent years, new approaches, including computer-based learning environments and distant learning applications introduced new challenges to learning researchers, in order to unveil the ways technology affects cognitive processes. Recent research shows that technology assists exploration and construction of knowledge, provides a learning-by-doing context, supports social interaction, and promote learning by reflecting. Computer-based learning environments include a variety of instructional applications, such as computer-based instruction (CBI), simulations and games, multimedia, distant learning and e-learning (Schunk, 2011, pp. 324-325).

1. *Computer-based instruction (CBI)* is a method for demonstrating information, prompting students to engage in learning exercises, and providing feedback to students' answers. Despite limitations, research shows that CBI draws students' attention in educational tasks, record learning performance, and promotes personalization of the content and rate of learning (Schunk, 2011, pp. 324-325).

2. *Simulations* refer to real or fictional visual environments that students could explore. This approach represents the ideal computer-based environment for discovery and inquiry learning. Researchers suggest that teaching with simulations are more effective than traditional approaches, inciting advanced cognitive processing and promoting problem solving skills. Instructors also use *games* to create pleasant educational settings for advancing reasoning and problem solving skills, as well as content instruction. Empirical evidence appear great results in students' motivation, and overall in the learning process, especially when the leaning material interconnects with games and simulations (Schunk, 2011, p. 326).

3. *Multimedia* refer to different forms of information, including audio, video, images, text, animation, and their combinations. Research indicates that teaching with multimedia promotes transfer and problem-solving abilities, primary for "students with little prior knowledge and high spatial ability" (Schunk, 2011, pp. 326-327). Advantages of multimedia use include, the presentation of information with multiple ways,

the active involvement of students, personalization of learning, repetition of material, as well as adaptation in different learning and teaching styles (Spantidakis, 2010, pp. 156-166).

4. *Distant learning* refers to the long-range delivery of instruction in remote places. Technological innovations, such as video conferencing and chats promote real-time or synchronous communication with an instructor, while viewing content, posting messages and commenting are means of asynchronous interaction. Research indicates that the synchronous approach was slightly less effective than traditional instruction, while findings for asynchronous in comparison to face-to-face instruction were controversial and misleading (Schunk, 2011, pp. 328-330). Educators materialize distant learning applications with e-learning technology.

5. *e-Learning* refers to "instruction delivered on a digital device such as a computer or mobile device that is intended to support learning" (Clark and Mayer, 2011, p. 8). e-Learning could facilitate traditional instruction or be tailored to distant learning applications. Although, researchers and educators often use the term e-learning with diverse meanings, e-learning typically involves the transmission of a lesson through a media channel, use of multimedia, synchronous and asynchronous approaches, and relevant content to learning objectives (Clark and Mayer, 2011, p. 8-11).

The invasion of multimedia and information technology in learning, draw the attention of researchers, delivering many theories about the implications of technology in the learning process and writing production. Mayer's cognitive theory of multimedia learning investigates the ways that words and pictures affect learning. The key assumptions of his theory involve dual channels for auditory and visual information processing and their limited processing capacity, as well as active processing which means, "attending to relevant incoming information, organizing selected information into coherent mental representations, and integrating mental representations with other knowledge" (Mayer, 2005, p. 31-48). Apart from Mayer's research these assumptions derive from Paivio's dual-coding theory, Sweller's theory of cognitive load, and Baddeley's model of working memory. Mayer also suggested design principles for educational environments, based on the premises of his theory (Spantidakis, 2010, pp. 227). We present Mayer's principles in the design methodology chapter.

Neuroscience is another scientific field that thrives in recent years. Developments in neuroscience and especially in models of the learning process, draw the attention of instructors and educators in an effort to investigate new instructional methodologies (Schunk, 2011, p. 62). Moreover in

the same direction, researchers attempt to formulate a new scientific field known as educational neuroscience. Kathryn Patten refers that "Educational neuroscience, as a first approximation, variously involves syntheses of theories, methods, and techniques of the neurosciences, as applied to and informed by educational research and practice" (Patten and Campbell, 2011, p. 1). Current findings of brain research with educational interest, highlight the significance of early childhood development, the complexity of cognitive functions and the personalized treatment of learning deficiencies (Schunk, 2011, pp. 63-64). In addition, neuroscientists stress the importance of specific instructional applications that conform with brain research, including problem-based learning, simulations and role-playing, active discussions, the use of graphics, as well as the positive climate in educational settings (Schunk, 2011, pp. 64-67).

Contemporary approaches in writing research, highlight the relation of metacognition and self-regulation with writing production. Proficient authors appear to have a variety of metacognitive and self-regulatory abilities, as well as a positive attitude towards writing. They engage in planning, organizing, writing and reviewing with evident convenience, while they have clear mental representations of the genre and the roles of the audience as well as the writer. Moreover, they can efficiently manage the involving cognitive load of the process, and at the same time they employ and adapt strategies, focusing in the attainment of their goals (Spantidakis, 2010, pp. 108-113).

In 1994, Kellogg in his book *The psychology of writing* evaluated the use of word processors in the writing process, and concluded that there are no substantial differences in writing quality using word processors in comparison with pen and paper. His research also indicated, that authors intensify planning and revising using word processing systems, while they reduce the cognitive effort of writing. In addition, Kellogg found that word processing discourage graphical planning, while he highlighted the differences between various authoring tools, regarding writing performance (Kellogg, 1999, p. 160). Kellogg separated the processes or writing and typing, arguing that the formating capabilities of word processors "make for a very slick typewriter", but do not address authoring issues. These difficulties form the concept of idea bankruptcy, including the possible unavailability and problematic retrieval of knowledge, due to cognitive overload or emotional interference, as well as issues in the incorporation of knowledge in a rhetorical context. To address idea bankruptcy, Kellogg proposed the solution of the meaning-making idea processor. The functions of idea processors include the librarian, the editor, the attentional funnel, the inventor and the therapist (Kellogg, 1999, pp. 161-162). This is summary of functions of an idea processor:

> "The first two are knowledge systems that aim to augment directly the availability and accessibility of knowledge. The librarian offers content knowledge, and the editor provides advice based on discourse knowledge. The attentional funnel aims to make the writer concentrate on only one or two processes in an effort to alleviate the attentional overload that commonly occurs when multiple processes are juggled simultaneously. The inventor offers heuristics for creating ideas that solve problems in composition. Lastly, the therapist aims to help the writer achieve a creative flow state, rather than being overcome by frustration, anxiety, or even boredom" (Kellogg, 1999, p. 162).

Kellogg described the model of idea processor, in order to address difficulties in writing, stressing that writing applications should be more than simple typing tools. In this direction, many writing researchers gradually involved in writing applications' design, in order to create useful computer programs for authoring as well as for instructional purposes. They delivered a variety of applications reflecting modern writing research, with a primary focus in the processes of planning and revising. Early attempts targeted adults and included minor revising capabilities, while latter applications for children featured copy/paste capabilities, thesaurus, and spell check. Research showed that these revising features created minor impact to substantial revising. Other approaches, included revising facilitations in writing styles, grammar and word selection, as well as content organization features. These applications delivered important improvements in writing skills, in comparison with users of simple word processors. Finally, the third wave of applications fully incorporated modern developments of writing research, providing authoring guidance to students, personalization and cooperation capabilities, as well as helping students to understand the procedure of writing. These approaches intent to improve the operation of the working memory, internalize metacognitive abilities regarding the phases of planning, organizing, writing and revising, as well as improve self-regulatory skills and reduce cognitive load. They typically used the models of Bereiter and Scardamalia, intending to guide students from knowledge telling to knowledge transforming, showing positive results in the writing process. In conclusion, Spantidakis stressed the role of computer applications and multimedia in writing instruction, since they enable students to develop and internalize metacognitive abilities, support writing strategies in planning and revising, as well as formulate understandings about literary genres (Spantidakis, 2010, pp. 314-322).

3.7 Conclusions

The purpose of this chapter was to unveil the processes of learning, writing and instruction, in order to transform research findings to design guidelines. The review of the literacy regarding major theoretical learning milestones, instructional applications and writing research concludes to the implementation of features that reflect these research principles to a single HTML editor. Our goal is to create a general-purpose writing application, that provides and implements educational facilitations.

The first feature deriving from writing research is a procedural facilitator for document structure. The *structural editor* provides tools, for the selection of the audience, general goal-setting and decisions on the literary genre. Using this tool, the author could divide the text in parts, plan and modify the overall structure of the text, create associations with external content, set subgoals in internal parts, as well as take notes. The structural editor is a key tool for the advancement of the metacognitive and self-regulatory competencies of the writer, aiming to facilitate the processes of planning, organizing and revising. *Revision control* will provide capabilities for the management of different versions of the text, including updating versions, dates, as well as track the evolution of revisions over time. This feature could facilitate major shifts in structure and content, without jeopardizing prior work. The *main editor* involves the process of entering content in text, incorporating structural processing involving adding, removing or modifying paragraphs, headers as well as bullets and numbering. It also involves facilitators, such as thesaurus, dictionary, spell check, emphases, copy/paste, annotations and corrections. This feature deals with technical writing competencies including spelling, grammatical and word meaning issues, which are essential in writing. Moreover, the *visual editor* accommodates the visual layout formatting of the text, including ready visual templates, and universal options for fonts, font sizes, line spacing, emphases and colors. This feature reduces the cognitive load of the writer in dealing with typographical and visual formating issues, modifying the layout according to the preferences of the author. The universal aspect of this feature presupposes the strict semantic annotation of the content in the main editor. Writing research stresses the role of communities of practice and the active involvement of an instructor in the writing process. Writing as a social procedure extends the necessity for *social feedback* capabilities in the HTML editor, enabling evaluations, corrections, annotation and comments from instructors or peers. This is also consistent with behaviorist reinforcement strategies, tutoring and mentoring, cognitive apprenticeship and with active discussions and debates.

The review of learning research concluded to the employment of learning

Structural editor	Audience, goals, genre, headers, paragraphs, notes, subgoals
Revision control	Document revisions and dates
Visual editor	Universal formating on fonts, font sizes, line spacing, emphases and colors
Main editor	Thesaurus, dictionary, spell check, emphases, copy/paste, links, references
Social feedback	Evaluations, corrections, annotation and comments
Exercise editor	Questionnaires, problems and worked examples
Multimedia editor	Video, animation, games, simulations, maps, diagrams, tables, images

Table 3.1: Blueprints based in learning and writing research

multimedia elements to the text, including videos, animations, games, simulations, maps, diagrams, tables and images. *Videos* reflect the principles of modeling and cognitive apprenticeship, while in a form of *animation* or a *simulation* could promote learning by demonstrating content, consistent with social cognitive theory, multimedia learning, sociocultural psychology and educational neuroscience. Interactive educational *games* provide excellent motivational support to students, reflecting mastery learning techniques and multimedia learning approaches. Finally the use of maps, diagrams, tables and images promote visual representation of information promoting learning primary in the principle of advanced organizers. Internet *links* and bibliographical *references* are essential, in order to cite external sources, promoting content generation. The *multimedia editor* provides a simple way to manage these elements including addition, removal and modification. Learning approaches also highlighted the process of problem solving, including social cognitive theory, constructivism and educational neuroscience. Especially in mathematics and science, problem-based learning represent the primary instructional application. The *exercise editor* provides facilitations for creating problems, worked examples and questionnaires, with an emphasis in visual representation and use of graphics. This approach reflects the instructional applications of problem-based learning, programmed instruction, worked examples and computer-aided instruction. A summary of the deriving design decision is shown in 3.1

Chapter 4

Design methodology

4.1 Human-computer interaction

Already from the beginning of the 20th century, researchers and engineers studied the performance of manual tasks in industrial environments. Some years later, the second World War contributed unexpectedly to the study of interaction between humans and machines, since there was a great interest to design effective weapons at the time. These studies eventually led to the formation of Ergonomics Research Society back in 1949, and contributed to the emergence of the discipline of *ergonomics*. *Human factors* was a relative domain, with a primary focus on cognitive aspects of interaction, in contrast with the physical characteristics of ergonomics. These domains investigated different systems, including mechanical, manual or computer-based. The advent of the computer era, cased many of the researches to specifically investigate physical and cognitive aspects of the interaction between computers and their users. These scholars formulated a new field known as *Man-Machine Interaction (MMI)*, which gradually transformed to the contemporary *Human-Computer Interaction (HCI)* (Dix et al., 2004, p. 3).

The core concepts of HCI research refer to users, computing devices and the interaction between them. The term *human* represents single users or a users groups, which engage in tasks and procedures using technological means. The *computer* involves systems, ranging from single desktop computers to large organizations' information systems, while *interaction* includes control and response mechanisms that facilitate the accomplishment of tasks. HCI research incorporates cognitive psychology to describe the users' input-output channels, memory processes, information processing and emotions, as well as the individual differences between them. Users gain input from

visual, auditory and haptic channels, while they respond with body movements (e.g. the hit of a key). Memory processes include the interconnection between sensory, working and long-term memories, while information processing regards reasoning, problem solving, skill acquisition and human errors. Dix et al. (2004) argue that errors occur from contextual changes in the environment of automatic behaviors, as well as from the users' poor understanding of the system's functionality. Users employ *mental models* "to understand the causal behavior of systems", which often are incomplete and inconsistent usually based on instinct instead of evidence. Another critical issue of HCI is the problem of individual differences. Although people share many comparable characteristics and limitations, there are also impressively diverse. Long-term differences in physical and cognitive traits, as well as temporary emotional variations can differentiate user's performance, obligating the designer to take these issues under consideration (Dix et al., 2004, pp. 11-58).

HCI also investigates computers in an analogous manner with humans to identify available tools for interaction, as well as their storing and processing capabilities. Researchers analyze input and output devices (e.g. keyboard, mouses, displays), physical controls and sensors, as well as processing limitations and performance issues (Dix et al., 2004, pp. 59-61). This research contributes to design decisions regarding interaction and to the communication between the user and the system. The study of interaction typically involves theoretical models, system frameworks, ergonomics and different interface styles, aiming to create usable and effective environments (Dix et al., 2004, pp. 123-163). In the next section, we focus on the definition and the stages of the iterative design process, based on the dominant approach of user-centered design (details in subsection 4.3.1).

4.2 The process of iterative design

According to Dix et al. (2004, p. 193-195), design is the process of "achieving goals within constraints", using trade-off techniques in order to maximize the effectiveness by "choosing which goals or constraints can be relaxed so that others can be met". They also refer to "the golden rule of design", which lies in understanding involving "materials", including human psychology, social environment, as well as the technological tools and their limitations. An iterative design process incorporates several distinct stages, including the definition of requirements and the analysis of the involving tasks at first. The actual design process contains repetitive prototyping, evaluation and re-design, in order to reach to a final specification. The last stage of implementation and deployment involves the materialization of that specification

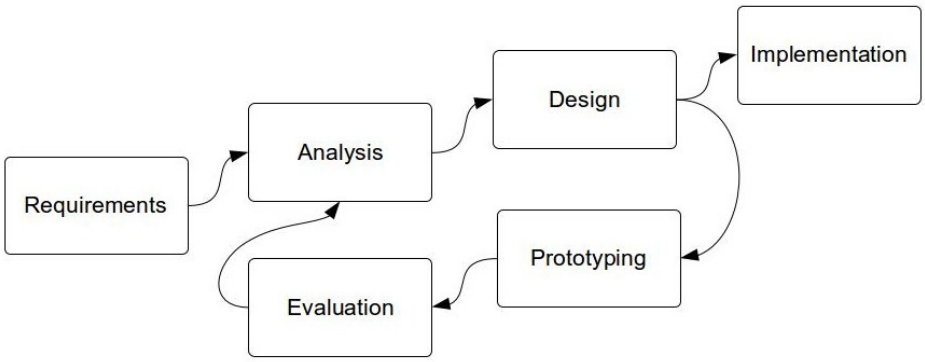

Figure 4.1: The stages of the iterative design

by software developers. This approach represents a typical design process based on user-centered design with an emphasis in iterative design, featuring extensive prototyping and evaluation (Figure 4.1).

Requirements reflect the desirable functionality of the system, representing *what* the system will eventually provide. This phase typically involves a deep understanding of the current situation, regarding users, existing systems and procedures (Dix et al., 2004, p. 196). Designers define as *stakeholders*, the group of people that interact directly or indirectly with the system, focusing their attention to them. They employ a variety of techniques for understanding users, including interviews and open discussions, as well as their direct observation during work. Design teams also conduct surveys with questionnaires, while often they encourage stakeholders to keep diaries. Another method is to create profiles of fictional people also known as personas that represent different user groups, and later predict their interaction with the system (Dix et al., 2004, pp. 197-201). Current developments in requirements engineering highlight the importance of ethnographic methods, referring to the detailed observation of social interactions in a specific environmental context (Dix et al., 2004, p. 470).

Analysis essentially refers to the transformation of requirements specification to detailed sequences of tasks. Designers use "rich stories of interaction" known as scenarios in combination with task analysis methods, to provide a detailed description of the systems functionality (Dix et al., 2004, p. 196). Task analysis uses three different approaches, including task decomposition, knowledge-based techniques and entity-relation-based analysis. The knowledge-based approach features taxonomies of involving objects and actions of a task (Dix et al., 2004, p. 519), while entity-relation-based analysis represents a technique inspired from database design and object-oriented

programming (Dix et al., 2004, p. 525). *Hierarchical task analysis (HTA)* is a popular task decomposition technique that creates hierarchies of tasks and subtasks, as well as defines the sequence and conditions of their execution with clear descriptions known as plans. This method provides a clear algorithmic overview of the system's tasks and processes, in a textual or diagrammatic form (Dix et al., 2004, pp. 512-518).

Design involves decisions on *how* designers could implement the results of task analysis. At this phase, they employ various design principles, guidelines and standards, regarding different types of users, usability, accessibility and navigation. In the next section, we introduce design rules from user-centered design, universal design and cognitive theory of multimedia learning, in order to confront usability, accessibility and learning issues. When the design process completes, there is a comprehensive description of the system ready for implementation. Before the actual implementation though, designers initiate an iterative process of prototyping, evaluating and redesigning, in order to identify problematic aspects of the design that could also affect the phase of task analysis (Dix et al., 2004, p. 196).

Prototyping is the process of creating "artifacts that simulate or animate some but not all features of the intended system" (Dix et al., 2004, p. 241). Typical methods of prototyping include storyboards, limited functionality simulations and high-level programming languages. Storyboards are simple graphical illustrations of the system's interface without functionality that designers create by hand (pen and paper mockups) or by using graphical drawing applications. On the other hand, simulations with limited functionality provide a more comprehensive representation of the final interface, since users can interact with animating components similar to the final product. Designers typically use prototyping tools to create throw-away prototypes like storyboards, or they develop actual programming code that incorporate in the final system, implementing an approach known as evolutionary prototyping. Finally, high-level programming languages can provide "certain features of an interactive system at the expense of other system features like speed of response or space efficiency" (Dix et al., 2004, pp. 244-248).

Evaluation is the assessment process of the prototype's usability and the user experience, while at the same time identifies problems in the functionality of the system. The primary perspectives of evaluation, involve expert analysis and user participation techniques. Expert methods involve cognitive walkthrough, heuristic and model-based evaluation, as well as the use of previous studies. Cognitive walkthrough is the step by step evaluation of system tasks, while heuristic evaluation involves three to five evaluators, which grade tasks based on importance. Model-based evaluation provide "means of combining design specification and evaluation into the same framework", while the use of previous research augments the usefulness of empirical ev-

idence in various situations. In contrast, user participation techniques take place in labs or the users' workspace and employ observational and empirical approaches, or even physiological measurements including heart rate, eye tracking of skin conductance (Dix et al., 2004, pp. 318-364).

Implementation and deployment is the final stage of the design process, after the iteration of task analysis, design, prototyping and evaluation. Developers decide the programming tools, design the software architecture, produce the code of the application, provide documentation and deploy the software for use (Dix et al., 2004, p. 196).

4.3 Design rules

Design teams often associate design decisions to usability results, in an effort to produce more usable environments in the future. This process gradually derived various *design rules*, originating either from empirical evidence or from disciplines that assist to the overall understanding of the user. These domains typically include ergonomics, psychology, sociology or economics, and primary derive design principles. *Principles* represent the most general and abstract form of design rules, focusing to unveil the reasons that promote usability based on the user's behavior (Dix et al., 2004, pp. 259-260). Other design rules involve standards, guidelines and golden rules. A variety of national and international organizations propose *standards*, which are specific and often strict design rules that usually concern large institutions, focusing primary on safety and usability (Dix et al., 2004, pp. 275). Moreover, *guidelines* are more specific rules in comparison to principles with their primary focus on technology, while *golden rules* represent summarized principles, in a form of design advice checklist. Golden rules assist designers to focus to important issues regarding design, producing usually more usable systems in comparison with others how ignore them (Dix et al., 2004, pp. 282). According to Dix et al. (2004, p. 259), "Design rules are mechanisms for restricting the space of design options, preventing a designer from pursuing design options that would be likely to lead to an unusable system". The following design rules include approaches from user-center and universal design, as well as deriving principles from Mayer's cognitive theory of multimedia learning.

4.3.1 User-centered design

In the 1980's, Donalt Norman described *user-centered design* as a philosophy and design methodology in his classic books *User-Centered System Design: New Perspectives on Human-Computer Interaction* and *The Psychology Of*

Everyday Things. His approach highlighted the idea that end-users eventually participate in the design process and influence design decisions, typically in the phases of requirements analysis and usability evaluation. Today, a number of user-centered design methods employ users as design partners, during the whole process (Abras et al., 2004, p. 1). Norman (1988, p. 188) stressed the properties of usability and understandability and concluded that design should:

- *Make it easy to determine what actions are possible at any moment.*
- *Make things visible, including the conceptual model of the system, the alternative actions, and the results of actions.*
- *Make it easy to evaluate the current state of the system.*
- *Follow natural mappings between intentions and the required actions; between actions and the resulting effect; and between the information that is visible and the interpretation of the system state.*

With these requirements in mind, Norman summarized his view of user-centered design in a set of golden rules, known as Norman's seven principles for transforming difficult tasks into simple ones (Norman, 1988, pp. 188-189).

1. *Use both knowledge in the world and knowledge in the head.* Norman suggested that successful design clearly provides the conceptual model of the designer though the system's image, creating an analogous model to the user. This image should clarify the association between existing functions and possible outcomes of the system, by displaying visible and consistent external information that the user could understand. The system's image provides this information either explicitly in the world (physical appearance, manuals) or through constrains in system's functionality. Finally, the external knowledge and functionality should not interfere with the system's usability by expert users that already have formed a conceptual model of its operation (Norman, 1988, pp. 189-190).
2. *Simplify the structure of tasks.* Tasks should reduce the overall complexity of the system, due to limitations of working and long-term memories. Designers need to provide mental aids in complex tasks, task information, feedback and automation, as well as to eventually change the structure of a task to a simpler form. These strategies are effective as far as they don't reduce the overall experience of the user and control capabilities (Dix et al., 2004, pp. 283-284).

3. *Make things visible: bridge the gulfs of execution and evaluation.* The system should visually unveil its functionality as well as proper the ways of executing these functions. Moreover, it should provide visual feedback to the user about the current state and system's outcomes, for evaluation reasons (Norman, 1988, pp. 197-198).

4. *Get the mappings right.* Mappings refer to associations between user's intentions and possible actions, actions and their outcomes, actual state of the system and perceived state, as well as the perceived state and user's requirements, objectives and expectations. Designers should position system controls in direct spacial relation to functionality, while the movements of controls should align to the expected operation (Norman, 1988, p. 199).

5. *Exploit the power of constraints, both natural and artificial.* The use of constrains reduce the number of available options in performing a task, allowing users to perform tasks that they never confronted before in a unique way (Norman, 1988, pp. 199-200).

6. *Design for error.* Design decisions should take into account that users will make errors, obligating systems to handle these errors by adjusting task steps, incorporating recovery capabilities and reversing operations (Norman, 1988, p. 200).

7. *When all else fails, standardize.* When designers cannot overcome difficulties and complexities in the design process, including complex structures of tasks or arbitrary mappings, the appropriate solution is standardization. Design teams standardize the actions of users, system's results, interfaces and displays, in order to provide a standard interface that users will learn once and use it effectively in the future. Learning through standardization also contributes to the reduction of the cognitive load of planning and problem solving in users (Norman, 1988, pp. 200-201).

4.3.2 Universal design

User-centered design stressed the significance of the user for designing useful and usable systems. A major design issue though is to address human diversity, regarding sensory, cognitive and physical abilities, as well as cultural differences, ages and sizes. The approach of *universal design* attempts to confront these issues and provide useful interfaces for all, without exclusions. Dix et al. (2004, p. 366) define universal design as "the process of designing products so that they can be used by as many people as possible in as many situations as possible". Often users face situations in which they

could not interact with the system with the same way they typically do. For instance, a car driver cannot drive and navigate with a GPS device at the same time, obligating designers to provide audio feedback instead of a visual one. Although a unified user experience for all is rather unrealistic, designers aim to provide an equivalent experience accessible to everybody (Dix et al., 2004, p. 366).

Typical differences in humans regard disabilities, various age groups and different cultures. People with some type of disability represent at least the 10% of the world population, demonstrating visual, physical, hearing and speech impairments, as well as a variety of cognitive disabilities. The most critical disability though for designing interactive systems is visual impairment, due to the use of modern graphical interfaces. Early text-based interfaces were providing full functionality to the visually impaired, using synthesized speech from screen readers and braille output devices for feedback. Contemporary approaches still involve tactile and auditory interaction, including electronic braille displays, speech recognition and synthesis, voice recordings, not-speech sounds and sound-icons (Dix et al., 2004, pp. 384-387). Visual impairments affect various visual functions, such the peripheral visual field, visual acuity, contrast sensitivity and color vision. Deficiencies in these functions affect vision categorizing users in colorblind, totally and partially blind. Another condition is macular degeneration allowing only peripheral vision to patients (Stephanidis, 2009, Chapter 6 pp. 3-5).

Hearing impairment affects the use of interactive systems primary in multimedia use and specifically in audio narrations. The typical solution to this issue involves the conversion of auditory information to text with audio caption, providing visual information instead, e.g. video subtitles. People with physical impairments confront difficulties in motor control, resulting input devices like keyboards and mouses often unusable. Designers employ more appropriate input devices, such eyegaze systems that track eye movements, as well as speech-based systems. Another form of disability is speech impairment, which primary affects communication. Solutions for these issues are speech synthesis and text predictive typing tools that could increase typing speeds, in combination with emoticons for communicating better emotional information (Dix et al., 2004, pp. 387-389). Cognitive disabilities derive from genetic and developmental reasons, as well as injuries, strokes, mental illnesses and aging, affecting cognitive functions. In 1999, Francik identified these functions in terms of memory, attention, visual and spatial perception, language and reading, executive functions, mathematical thinking, emotional control, expression and understanding, as well as reasoning speed, the ability to solve novel problems using or not prior experience (Stephanidis, 2009, Chapter 7 pp. 1-3).

Apart from disabilities, different age groups also influence design decisions

of interactive systems. The growing population of older people produces users with reduced sensory, cognitive and physical abilities, as well as with other characteristic, e.g. fear of learning and unfamiliar vocabulary. To address these issues, designers create simple and understandable systems with emphasis in error recovery, which also provide facilitations analogous to disabled users. Similarly, designers consider the varying abilities, preferences and traits of children resorting to participatory design techniques involving children in design teams. Children often appear difficulties in vocabulary or the use of keyboards, obligating designers to create simple systems with multiple input devices, including touch and handwriting interfaces (Dix et al., 2004, pp. 390-391). The last consideration of universal design is cultural differences. People differ in terms of nationality, "age, gender, race, sexuality, class, religion and political persuasion", which influence design decisions mainly for language, cultural symbols, gesture communication and color usage. Localization issues typically involve text orientation for reading (left to right, up to down), while symbols express varying meanings in different cultures. Gestures sometimes express opposite meanings in different cultures affecting multimedia decisions, while color conventions vary. For instance red in the Western culture represents danger, while in China expresses happiness (Dix et al., 2004, pp. 391-392).

Universal design uses *multi-modal interaction* to address human diversity, incorporating different senses in the procedure, mainly vision, touch and hearing. Presenting the same information with multiple modes, multi-modal interaction reduces differences in users, providing an equivalent experience for all. Guidance for universal design consideration derived from North Carolina State University researchers, which they summarized universal design in seven principles, in late 1990's (Dix et al., 2004, pp. 367-368).

1. *Equitable use* stresses the main ideas of universal design for providing the same or equivalent access in usable interfaces for all without exclusions, in terms of usability, privacy, security and safety.

2. *Flexibility in use* refers to the customization of interfaces, in order to adapt to users' needs, preferences and abilities.

3. *Simple and intuitive to use* are the basic features of usable interfaces despite the user's existing expertise, prior experience, concentration and language. The system should comply to the expectations of the user, support localization options and different levels of literacy, while at the same time reduce unnecessary complexity.

4. *Perceptible information* refers to the "effective communication of information regardless of the environmental conditions or the user's abilities". Multi-modal presentation and emphasis in important aspects

of information augments the perceptual results in users.

5. *Tolerance for error* aims to the reduction of damaging consequences caused from user errors or involuntary actions. Effective strategies for error handling are supporting users in complex tasks and providing warning feedback in hazardous situations.

6. *Low physical effort* and reduction of fatigue is a another requirement for universal design, since the physical competencies of users vary.

7. *Size and space for approach and use* refers to the system's usability regarding its physical characteristics and position in space. Users should be able to use it despite their body size and movement capabilities, reaching all the physical controls and also allowing the employment of assistive devices. An important aspect is the system's usability from both standing and seated users.

4.3.3 Multimedia learning

Mayer (2005) investigated the implications of technology in the learning process and instruction. His findings formulated cognitive theory of multimedia learning and concluded in seven principles for designing electronic learning environments. These principles regard e-learning environments, as well as the use of words and graphics, audio narrations, educational material, styles and learning complexity management (Clark and Mayer, 2011).

1. *Multimedia principle: use words and graphics rather than words alone.* Words refer either to textual or oral information, while graphics involve the use of static or dynamic illustrations. Static graphics include diagrams, maps, pictures and drawings, while dynamic involve videos and animations. Researchers define as *multimedia presentation* the simultaneous presentation of words and graphics. The employment of graphics in these presentations aim to provide a deeper content understanding by the user. Researchers though discourage the use of graphics as decorative elements or simple graphical presentations of objects. Instead they suggest illustrations that describe visible or hidden relations between content variables and their change over time, in an effort to organize and interpret the learning content. Empirical evidence suggest that the multimedia principle promotes learning, while the usefulness of static versus dynamic illustrations depend on the learning subject (Clark and Mayer, 2011, pp. 67-89).

2. *Contiguity principle: align words to corresponding graphics.* A common problem in multimedia presentation is the often detachment of

graphics from the corresponding text, when users browse the material scrolling up and down. The contiguity principle highlights the learning importance of presenting graphical illustrations consistently with text, and audio narrations at the same time with graphics. This strategy reduces the learner's overall cognitive effort for understanding content, since there is no need to search and link illustrations to descriptive text (Clark and Mayer, 2011, pp. 91-113).

3. *Modality principle: present words as audio narration rather than on-screen text.* Multimedia presentations with graphical illustrations and text, often overload the visual channel of the learner. During this type of presentation the learner's auditory channel remains inactive. The modality principle suggests that the replacement of text with audio narration appears to have better learning outcomes than text. This technique provides graphical information to the visual channel, and audio content to the auditory channel, reducing the cognitive load of the learner. The limitations of this strategy relate to technical constrains for delivering audio, as well as memory shortcomings that text supports more efficiently (Clark and Mayer, 2011, pp. 115-130).

4. *Redundancy principle: explain visuals with words in audio or text, not both.* Redundant on-screen text refers to textual information that repeats and accompanies an audio narration. The redundancy principle discourages the use of this technique, due to poor learning results, in comparison with either only text or only audio for explaining graphics. Empirical evidence unveil that redundant visual information causes the overload of the visual channel, increasing the learning effort (Clark and Mayer, 2011, pp. 13-149).

5. *Coherence principle: adding material can hurt learning.* A common misconception in e-learning design involves the inclusion of information that not contributes to the achievement of an instructional objective. Researchers concluded that the exclusion of background sounds, and unnecessary text and audio, leads to better learning results. Coherence principle also encourages the employment of simple visual and textual information in the learning process, instead of more detailed descriptions either graphical or textual (Clark and Mayer, 2011, pp. 151-176).

6. *Personalization principle: use conversational style and virtual coaches.* E-learning environments often introduce information with formal writing. The personalization principle encourages designers to use conversational style in writing instead, including first and second person expressions. This technique along with the employment of pedagogical virtual agents in instruction with polite human voice, contribute to the "visibility" of the author. Visual authors highlight their personal

perspective, and as a technique promote the motivation of the learner and learning outcomes (Clark and Mayer, 2011, pp. 179-203).

7. *Segmenting and pretraining principles: managing complexity by breaking a lesson into parts. Essential cognitive processing* refers to the cognitive effort deriving from the complexity of the learning subject. Demanding tasks overload the learners cognitive resources, leading to difficulties in the learning process. The segmenting and pretraining principles attempt to reduce this effort by dividing the material in parts, and by highlighting the key concepts before the multimedia presentation (Clark and Mayer, 2011, pp. 205-220).

Clark and Mayer (2011) also provided insights from empirical evidence for worked examples, practice, collaborative learning, learner's control, thinking skills, as well as games and simulations. Below there is a summary of guidelines referring to worked examples and practice in e-learning environments (Clark and Mayer, 2011, pp. 406-407).

1. *Worked example principle:* Transition from full worked examples to full practice assignments using fading.
2. *Self-explanation principle:* Insert questions next to worked steps to promote self-explanations.
3. *Guidance principle:* Add explanations to worked out steps in some situations.
4. *Varied context principle:* Provide several diverse worked examples for far transfer skills.
5. *Transfer principle:* Promote active comparisons of varied context worked examples.
6. *Spaced vs. massed practice principle:* Provide job-relevant practice questions interspersed throughout and among the lessons.
7. *Practice principle:* For more critical skills and knowledge, include more practice questions.
8. *Distributed practice principle:* Mix practice types throughout lessons rather than grouping similar types together.
9. *Feedback principle:* Provide explanatory feedback in text for correct and incorrect answers.
10. *Contiguity principle:* Design space for feedback to be visible close to practice answers.

11. *Feedback attention focus principle:* Avoid praise or negative comments in feedback that direct attention to the self rather than to the task.

According to Clark and Mayer (2011, pp. 407-408) collaboration is critical to e-learning environments. They encourage small group projects and structured assignments with distinct participant roles based on synchronous and asynchronous communication. They also suggest that the final grade of the assignment should derive from the sum of individual performances. In addition, educational environments should allow users to always control the pace of instruction, the flow of audio and animation, as well as to access previous educational material. E-learning systems could allow experienced learners with more options regarding navigation and instruction control, usually in informational or advanced topics. Contrary e-learning lessons should limit control in students with poor self-regulatory and learning skills.

Suggestions for building problem-solving skills in users include the presentation of expert strategies in worked examples and their continuous comparison to students' problem-solving procedures. Moreover, another key issue is the provision of sufficient guidance for accomplishing tasks, throughout the process. When learning involves job-related problem-solving competencies, Clark and Mayer encourage the employment of specific relevant tools in the instructional process, while they stress video commentary and questions "to ensure that learners attend to and process specific behaviors of expert models". A new development in instruction with e-learning environments is the use of games and simulations. Clark and Mayer underline the association of "goals, rules, activities, feedback, and consequences of the game or simulation" to specific instructional objectives, which learners reach through the system's direction. The interface should be clear and focused on the learning activity, in order to reduce the cognitive effort, while techniques such as feedback explanations and questions also promote the effectiveness of the method (Clark and Mayer, 2011, pp. 407-408).

Chapter 5

Design process

5.1 User requirements

Our approach in user requirements analysis involved an extensive historical review of writing technology from antiquity until today, as well as perspectives from learning theories and writing research. Moreover, my professional experience as a teacher makes me also a stakeholder that could contribute to the application's design. My contribution to requirements analysis involve the methods of direct observation of students, as well as the open discussions with them and my colleagues. In this section, we present a general system description, the categories of the users and the context of use, as well as functional and non-functional requirements.

5.1.1 System description

The name of this writing application is MeanWriter, which emphasizes the WYSIWYM writing approach, and also incorporates the feature of global visual feedback. The main technical reasons for the development of MeanWriter rely on the usability shortcomings of word processors, the WYSIWYG approach and fixed sized formats in everyday life and education.

In addition, my main motivation for the development of MeanWriter derive from my professional experience as a teacher. My work involves teaching basic topics about computers and information technology, developing word processing and office automation skills to my students, as well as assisting them to complete writing projects using technology. The most obvious problem involves the writing process and especially planning and goal-setting, as well as organizing information in a meaningful structure. Word processors do not provide any facilitations or guiding cues for these processes, in order

to assist unexperienced writers to determine their audience, the document's literary genre, overall goals or subgoals of the structural entities. Moreover, the students' majority can not achieve a visually consistent document, due to poor formating skills, indifference for the document's visual representation or even inability to distinguish differences in various fonts and font sizes, as well as headers, bolds or italics. The plethora of word processors' formating capabilities often confuse students, while they facing difficulties to insert, position and manipulate images, links, and tables.

The main goals of MeanWriter involve to provide authors a writing application that they could plan, organize and revise their documents, as well as explicitly markup document structure. This application restricts the visual formating capabilities of the author, and instead provides an automatic global visual presentation, which it fits the preferences of each reader. Another goal is to provide relevant functionality for each working element, different for paragraphs, headers or other document elements. Finally, the author could embed and manipulate various multimedia types, tables and graphs, as well as web applications.

5.1.2 User categories

In this section, we present the profiles of the users, regarding their age and gender, as well as the levels of experience and education. In addition, we define the goals of the application's use for each user. Our user categories include editors and readers. In these categories could participate either teachers or students.

Editors

The characteristics of editors show in Table 5.1.

Readers

The characteristics of editors show in Table 5.2.

5.1.3 Context of use

In this section, we describe requirements of the application's context of use, regarding the surrounding environment, involving users, operating systems, hardware devices and required software. We consider two different contexts, indoors and outdoors.

Goals from system's use	Editors can view, add, remove and modify content, regarding text, document structure and multimedia. They can define the reading audience, the document's general goals and subgoals, as well as the literary genre. They can manage the presentation of the document, as well as to annotate content.
Age	Any 13 to 75 years of age
Gender	Any
Education	Any
Experience	We consider as editors both instructors, as well as students which are guided from instructors.
Physical characteristics	Visual impairments

Table 5.1: Characteristics of editors

Goals from system's use	Readers can view the document's content, structure, audience, goals and literary genre, as well as modify presentation and annotate content
Age	Any 13 to 75 years of age
Gender	Any
Education	Any
Experience	The users should have basic experience and skills in web browsing
Physical characteristics	Visual impairments

Table 5.2: Characteristics of readers

Indoors context

1. *User categories* include both editors and readers that could use this application in the indoors context.
2. *Environment* refers to indoor places, such as the user's home and office, or educational settings, including schools or libraries.
3. *Operating systems* requirements include any operating system with a modern web browser.
4. *Hardware* include devices, including desktop computers with mouse and keyboard, or laptops.
5. *Software* requirements involve the use of any modern browser, but Firefox 27.0 is preferred due to better HTML5 and MathML implementation.

Outdoors context

1. *User categories* include only readers that could use this application in the outdoors context.
2. *Environment* refers to outdoors settings including parks, transportation, or other
3. *Operating systems* requirements include any operating system with a modern web browser.
4. *Hardware* include touch mobile devices, including tablets and smartphones.
5. *Software* requirements involve the use of any modern browser, but the mobile version of Firefox is preferred due to better HTML5 and MathML implementation.

5.1.4 Functional requirements

This section presents the functional requirements of MeanWriter. Below we list the functions of the system with brief description, as well as the functional requirements of each function and the involving users.

Display Structure

Readers and editors can use this feature, in order to view information about the document and its structure. Users can view information about the reading audience, the literary genre, the overall goals and a tree structure of the headers.

1. *Display reading audience.* Users can view information about the reading audience.
2. *Display type of literary genre.* Here they can find out information about the literary genre.
3. *Display document's goals.* They also can view the document's overall goals.
4. *Display document headers.* The users can view a tree-structure of the document's headers.

Insert Structure

This function promotes the initial planning of the document, by setting goals, as well as by defining the reading audience and the literary genre. The user could insert, delete and modify the headers of the document, as well as to set subgoals for each one of them. The involving users are editors.

1. *Insert reading audience.* The user could insert text involving the reading audience.
2. *Insert type of literary genre.* This option allows the user to insert information about the literary genre.
3. *Insert document's goals.* Here the users define the overall goals of the document.
4. *View, insert and manipulate document headers.* The user could insert new headers, as well as delete and modify them.
5. *Insert subgoals to headers.* Here the user define subgoals for each document's header.

Insert Content

The involving users are editors. This function enables many features in text editing, ranging from inserting and deleting text, to copy, paste and cut textual selections.

1. *View the content.* This option unveils the document's content, involving text, headers, tables, links, multimedia and HTML code embeds.
2. *Insert text.* The user can insert text from the keyboard.
3. *Delete text.* Also users can delete text from the keyboard.
4. *Copy/paste and cut.* The features of copy, paste and cut, are also supported with key bindings.
5. *Spell check.* This application uses the default spell check support of the browser with a right-click.
6. *Select text.* The user selects text from paragraphs and markup emphasis and strong emphasis, annotate text, remove format and delete text. In headers can only annotate remove format and delete.
7. *Insert paragraphs.* The only structural entities that the user can add to the text is paragraphs.
8. *Insert and manipulate document sections.* The user could delete and change type of the document's paragraphs to different headers, bullets and numbering.
9. *Insert subgoals to headers.* Here the user define subgoals for each document's header.

Display Content

The involving users are editors. This function enables the reader to select and annotate text.

1. *View the content.* This option unveils the document's content, involving text, headers, tables, links, multimedia and HTML code embeds.
2. *Select text.* The user can select text from paragraphs and headers and can annotate it or remove the format of annotation.

Define Presentation

This feature implements the automatic global visual presentation, which allows the user to select the type of font, font size, line spacing, text alignment, the theme and the visual representation of emphasis and strong emphasis. The involving users are editors and readers.

1. *Select font type.* The user can select the font that could uniformly used in the document and the application, from arial, cursive, verdana, courier, and times.
2. *Select font size.* Users can select the font size of the text, while the headers and application sizes change accordingly.
3. *Select line spacing.* Here the user can define the distance between two lines in the document, ranging from single, to 1.25, and 1.5 to double line spacing.
4. *Select text alignment.* The options for text alignments are left and justify.
5. *Select theme.* Here the user can choose between different application styles, as well as document's presentation.
6. *Select emphasis type.* Users can change the default emphasis representation from italics to a predetermined color.
7. *Select strong emphasis type.* Accordingly here for the strong emphasis representation from bolds to predetermined color.

Insert Non-textual Content

This feature concerns the editors, and imports no textual content to the document, including links, mathematical notation, tables, charts, web applications, such as the Geogebra web application, images, audios, video, and HTML code. The user can manipulate the imports after their insertion.

1. *Insert link.* With this option the user imports a generic link, with no reference with the name "link".
2. *Insert table.* This inserts a predefined 5x4 table.
3. *Insert chart.* The user adds a predetermined line graph.
4. *Insert GeoGebra.* This option allows the user to integrate the web application of GeoGebra.

5. *Insert image.* This adds an standard image.
6. *Insert audio.* This option inserts an standard audio.
7. *Insert video.* The user adds a standard video.
8. *Insert HTML and MathML code.* The user inserts an iframe.

Save

This option concerns the editing users, and provides the transfer and storage of the editing changes of the document to its location.

Undo

This represents the typical Undo functionality, that editor can apply.

Redo

Accordingly for the Redo functionality.

Change Mode

MeanWriter uses three modes in its general functionality, editing, sorting, and reading.

1. *Editing mode.* This mode allows full functionality for document editing and manipulation, except sorting.
2. *Sorting mode.* Here the application allows the editor change the position of the structural entities of the document with drag and drop.
3. *Reading mode.* In this mode, the user can view the structure and content of the document, as well as change its visual layout with the option of presentation. The only editing refers to text annotation.

Selection controls

This functionality regards both editing and reading users, but with some differentiations. Selection controls feature the operations that a user can apply in a selected area. The full functionality of selection controls exists

only in editing mode in the main text, while in headers there is a reduced set of options. In reading mode, there is no editing at all except annotating pieces of text, as well as removing these annotations.

1. *Insert emphasis.* This option removes prior emphasis formating and emphasizes the selected text. The default visual output is italics. This option is available only on the main text on editing mode, and not on header selections.

2. *Insert strong emphasis.* This option acts accordingly for strong emphasis, with the default visual output of bolds. This is also is available only on the main text on editing mode, and not on header selections.

3. *Insert annotation.* This option highlights the text selection with color, in order to create a visual textual annotation. This option do not remove emphasis formating, while it also provides facilitations to comment on annotations. This option is available both in editing and reading modes, as well as in all textual elements, including headers and main text.

4. *Remove format.* This option removes emphases and annotations, in editing and reading modes and all kinds of textual elements.

5. *Delete selection.* Here the selected text is removed. This is available in editing mode only, in both headers and main text.

6. *Copy selection.* With this option the user copies text in the clipboard, and this is available only in the main text in editing mode.

Structural controls

Structural controls emerge when a textual element (headers, paragraphs, bullets and numbering), becomes active by a user click. They emerge in editing mode with differentiations, between them.

1. *Select the type of the element.* Here the user can change the type of the textual elements, the current type to others, including paragraphs, headings 1 to 6, as well as bullets and numbering.

2. *Delete the element.* This option removes the active element.

3. *Add a paragraph.* With this option the user adds a paragraph to the document in the content tab.

4. *Add a header.* This option adds a header 1 to the structure tab.

5. *Hide the controls.* The user can close the menu of structural controls.
6. *Define header goals.* Here the user can define the goals of the section. This option is available only for headers.
7. *Indent.* With Indent the user can create a level lower in bullets and numbering. This option is only available in bullets and numbering elements.
8. *Outdent.* Accordingly Outdent creates a level higher, and also is available only in bullets and numbering elements.

Goal controls

These controls provide the interface for defining writing goals in the document's sections. This feature applies only in editing mode and headers.

1. *Insert section's goals.* Here the user inserts a text with the writing goals of the section.
2. *Save changes.* This is the functionality that saves the text.
3. *Hide the controls.* The user can close the menu of goal controls.

Link controls

This functionality is available in editing mode, and configures links' text and location.

1. *Insert link's text.* Here the user inserts the display text that will show in the document.
2. *Insert link's location.* This is the Internet address of the link.
3. *Save changes.* This is the functionality that saves changes in properties.
4. *Open link.* This feature provides a new browser tab with the link's location.
5. *Delete the link.* Here the user can delete the link.
6. *Hide the controls.* The user can close the menu of link controls.

Table controls

Table controls provide basic functionality of the table element. This is only available in editing mode.

1. *Change table's size.* This option allows the user to manipulate the size of the table.
2. *Add row above.* This control adds a row above the active cell.
3. *Add row below.* Here a row is added below the active cell.
4. *Delete row.* This feature deletes the row of the active cell.
5. *Add column left.* Accordingly with row this controls adds a column left of the active cell.
6. *Add column right.* The same functionality here, but the new row appears right of the active cell.
7. *Delete active column.* This feature deletes the column of the active cell.
8. *Add a paragraph.* With this option the user adds a paragraph after the table.
9. *Delete table.* This option removes the table.
10. *Hide the controls.* The user can close table controls.

Image controls

These controls are functional only in editing mode and configure images.

1. *Change image's size.* This option allows the user to manipulate the size of the image.
2. *Insert caption.* This allows the user to modify the caption of the image.
3. *Insert description.* Here the author can provide a short description of the image.
4. *Insert location.* This is the Internet address of the image.
5. *Add a paragraph.* With this option the user adds a paragraph after the image.
6. *Delete image.* This option removes the image.
7. *Hide the controls.* The user can close image controls.

Audio controls

Audio controls emerge on audio elements in editing mode, and configure their properties.

1. *Insert location.* This is the Internet address of the audio file.
2. *Add a paragraph.* With this option the user adds a paragraph after the audio element.
3. *Delete audio.* This option removes the audio element.
4. *Hide the controls.* The user can close audio controls.

Video controls

Here editing users configure the video elements.

1. *Insert location.* This is the Internet address of the video file.
2. *Add a paragraph.* With this option the user adds a paragraph after the video element.
3. *Delete video.* This option removes the video element.
4. *Hide the controls.* The user can close video controls.

HTML and MathML code controls

These controls manipulate embedded HTML and MathML code, in order to provide mathematical formulas and web applications, including maps, diagrams, games, videos and many more. HTML code and MathML code controls are available in editing mode.

1. *Insert code.* Here the user embeds the code, including iframes, objects and MathML.
2. *Add a paragraph.* With this option the user adds a paragraph after the code element.
3. *Delete embedded code.* This option removes the embedded code element.
4. *Hide the controls.* The user can close these controls.

Annotation controls

Annotation controls allow the user to insert comments on annotations.

1. *Insert comment.* This feature allows the user to enter a comment for the active annotation.
2. *Save changes.* This is the functionality that saves the comments.
3. *Hide the controls.* The user can close annotation controls.

Chart controls

Chart controls help the user to configure charts elements.

1. *Insert x-axis title.* Here the user inserts the x-axis title.
2. *Insert y-axis title.* This option allows the user to insert the y-axis title.
3. *Add x-axis categories.* Categories could be either textual elements or numbers, that represent the data in the x-axis.
4. *Add y-axis data.* Here the user adds data, which represent numbers for each category.
5. *Add a paragraph.* With this option the user adds a paragraph after the chart element.
6. *Delete code.* This option removes the chart element.
7. *Hide the controls.* The user can close chart controls.

5.1.5 Non-functional requirements

In the previous section, we described requirements relevant to functionality, while in this section we present non-functional requirements involving performance, interfaces, usability and documentation.

1. *Performance requirements.* The overall response of the application should be less than a second, considering the client-based nature of the application and the minimal interaction with a server. An experienced user, such as a teacher, should be able to use and understand the basic functionality of the application as an editor, in less than ten minutes.

In contrast, we do not expect an editing student to be as much efficient, since a teacher should guide through the writing process, as well as the application's functionality. A reader should be able to understand the functionality in five minutes, since it is extensively reduced.

2. *Interface requirements.* The required hardware for the interaction with the application includes a keyboard, a mouse or a touchpad, while in the reading mode should be also usable from a touch screen. The required software is Firefox 27.0, due to better implementation of experimental features of the HTML5 and MathML specification.

3. *Acceptance testing requirements.* To ensure the usability of our design, we consider that MeanWriter should be tested with three users with different levels of experience representing teachers. These users should interact with the application in editing and reading mode.

4. *Documentation requirements.* We will integrate into the application simple explanatory texts in the form of documentation, that will act as guiding cues to authors. In addition, we will provide explanatory feedback in help buttons.

5.2 Task analysis

In this section, we introduce the stage of analysis in the iterative design process. Our approach follows a task decomposition technique known as Hierarchical Task Analysis (HTA), in order to provide a detailed overview of our application's functionality. HTA creates hierarchies of tasks and subtasks, describing the conditions and the sequence of their execution with plans. Below we provide the HTA for the two user categories of our application, editors and readers.

Tasks for Editors

0. System's use

    1. Option "Content"

        1.1. Display content

            1.1.1. Display document headers

            1.1.2. Display document paragraphs

            1.1.3. Display document annotations

            1.1.4. Display document links

1.1.5. Display document tables
1.1.6. Display document charts
1.1.7. Display document geogebra
1.1.8. Display document images
1.1.9. Display document audio
1.1.10. Display document video
1.1.11. Display document HTML and MathML code

2. Option "Structure"

2.1. Display Structure
2.1.1. Display header "Audience"
2.1.2. Display information about the reading audience
2.1.3. Display header "Literary genre"
2.1.4. Display information about the literary genre of the text
2.1.5. Display header "Goals"
2.1.6. Display information about the overall document goals
2.1.7. Display a tree structure of document headers

3. Option "Presentation"

3.1. Option "Font"
3.1.1. Option "Arial"
3.1.2. Option "Times"
3.1.3. Option "Cursive"
3.1.4. Option "Verdana"
3.1.5. Option "Courier"
3.2. Option "Size"
3.2.1. Options "6"
3.2.2. Options "7"
3.2.3. Options ...
3.2.4. Options "23"
3.2.5. Options "24"
3.3. Option "Line spacing"
3.3.1. Option "Single"
3.3.2. Option "1.25"
3.3.3. Option "1.5"
3.3.4. Option "Double"
3.4. Option "Text alignment"
3.4.1. Option "Left"

3.4.2. Option "Justify"

3.5. Option "Themes"

3.5.1. Option "Default"

3.5.2. Option "High-contrast"

3.5.3. Option "Green"

3.6. Option "Emphasis"

3.6.1. Option "Italic"

3.6.2. Option "Color"

3.7. Option "Strong"

3.7.1. Option "Bold"

3.7.2. Option "Color"

4. Option "Insert"

4.1. Option "Link"

4.2. Option "Table"

4.3. Option "Chart"

4.4. Option "GeoGebra"

4.5. Option "Image"

4.6. Option "Audio"

4.7. Option "Video"

4.8. Option "HTML and MathML Code"

5. Option "Save"

5.1. Saves the content

6. Option "Undo"

6.1. Undo the previous action

7. Option "Redo"

7.1. Redo the action

8. Option "Mode"

8.1. Option "Editing mode"

8.1.1. Display the options 4, 5, 6

8.1.2. Activate functionality of 10 to 18

8.1.3. Activate functionality of 11 to 17

8.2. Option "Sorting mode"

8.2.1. Hide the options 4, 5, 6 and deactivate functionality of 10 to 18
8.2.2. Execution of 19
8.3. Option "Reading mode"
8.3.1. Hide the options 4, 5, 6 and deactivate functionality of 11, and 13 to 17

9. Option "Resize"
9.1. Click and hold the handler
9.2. Move the mouse to resize the documents width

10. Select text
10.1. Display "Selection controls" menu
10.1.1. Option "Emphasis"
10.1.2. Option "Strong"
10.1.3. Option "Annotation"
10.1.4. Option "Remove format"
10.1.5. Option "Delete"
10.1.6. Option "Copy"

11. Click textual element (paragraph, header, bullets and numbering)
11.1. Display "Structural controls" menu
11.1.1. Option "Type of active element"
11.1.1.1. Option "Paragraph"
11.1.1.2. Option "Header 1"
11.1.1.3. Option "Header 2"
11.1.1.4. Option "Header 3"
11.1.1.5. Option "Header 4"
11.1.1.6. Option "Header 5"
11.1.1.7. Option "Header 6"
11.1.1.8. Option "Bullets"
11.1.1.9. Option "Numbering"
11.1.2. Option "Delete"
11.1.2.1. Delete the active paragraph
11.1.2.2. Delete the active header
11.1.2.3. Delete the active bullets
11.1.2.4. Delete the active numbering
11.1.3. Option "Add"

11.1.3.1. Insert a new paragraph
11.1.3.2. Insert a new header 1
11.1.4. Option "Goals"
11.1.4.1. Display "Goal controls" menu
11.1.4.1.1. Display title "Goals"
11.1.4.1.2. Insert goals text
11.1.4.1.3. Option "Save"
11.1.4.1.4. Option "Hide"
11.1.5. Option "Hide"
11.1.5.1. Close Structural controls
11.1.6. Option "Indent"
11.1.6.1. Create a level lower in bullets or numbering sub-element
11.1.7. Option "Outdent"
11.1.7.1. Create a level higher in bullets or numbering sub-element
11.2. Insert enter
11.2.1. Insert a paragraph at the cursor point
11.2.2. Insert a header 1 at the cursor point
11.3. Insert text
11.4. Delete text from keyboard

12. Click link element

12.1. Display "Link controls" menu
12.1.1. Display title "Link properties"
12.1.2. Insert link's Text
12.1.3. Insert link's Location
12.1.4. Option "Save"
12.1.5. Option "Open"
12.1.6. Option "Delete"
12.1.7. Option "Hide"
12.2. Open link in a new browser tab

13. Click table element

13.1. Display "Table controls" menu
13.1.1. Display title "Table properties"
13.1.2. Option "Add"
13.1.3. Option "Delete"
13.1.4. Option "Hide"

14. Click figure (image) element

14.1. Display "Image controls" menu
14.1.1. Display title "Image properties"
14.1.2. Insert image's Caption
14.1.3. Insert image's Description
14.1.4. Insert image's Location
14.1.5. Option "Save"
14.1.6. Option "Delete"
14.1.7. Option "Add"
14.1.8. Option "Hide"

15. Click audio element

15.1. Display "Audio controls" menu
15.1.1. Display title "Audio properties"
15.1.2. Insert audio Location
15.1.3. Option "Save"
15.1.4. Option "Delete"
15.1.5. Option "Add"
15.1.6. Option "Hide"

16. Click video element

16.1. Display "Video controls" menu
16.1.1. Display title "Video properties"
16.1.2. Insert video Description
16.1.3. Insert video Subtitles
16.1.4. Option "Save"
16.1.5. Option "Delete"
16.1.6. Option "Add"
16.1.7. Option "Hide"

17. Click iframe, embed, object, MathML, GeoGebra element

17.1. Display "HTML and MathML code controls" menu
17.1.1. Display title "HTML Code"
17.1.2. Insert HTML Code
17.1.3. Option "Save"
17.1.4. Option "Delete"
17.1.5. Option "Add"
17.1.6. Option "Hide"

18. Click on annotated text

18.1. Display "Annotation controls" menu
18.1.1. Display title "Annotation"
18.1.2. Insert annotation text
18.1.3. Option "Save"
18.1.4. Option "Hide"

19. Click and hold a document element
19.1. Move the element along the mouse pointer
19.2. Drop the element on release

Plans for Editors

Below we illustrate the plans of task analysis. Plans define the sequence and the conditions of execution of every task.

0. Execution of 1 and then optional execution of 2 - 18.

1. Execution of 1.1
1.1. Execution of 1.1.1 to 1.1.11 if the corresponding elements exist in the document.
2. Execution of 2.1
2.1. Execution of 2.1.1 to 2.1.7
3. Optional execution of 3.1 to 3.7
3.1. Optional execution of 3.1.1 to 3.1.5
3.2. Optional execution of 3.2.1 to 3.2.5
3.3. Optional execution of 3.3.1 to 3.3.4
3.4. Optional execution of 3.4.1 to 3.4.2
3.5. Optional execution of 3.5.1 to 3.5.3
3.6. Optional execution of 3.6.1 to 3.6.2
3.7. Optional execution of 3.7.1 to 3.7.3
4. Optional execution of 4.1 to 4.8
5. Execution of 5.1
6. Execution of 6.1
7. Execution of 7.1

8. Optional execution of 8.1 to 8.3
    8.1. Execution of 8.1.1, and then 8.1.2 if the previous mode was the sorting mode or 8.1.3 if the previous mode was the reading mode
    8.2. Execution of 8.2.1 first and after 8.2.2
    8.3. Execution of 8.3.1
9. Execution of 9.1 first and then 9.2
10. Execution of 10.1.1 to 10.1.6, if the text selection regard paragraph text. If the selection contains header text execute 10.1.3, 10.1.4 and 10.1.5. In reading mode the 10.1.3 and 10.1.4 for all textual elements.
11. Execution of 11.1 and optional execution of 11.2, 11.3 and 11.4
    11.1. Execution of 11.1.1, 11.1.2, 11.1.3, 11.1.5 if the textual element is paragraph. If the element is a header, execution of 11.1.1, 11.1.2, 11.1.3, 11.1.4, 11.1.5. Finally, execution of 11.1.1, 11.1.2, 11.1.3, 11.1.5, 11.1.6, 11.1.7 if the textual element is bullets or numbering.
        11.1.1. Execution of 11.1.1.1 to 11.1.1.9 if the active element is paragraph. If a header is active then execute 11.1.1.1 to 11.1.1.7 and if a header is active in the tree structure execution of 11.1.1.2 to 11.1.1.7
        11.1.2. Execute 11.1.2.1 if the active element is paragraph, 11.1.2.2 if header, 11.1.2.3 if bullets and 11.1.2.4 if numbering.
        11.1.3. Execute 11.1.3.1 except the active element is a header in the structure tab tree structure.
        11.1.4. Execution of 11.1.4.1
            11.1.4.1. Execution 11.1.4.1.1 to 11.1.4.1.4
        11.1.5. Execution of 11.1.5.1
        11.1.6. Execution of 11.1.6.1
        11.1.7. Execution of 11.1.7.1
    11.2. Execute 11.2.1 in all cases. If the cursor is in the tree structure in structure tab, then 11.2.2
12. Execution of 12.1
    12.1. Execution of 12.1.1 to 12.1.7 in editing mode, and 12.2 in reading mode
13. Execution of 13.1
    13.1. Execution of 13.1.1 to 13.1.4

14. Execution of 14.1
    14.1. Execution of 14.1.1 to 14.1.8
15. Execution of 15.1
    15.1. Execution of 15.1.1 to 15.1.6
16. Execution of 16.1
    16.1. Execution of 16.1.1 to 16.1.7
17. Execution of 17.1
    17.1. Execution of 17.1.1 to 17.1.6
18. Execution of 18.1
    18.1. Execution of 18.1.1 to 18.1.4
19. Execution of 19.1 first and then 19.2

Tasks for Readers

0. System's use

1. Option "Content"
    1.1. Identical to editors
2. Option "Structure"
    2.1. Identical to editors
3. Option "Presentation"
    3.1. Identical to editors
4. Option "Resize"
    4.1. Identical to editors
5. Select text
    5.1. Display "Selection controls" menu
        5.1.1. Option "Annotation"
        5.1.2. Option "Remove format"
6. Click link element

6.1. Open link in a new browser tab

7. Click on annotated text

7.1. Identical to editors

Plans for Readers

0. Execution of 1 and then optional execution of 2 - 18.

1. Identical to plan 1 of editors
2. Identical to plan 2 of editors
3. Identical to plan 3 of editors
4. Identical to plan 4 of editors
5. Execution of 5.1

5.1. Execution of 5.1.1 and 5.1.2 in all textual elements

6. Execution of 6.1
7. Identical to plan 18 of editors

5.3 User interface design

In this section, we present our user interface design process, including prototyping and evaluation. Our approach uses the process of iterative design described in Chapter 4, as well as the method of evolutionary prototyping. The evaluation of the prototypes included the execution of various tasks by users, and observation of their performance in usability indexes.

5.3.1 Prototyping and design rationale

The prototypes of MeanWriter are written in the HTML5/CSS3/Javascript software stack and they provide limited functionality for the evaluation process. We will use the produced code as a starting point of the final implementation of the application, following the method of evolutionary prototyping. In this section, we present the prototypes that materialize the project's requirements and their design rationale. The basic guidelines we employ in our design decisions involve simplicity, accessibility, adaptability,

Figure 5.1: The main navigation bar in editing mode

and concealment of functionality, in order to implement the element-oriented approach. The editing users can access all modes, including editing, sorting and reading, while readers can only utilize the reading mode.

Main navigation

We designed the main navigation bar (Figure 5.1), in order to be simple, adaptable to different fonts, font sizes and styles, as well as to occupy the least amount of height on the screen. Moreover the same bar should also accommodate the other two functionality modes of sorting and reading. Following the principle of simplicity and adaptability, we decided not to use any graphical elements, such as icons or decorative graphics in main navigation, providing an entirely textual experience. This address issues of adaption in different screen and font sizes, as well as the simplicity of text reduces the cognitive load of the author during writing. The colors of the default theme incorporate orange as the basic color, which could change in different themes. We decided the default fonts of main navigation to be boldface and white, in order to create the maximum contrast. Moving the mouse on the main options the background changes to gray, while the backgrounds of structure, content, presentation and insert, become black when they are active. This informs the users, about the active tab or option, and creates visual correspondence with sub-menus. High contrast between the application's colors are very useful in accessibility of MeanWriter for the visually impaired. Moreover, we implemented a drop-down list of modes, as a distinct visual element of the menu, in order to highlight the active mode. Finally, the resize option of MeanWriter is a white triangle pointing to the direction that the user can reduce the width. When the mouse pointer is over this option the mouse pointer changes to a resize arrow.

The default active tab of the navigation bar in all modes is the *Content* tab, which displays the actual content of the document. In Figure 5.2, there is an illustration of this option showing the main navigation bar along with the explanatory text about the functionality of MeanWriter. The idea behind the supporting text is to provide the user a starting point for the exploration of MeanWriter's functionality, by inserting, deleting and manipulating textual elements, such as headers and paragraphs. Moreover, the headers of the sample text, create an outline of the document structure in the *Structure*

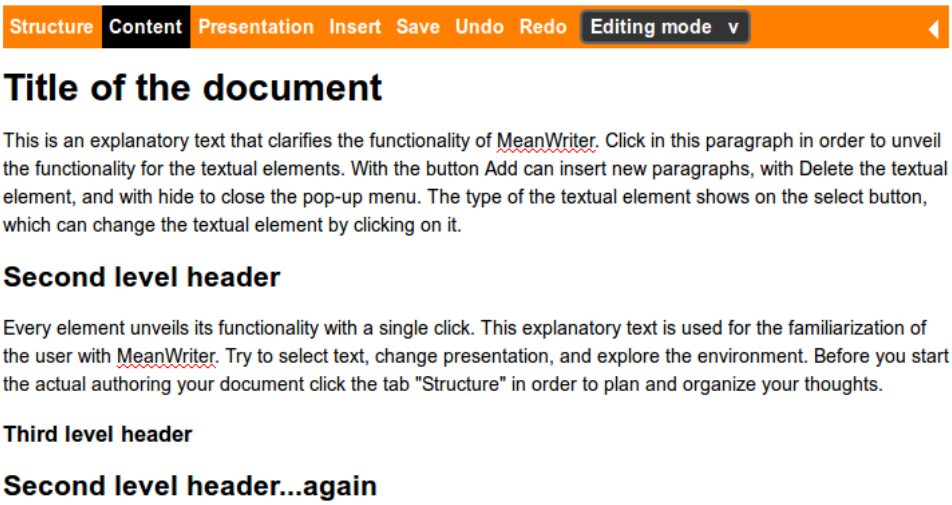

Figure 5.2: The Content tab

tab. This structure shows in Figure 5.3, along with explanatory paragraphs about the reading audience, the literary genre and the document's general goals. We decided to include all the planning and organizing features in a single tab, in order to the author could instantly go back to review structure, as well as set specific subgoals for each section. The sections of audience, literary genre and goals are flexible to edit as normal MeanWriter textual elements.

In Figure 5.4, we illustrate *Presentation*, the next option that controls the visual layout of the document, and implements the automatic global visual presentation feature. This menu unveils further drop-down lists considering the text's font and size, line spacing, text alignment, themes and the visual presentation of emphasis and strong emphasis. The background of the drop-down menu is black, in order to form a consistent visual entity with the top-menu option. There is some styling to the drop-down lists to maximize contrast with a gray background and light gray border, while when the menu becomes active, the background turns black and the border white. When drop-down list options follow the same pattern with white bold option in a black background.

Insert (Figure 5.5) is an option that adds non-textual elements in the document, including links, mathematical notation, tables, charts, the web application of GeoGebra, images, audio and videos, as well as embeds HTML and MathML code from various sources. The options of this menu are simple links, which follow the same design approach featuring bold white fonts contrasting with a black background, positioned in a horizontal line, in order

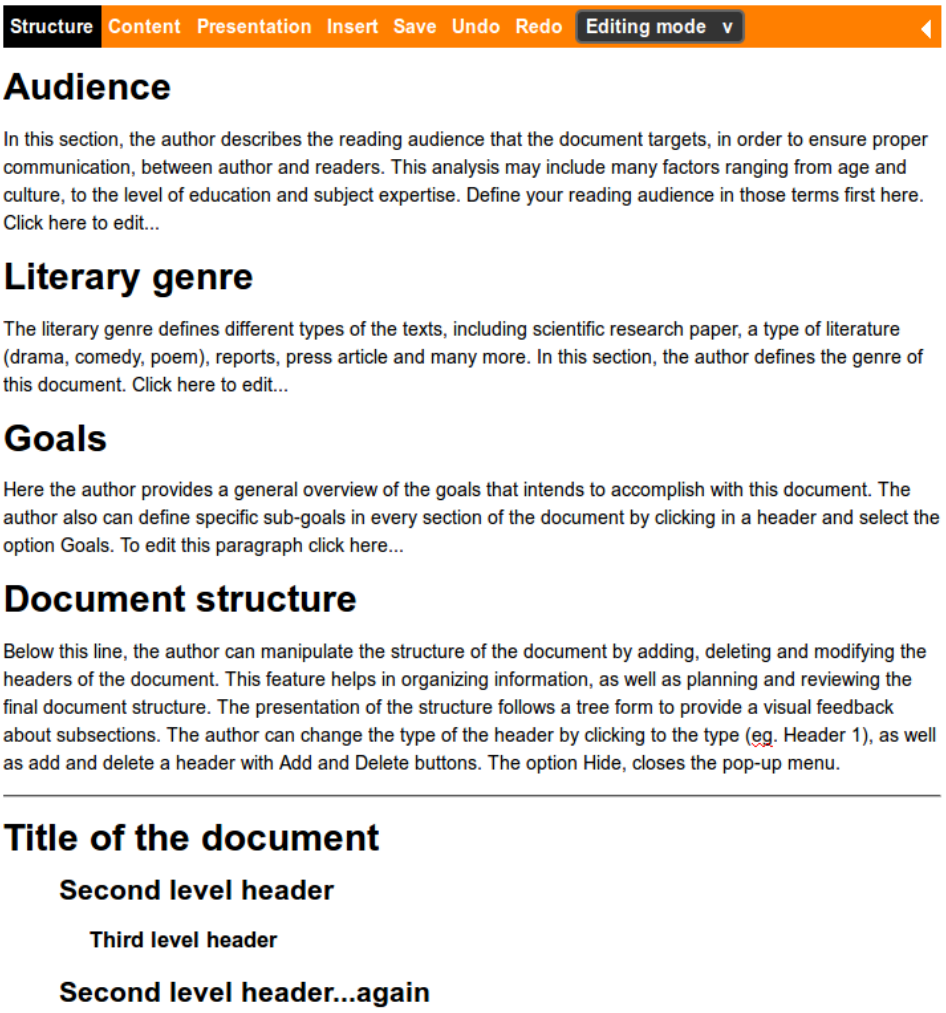

Figure 5.3: The Structure tab

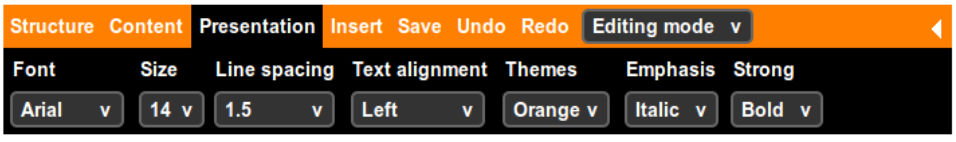

Figure 5.4: The Presentation menu

Figure 5.5: The Insert menu

Figure 5.6: The main navigation bar in sorting, reading and only-reading modes

to save vertical space. The options of the menu change their color in gray, when active. The options of *Save*, *Undo* and *Redo*, are visible only in the editing mode, representing the only general options of MeanWriter, which also turn to gray when active.

The main navigation bar design in sorting and reading modes is identical to the editing mode, with the only difference the unavailability of the Insert, Undo, Redo and Save options. The transition between modes, as well as the indication of the working mode is available in same drop-down list with editing mode. MeanWriter also supports the deployment of only reading mode version, which features the same functionality with reading mode, but without the ability to switch between modes. The illustrations of the main navigation bar in reading, sorting modes and only reading deployment shows in Figure 5.6.

Controls of textual elements

The main idea behind the controls of MeanWriter is that they are not visible and they pop-up only when the user clicks a structural element. This design decision materializes the element-oriented approach, a concept that narrows down the relevant operations of the user to those which apply to the active textual element. When another structural element becomes active the previous pop-up menu hides and another pop-up emerges in the newly activated element. The general visual design of MeanWriter's controls involve a high-contrast button-like links and drop-down lists with white bold characters in a black background. These controls operate in editing mode, in both content and structure tab of MeanWriter.

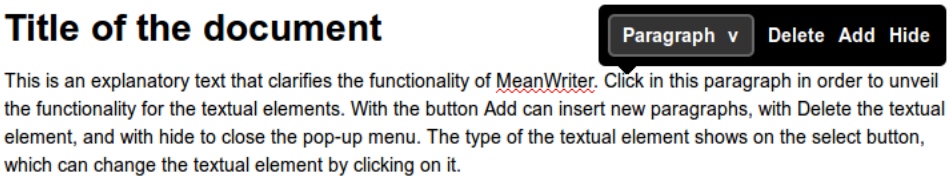

Figure 5.7: Paragraph controls

Figure 5.8: Header controls and goal controls

We use two approaches in the positioning of pop-up controls, with the first to be above and the other to be below the structural element. In paragraphs, headers, bullets and numbering the pop-up menus are over the element in a horizontal arrangement, in order to increase visibility and at the same time to reduce the interference of controls to the rest of the document. During writing, these controls are constantly visible and their appearance and positioning is critical to ensure usability. Paragraph controls shows in Figure 5.7, which they pop-up with a single click on a paragraph, and they appear in the right side of the screen.

Header controls follow the same design approach with paragraph controls, differing only in positioning, since they appear above the header in the y axis and at the x axis coordinate of the mouse pointer. This increases the mobility of the pop-up, allowing the user to place these controls in a preferred position with a single click. In Figure 5.8 we illustrate header controls and header goal controls, a pop-up that appears when a user selects the Goals option.

Goal controls use a title, a simple adjustable in height text area to enter the goals of the section, a saving option and an option that closes the pop-up. The positioning of the pop-up is fixed below the header, in order to easily adjust the height of the text area. We decided to use a pop-up to insert and

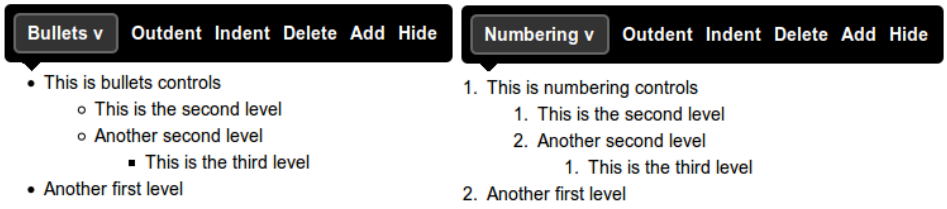

Figure 5.9: Bullets and numbering controls

view the goals of the section, instead of a always visible representation of goals, in order to separate planning and organizing from writing, as well as to sustain a consistent design with the structure tab, in which we employ the same pop-up.

Bullets and numbering follow the same design approach with the controls of previous textual elements. These controls align their options horizontally in fixed position on the right of the screen, as shown in Figure 5.9. They are wider than previous controls, since they incorporate two additional options, indent and outdent.

Besides paragraphs, headers, bullets and numbering the author can apply operations on the document's text as well. The user initially selects some text and then a vertical pop-up menu appears with the available operations automatically in the mouse pointer position. The user can emphasize, annotate, delete, or copy pieces of text, if these belong to a paragraph or to a bullets or numbering list. The option of Remove format removes all the text editing that the author formerly applied in the document. When the author selects text, which is a part of header the options are limited, but follow the same design pattern. The text selection controls show in Figure 5.10, both for header and main text. We preferred a vertical orientation for this pop-up menu, due to the number of available operations, which made the approach of a horizontal menu unusable. Moreover, this alignment scales smoothly and

Figure 5.10: Text selection controls in editing mode

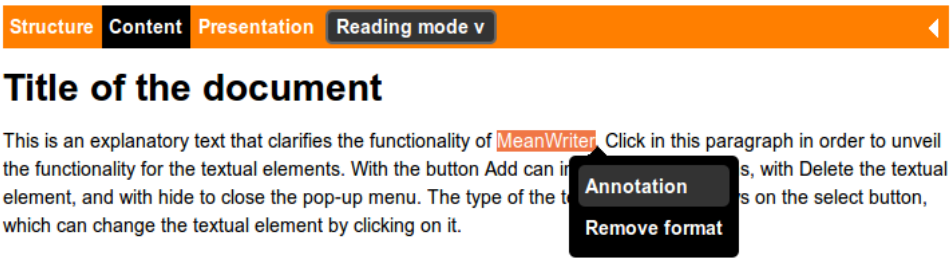

Figure 5.11: Text selection controls in reading mode

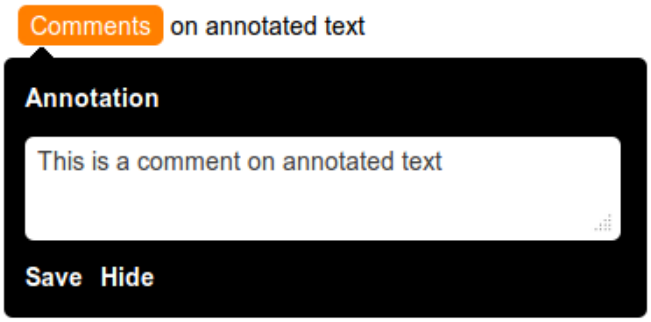

Figure 5.12: Annotation controls

consistently in various font sizes that the user could select.

In reading mode, the selection controls menu limits its functionality only in creating and removing annotations (Figure 5.11). We preferred the term "Remove format" instead of a possible "Remove annotation",in order to ensure consistency between editing and reading mode. This is the only operation available in reading mode, which is the same for header text.

When the user annotates a piece of text, the application changes its color to white, while at the same time highlights the background of the selection to orange. This creates a strong visual annotation for editing, as well as reading purposes, which explains the availability of the feature also in reading mode. In Figure 5.12 we illustrate the annotations controls that appear when the user clicks on an annotation. This pop-up allows the user to add comments to the annotated text as a feedback to the author or as a personal note, by entering text in the available text area. The user can adjust the height and the width of the pop-up starting from a minimum size.

Controls of non-textual elements

In this section, we illustrate the prototypes of the pop-up controls involving non-textual elements, such as links, mathematical notation, tables, charts, GeoGebra, images, audio, videos and HTML code. These controls are available only in editing mode of MeanWriter, while in sorting and reading do not display at all. When the user adds from the Insert menu a non-textual element, the application incorporates in the document structure a standardized element of each type. With a single user click, a pop-up menu emerges including a title, the values of element properties and available operations, following MeanWriter's design philosophy. The controls appear below the element with a vertical orientation in a simple form consisting of the name of the property and an input box with the current value. Below the properties appear the available operations of each element, in a horizontal orientation.

Links in MeanWriter are non editable as simple text, and they can alter their properties only with link controls shown in Figure 5.13. In editing mode, the author pops up link controls with a single click on each link, featuring the properties of link's text and its location. The user can manipulate these values and save the changes, as well as open the link for testing or delete it. In reading mode, when a reader clicks a link, opens a new tab in the browser with the location specified.

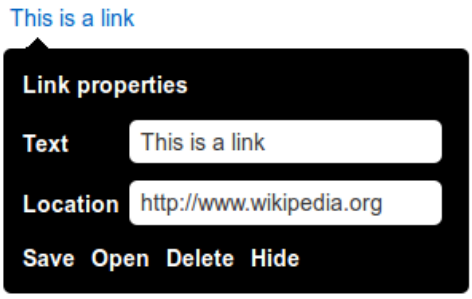

Figure 5.13: Link controls

We illustrate table controls in Figure 5.14, on the standardized table that MeanWriter adds in the document structure. These controls feature only the functions of add a paragraph, delete the table and hide the controls, since the browser provides build-in operations of the table, such as add and delete columns as well as rows. Another built-in functionality of the browser is the resize of the table, as well as inserting and deleting text in table's cells.

Image controls (Figure 5.15) can configure the caption, description and the location of an image. The description changes the alternate text of the image, which provides a small description if the image can not be loaded. The default standardized image features an explanatory text that prompts the user to click the image to pop-up table controls.

Following the same concept, audio controls appear when clicking an audio element as shown in Figure 5.16. The only property that the user can alter

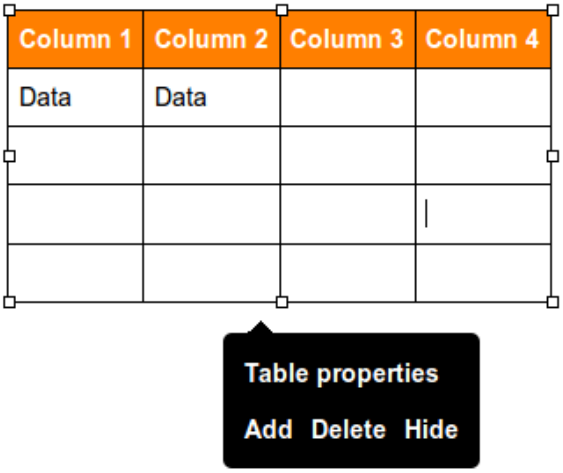

Figure 5.14: Table controls

Figure 5.15: Image controls

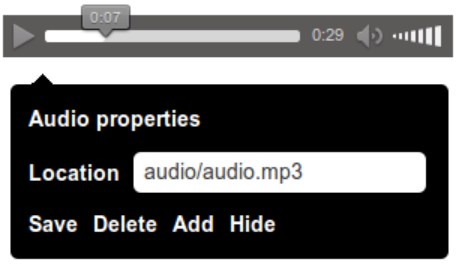

Figure 5.16: Audio controls

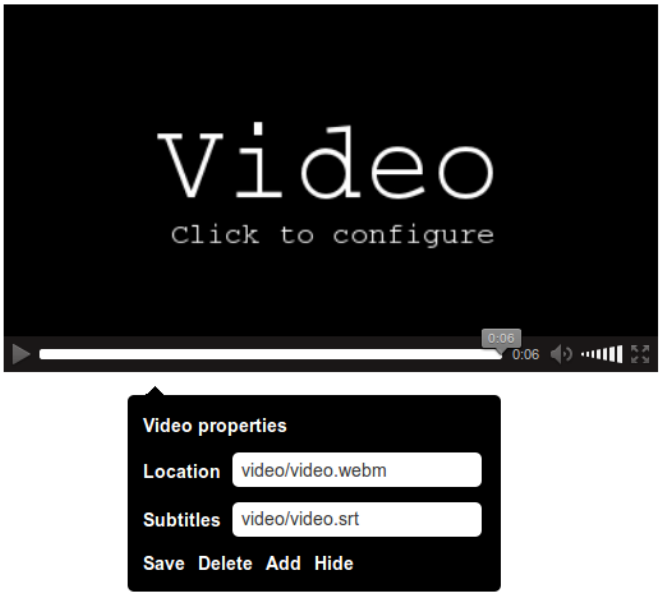

Figure 5.17: Video controls

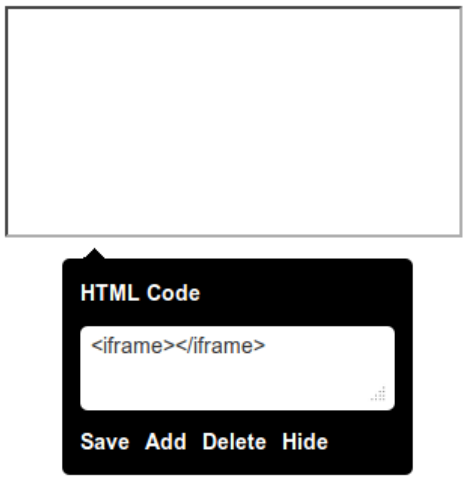

Figure 5.18: HTML code controls

is the location of the audio file. In Figure 5.17 video controls configure the properties of the video element, such as the video file location and subtitles, featuring a standardized video which prompts the user to click it, in the same fashion with the standardized image.

Finally, HTML code controls configures the embedded elements of the document. The standardized HTML code element provides a simple iframe element, in which the user can delete the initial content and paste the HTML code of choice. In Figure 5.18 there is an illustration of the default element, with the HTML code controls, while in Figure 5.19 there is the web application of GeoGebra. In these cases the activating event of HTML code controls is the mouse enter in the element, since the events that occur in the embedded element is consumed from internal functionality. Other useful, embedded elements include YouTube videos, Google maps, and other web applications.

5.3.2 Evaluation

In this phase of the design process, we will evaluate the usability and user experience of our prototypes, employing a user evaluation method. This approach includes the observation of the users during the execution of various tasks, monitoring their performance on specific usability indexes. These metrics include the task completion duration, the ratio of the mistaken to total actions, as well as user satisfaction, which emerged from mini interviews.

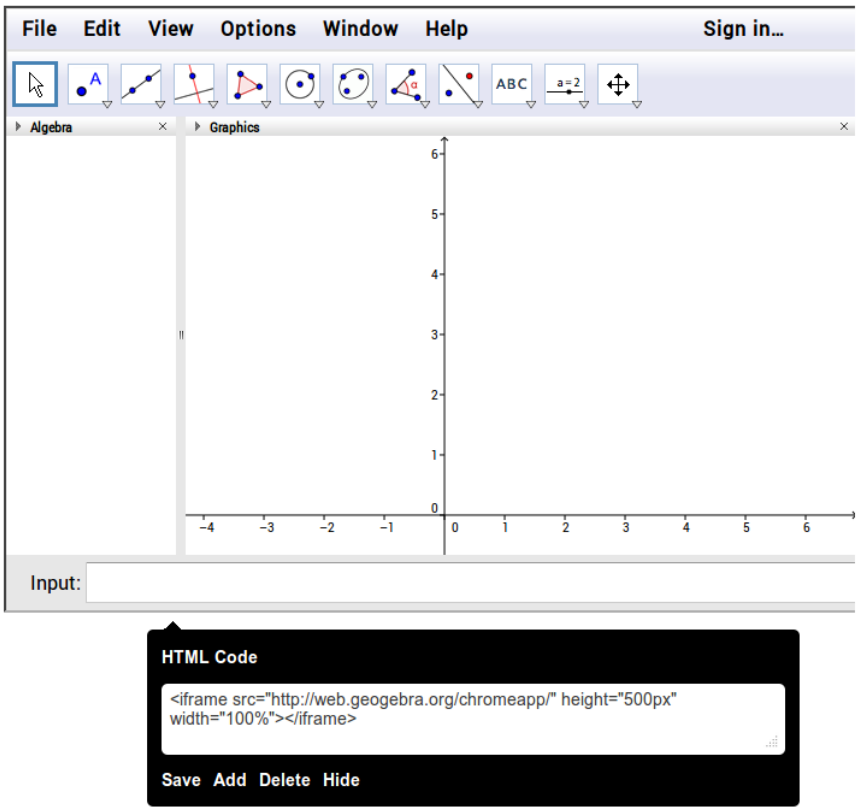

Figure 5.19: GeoGebra controls

User evaluation

In this section, we illustrate the results of the user evaluation process, regarding three users. We selected users with prior experience in general computer use and WYSIWYG word processing, in order to be able to make a comparison between the two approaches. We didn't use unexperienced users, since we assume the presence of an guiding instructor in the writing process or application's use. User 1 is a software designer in the telecommunications organization of Greece, with experience in word processing and computer use. User 2 is informations technology instructor in secondary education, with extensive experience in computer use, word processing, as well as building the word processing skills of students. Finally, User 3 is a Spanish literature instructor, with the basic skills both in computer use and word processing. Users haven't interacted or explored the application before the evaluation process, in an effort to measure how intuitive is the design, and how users could complete task without prior knowledge. Below we present a usage scenario with various tasks that we asked the users to complete.

1. Reduce the width of the document in half, and restore it back
2. Insert and emphasize the word "Evaluation" at the end of the first paragraph in the Content tab
3. Emphasize strongly the word "MeanWriter" in the same paragraph
4. Undo and then Redo the action
5. Check the spelling of MeanWriter
6. Annotate the word "MeanWriter" in the second paragraph and create the comment "This is an HTML editor" in this annotation
7. Delete the sentence "Try to select text, change presentation, and explore the environment" in second paragraph
8. Remove emphasis from the word "Evaluation", annotate the word "Evaluation" and then remove the annotation
9. Copy and paste the word "Evaluation" in the same paragraph
10. Change the name of "Second level header" to "Third level header" and its level to "Header 3"
11. Define the writing goals of the section, by inserting the text "These are the goals"

12. Add a paragraph after the header with the text "My paragraph" and then delete the paragraph
13. Add a "Header 1" header in the same location with the text "My header"
14. Close the pop-up controls of "My header" and then reenable the pop-up controls of "My header"
15. Check the overall structure of the document
16. Add a "Header 3" header at the end of the document's structure with the title "My second header"
17. Create a paragraph at the Audience section with the phrase "My audience" and delete the explanatory text
18. Change the font to Verdana, its size to 12, double line spacing, justify the text alignment and strong emphasis to color
19. Insert the link "Wikipedia" after the word "Evaluation" with the internet address "http://wikipedia.org" and then open it.
20. Insert an image of your preference from the web, with the caption "My image"
21. Insert a 3x3 table under "My header" and then add one more line
22. Delete the table and the image and insert the HTML code of a Google maps image of Crete after the first paragraph
23. Insert any video file from "http://commons.wikimedia.org/wiki/Commons:Video"
24. Insert the GeoGebra web application at the end of the document and then delete both Geogebra and the video
25. Click the Wikipedia link in reading mode
26. Annotate the text "This is an explanatory text", and then remove the annotation
27. Move the second paragraph under the "Title of the document" in sorting mode
28. Change in editing mode and insert the bullet list bellow, at the end of the document

    - Level 1
      - Level 2

  - Level 2
- Level 1

The results of the evaluation process illustrate in the tables bellow. In Table 5.3, we summarize various remarks and observations during evaluation that provided valuable feedback, while in Table 5.4 we measure the time that the user needs to complete a task. The final metric include the ratio of mistaken actions to total actions shown in Table 5.5. We consider as a mistake any unnecessary or irrelevant action, in comparison to the optimal execution of the task. Due to this assumption we expect some mistakes that regard the initial exploration of the environment. Finally, we use the term "Fail" to indicate, when a user made a series of attempts in a long period of time, but didn't complete the task and gave up.

The starting point of the conclusions of the evaluation regard the tasks that users did not complete. User 2 completed all the tasks, while User 1 didn't complete the spell check task. He didn't expected the typical browser functionality, since the application provided its own events and menus, resulting not to right-click the word. User 3 also failed to complete this task, searching the applications menus and the text selection menu for spell checking functionality. Moreover, User 3 failed to complete the resize task, thinking that the handler is a decorative element. Finally, User 3 although managed to create a bullets list couldn't create the structure because couldn't understand Outdent and Indent functionality. In the implementation phase, we will consider the possibility to add text in the resize handler and replace the words Outdent and Indent with arrows.

The tasks that troubled users the most, show in Table 5.5. User 1, in task 20 tried to copy and paste the whole image from the web and not just its web address, while User 2 puzzled over task 18, in which tried to actually format the text and not its presentation. User 3 surprised from the automatic pop-up in text selection, while in task 6 didn't see the annotation option or intuitively click on the annotated area. In task 26, Users 1 and 3 went back to editing mode to remove the annotation, because they didn't see the Remove format option. Finally, in table Table 5.4 we see that the more experienced users were similarly fast, displaying variations only in tasks with high error rate, while User 3 was significantly slower. We may consider to implement help buttons in every element control, and a pop-up in annotations that could inform the users to click on the text to comment.

Table 5.3 features useful notes from the observation of users during evaluation. The most obvious observation is that nobody used the Add button to insert a new paragraph or a header. Although this approach is correct we must conclude that this functionality is not really useful in experienced users. User 1 used in many other cases the keyboard, either to copy/paste

Step	User 1	User 2	User 3
1	Ok	Ok	Searched all the options, failed
2	Used the keyboard	Ok	Selected and right clicked, surprised by pop-up
3	Ok	Clicked emphasis	Ok
4	Ok	Ok	Ok
5	Did't know what to do, failed	Ok	Did't know what to do, failed
6	Ok	Ok	Many wrong attempts
7	Used the keyboard	Ok	Ok
8	Emphasis again	Ok	Ok
9	Ok	Ok	Ok
10	Ok	Selected text without reason	Ok
11	Didn't saw Goals	Ok	Ok
12	Used enter to add a paragraph	Used enter to add a paragraph	Used enter to add a paragraph
13	Ok		Ok
14	Ok		Ok
15	Ok		Ok
16	Used enter to add a paragraph	Used enter to add a paragraph	Used enter to add a paragraph
17	Used enter to add a paragraph	Used enter to add a paragraph	Used enter to add a paragraph
18	Ok	Tried to change text formating	Ok
19	Ok	Ok	Ok
20	Copied/pasted the whole image	Added a paragraph without reason	Copied/pasted the whole image
21	Ok	Added a paragraph without reason	Ok
22	Thought to insert image	Ok	Ok
23	Ok	Ok	Ok
24	Added a paragraph without reason	Ok	Ok
25	Ok	Ok	Ok
26	Went back to editing mode	Ok	Confused, thought to go back to editing mode
27	Ok	Ok	Ok
28	Checked Insert First	Checked Insert First	Checked Insert First, couldn't see or understand Indent, Outdent, failed

Table 5.3: Observations on user task execution

Step	User 1	User 2	User 3
1	3	10	Fail
2	27	7	51
3	4	8	16
4	3	3	17
5	Fail	9	Fail
6	14	18	133
7	3	5	6
8	7	20	46
9	3	9	20
10	9	16	51
11	30	34	16
12	10	18	43
13	20	19	56
14	5	5	19
15	5	8	17
16	10	13	35
17	35	22	36
18	35	98	41
19	33	45	134
20	103	63	149
21	29	22	62
22	90	105	185
23	65	47	67
24	15	7	28
25	9	7	53
26	41	26	140
27	30	8	26
28	72	64	Fail

Table 5.4: Duration of task execution

Step	User 1	User 2	User 3	Least
1	0/2	2/4	Fail	2
2	0/3	0/3	4/7	3
3	0/1	1/2	0/1	1
4	0/2	0/2	0/2	2
5	Fail	1/2	Fail	1
6	0/4	0/4	5/9	4
7	0/1	0/1	0/1	1
8	1/4	0/3	1/4	3
9	0/2	0/2	0/2	2
10	0/3	1/3	0/2	2
11	1/3	0/3	0/3	3
12	0/3	0/3	0/3	3
13	1/3	1/3	1/3	2
14	0/2	0/2	1/3	2
15	0/1	0/1	1/2	1
16	0/3	0/3	0/3	3
17	0/3	1/4	1/4	3
18	1/7	6/12	1/7	6
19	1/5	2/6	1/5	4
20	3/9	1/7	1/7	6
21	0/3	1/4	0/3	3
22	1/7	0/6	0/6	6
23	0/4	0/4	0/4	4
24	1/4	0/3	0/3	3
25	0/2	0/2	1/3	2
26	1/3	2/4	4/6	2
27	0/2	0/2	0/2	2
28	1/10	2/11	Fail	9

Table 5.5: Ratio of mistakes to total actions

or delete text. Another interesting behavior is the addition of a paragraph before the insertion of a non-textual content, which User 1 and 2 made in three different cases. Finally, all the users checked first the Insert tab to insert a bullets list in the last task. Some ideas for implementation regard the integration of bullets and numbering in the Insert tab and provide a visual feedback for new paragraphs, in order to highlight that inserting a paragraph is irrelevant to non-textual element addition.

The users commented on the usability and usefulness of MeanWriter, during evaluation, as well as in a mini interview. All the users found the application overall usable without major misconceptions and inconsistencies, as well as with great aesthetics. The WYSIWYM approach and the global visual presentation impressed User 1, which found it very useful in the final document quality and visual output. User 2 found very useful the facilitations of adding non-textual elements, as well as planning and organizing, features that could help users in the writing process. MeanWriter's simplicity and absence of large sets of formating capabilities impressed User 3, that she found this perspective much less confusing than the WYSIWYG approach. Our conclusion, is that every user found interesting features regarding his/her own interests and abilities, as well as they could eventually cope with various tasks. User 3 stated that she utilize some training before use, while she required some additional help options.

5.4 Implementation and problems

The technologies that we used to implement the design of MeanWriter involved HTML5, CSS3, Ajax and Javascript, as well as three Javascript libraries, such as jQuery, jQueryUI and HighCharts. JQuery[1] is a fast cross-browser Javascript library for "document traversal and manipulation, event handling, animation", and simple Ajax interactions, that simplifies and enriches typical Javascript scripting. Moreover, jQueryUI[2] is a library built on jQuery, used for "user interface interactions, effects, widgets, and themes", that simplifies and automates visual interaction. Finally, HighCharts[3] is a Javascript library, which offers interactive charts for websites and web applications.

One of the most promising and exciting features of the HTML5 specification is the `contenteditable` attribute. Just by setting `contenteditable="true"` as an attribute in an HTML element, the user can modify the element, providing the base of modern rich-text editing. This applies also in non-textual

[1] `http://www.jquery.com/`
[2] `http://www.jqueryui.com/`
[3] `http://www.highcharts.com/`

elements (e.g. images or tables), in which implementations of the attribute involve browser built-in functionality, including resize options or other manipulation. For instance, Firefox provides controls for resize, add and remove, rows as well as columns in a content-editable table, while Google Chrome do not. This highlights the experimental development state of this technology, which results different implementations and support from browsers. Our approach utilizes a content-editable div element, in which the user can modify and insert textual, as well as non-textual content. All of the MeanWriter's controls interact with this div, in order to provide an text editing experience. Moreover, another cross-browser issue we encountered is that Firefox can not paste text in our content-editable div with its built-in tools (right click menu), while Chrome can.

Another critical technology in rich-text editing is `execCommand`. This command provides functionality for formating, adding, removing and manipulating HTML elements and text, as well as features such as Undo, Redo, Copy and Paste. A challenging issue that we encountered is the HTML element manipulation with execCommand, with is very different from other Javascript commands and jQuery. For instance a simple removal of an HTML element, requires the position of the caret to the start of the element, a creation a textual selection that includes the element and its deletion with execCommand. If we used a simple jQuery command for the removal of the element, the user couldn't access the previous states with Undo, since this functionality apply only in execCommand actions. These technologies are still in early development in most browsers creating functional inconsistencies. Another issue we faced is the inability to use the clipboard for copy and paste functions, due to security reasons. Browsers disable these functionalities from web applications, providing only their built-in menus and shortcuts. Our solution to the copy/paste issue is a message when the user selects the Copy option, to use instead keyboard shortcuts.

Many bugs also exist in jQueryUI's functionality, including the resize handler and the sorting mode. The resize handler cannot be fixed in a position, resulting its disappearance when the user scrolls down and the stick menu shows. Moreover, the sorting mode is usable only for once, due to a known bug that creates problems when the user disables and re-enables the `sortable` command. Another issue is the revealing of the controls of a web application, since the application's functionality consumes the click event. In the prototyping process, we employed a mouse enter event, which is relatively usable and annoying to the user. In the final implementation, we created a white frame at the bottom (black for charts), which the user clicks and reveals the controls.

During the design process, we researched the state of MathML implementation in browsers, as well as various math equation editors, in order to

check the potentials of integration to MeanWriter. Firefox and Safari are the only browsers that provide a complete MathML implementation, which is also content-editable, meaning that the user could access and manipulate mathematical notation in a WYSIWYG approach as text. We couldn't find though a client-based WYSIWYG equation editor written in Javascript or jQuery that we could integrate to MeanWriter, so we decided to provide only the integration of MathML code. Today's common practice is the use of math rendering systems, such as MathJax[4], in order to display mathematical notation, without editing capability. Our approach provides the editing of MathML code, as well as the rendering from the browser.

The modifications we implemented of the original design include minor improvements, such as the replacement of "Indent" and "Outdent" with ">" and "<" characters accordingly, as well as the addition of "Strong emphasis" in the selection's menu instead of "Strong". Moreover, we added help buttons in all of our pop-up menus in order to ensure that the user can get relevant assistance for the completion of any task. We didn't implement evaluation suggestions, such as the inclusion of text in the resize handler, due to technical difficulties. Moreover, we didn't modify the right click default functionality, in order to keep the copy/paste options when the user imports text from other sources, as well as the annoying mouse enter event for charts and embedded HTML and MathML code, due to time limitations. We also rejected the inclusion of Bullets and Numbering in the Insert menu, because if these options are clicked again they remove bullets and numbering, which is inconsistent with the concept of the Insert menu.

Moreover, we conducted a final evaluation regarding MeanWriter's accessibility and conformance to W3C standards of our HTML5 and CSS3 files. We used the official Markup[5] and CSS[6] Validation Services of W3C, as well as Accessibility Color Wheel[7], in order to validate our color decisions. All our files include valid HTML5 and CSS3 code, while our accessibility themes AA and AAA conform with Web Content Accessibility Guidelines (WCAG) 2.0[8] for color use, for normal (AA) and enhanced (AAA) accessibility level.

[4] `http://http://www.mathjax.org/`
[5] http://validator.w3.org/
[6] http://jigsaw.w3.org/css-validator/
[7] http://gmazzocato.altervista.org/colorwheel/wheel.php
[8] http://www.w3.org/TR/WCAG20/

Chapter 6

Conclusions and future work

This thesis highlighted the necessity to create a paradigm shift from the visual formating of WYSIWYG to the structural semantic annotation of WYSIWYM. The main advantage of this, is that the reader becomes the visual designer and transforms the document layout according to his/her own preferences and visual abilities. Moreover, we identified the changes in computing with the emergence of mobile devices and the growing reduction of paper use, that consist the employment of fixed sized formats increasingly irrelevant. The growing penetration of web applications and multimedia sharing in every day life should encourage writing applications designers to create facilitations for their integration.

In the field of education and writing, e-learning platforms had become increasingly popular, shifting the focus to the integration of new technologies in the educational practice. Games, simulations, videos, images, animations, charts, diagrams and other resources slowly appear in the classroom demanding their place in educational documents. Moreover, the difficulties of the process and instruction of writing, demonstrates the necessity for the computer to transform from a "slick typewriter" to a writing instrument that provides facilitations for planning, organizing and reviewing. Finally, the emerging web technologies, such as rich-text editing, HTML5, jQuery, CSS3 and MathML, promise a bright future for the web, despite the functional shortcomings and the poor browser implementations. Nevertheless, these technologies are fairly new and the growing adoption will gradually address today's issues.

For the future of MeanWriter, we will continue the journey this time in the classroom, in order to evaluate its significance for building writing skills to students, as well as to create modern quality documents for educational purposes. The main improvement we consider for the future is the imple-

mentation of a equation editor that uses the content-editable attribute and the MathML notation. Another improvement is to provide many more type of charts and diagrams, such as pie charts, while we also consider to integrate more open educational web applications, such as Geogebra, in order to provide a more interactive experience for the user, in other fields beside mathematics. Finally, we will work in the implementation of references for every structural part of the document, including textual and non-textual elements, as well as the incorporation of bibliographic citations.

Bibliography

Abras, C., Maloney-Krichmar, D., and Preece, J. (2004). User-centered design. *Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications*, 37(4):445–56.

Adobe Systems Inc. (1999). *PostScript Language Reference*. Addison-Wesley, 3rd edition.

Adobe Systems Inc. (2012). Adobe indesign help and tutorials. `helpx.adobe.com/pdf/indesign_reference.pdf`. Retrieved in 26/03/2013.

Anderson, J. (2009). *Cognitive Psychology and Its Implications*. Worth Publishers.

Anderson, P. (2007). What is web 2.0?: ideas, technologies and implications for education. Technical report, JISC Technology and Standards Watch, Bristol, UK.

Apache OpenOffice Project (2012). The apache openoffice project announces apache openoffice™ 3.4. `http://www.openoffice.org/news/aoo34.html`. Retrieved in 24/03/2013.

Apple Inc. (2012). Mountain lion available today from the mac app store. `http://www.apple.com/pr/library/2012/07/25Mountain-Lion-Available-Today-From-the-Mac-App-Store.html`. Retrieved in 16/03/2013.

Bereiter, C. and Scardamalia, M. (1987). *The psychology of written composition*. Lawrence Erlbaum Associates Hillsdale, New Jersey.

Bergin, T. (2006a). The origins of word processing software for personal computers. *Annals of the History of Computing, IEEE*, 28(4):32–47.

Bergin, T. (2006b). The proliferation and consolidation of word processing software: 1985-1995. *Annals of the History of Computing, IEEE*, 28(4):48–63.

Berners-Lee, T. and Cailliau, R. (1990). Worldwideweb: Proposal for a hypertext project. *European Particle Physics Laboratory (CERN)*.

Berners-Lee, T., Hendler, J., Lassila, O., et al. (2001). The semantic web. *Scientific american*, 284(5):28–37.

Blackboard Help (2013). Content editor. `https://help.blackboard.com/en-us/Learn/9.1_SP_10_and_SP_11/Student/040_Tools/Content_Editor`. Retrieved in 09/05/2013.

Burnaby, B. (1997). Writing systems and orthographies. *Encyclopedia of language and education*, pages 59–68.

Canonical Ltd. (2006). Canonical launches new ubuntu release for desktops and servers. `http://www.canonical.com/news/610released`. Retrieved in 16/03/2013.

Canonical Ltd. (2013). Ubuntu unveils tablet experience with multi-tasking. `http://www.canonical.com/content/ubuntu-unveils-tablet-experience-multi-tasking`. Retrieved in 16/03/2013.

Ceruzzi, P. (2003). *A history of modern computing*. Mit Press.

Chi-Chen, W. (1930). Notes on chinese ink. *Metropolitan Museum Studies*, 3(1):114–133.

Chinn, M. and Fairlie, R. (2010). Ict use in the developing world: An analysis of differences in computer and internet penetration. *Review of International Economics*, 18(1):153–167.

Clark, R. C. and Mayer, R. E. (2011). *E-learning and the science of instruction: Proven guidelines for consumers and designers of multimedia learning*. Pfeiffer.

Coates, H., James, R., and Baldwin, G. (2005). A critical examination of the effects of learning management systems on university teaching and learning. *Tertiary Education & Management*, 11(1):19–36.

Collins, A., Brown, J. S., and Holum, A. (1991). Cognitive apprenticeship: Making thinking visible. *American Educator*, 6(11):38–46.

Condoor, S. (2004). Importance of teaching the history of technology. In *Frontiers in Education, 2004. FIE 2004. 34th Annual*, pages T2G–7. IEEE.

Coulmas, F. (1989). *The writing systems of the world*. Blackwell Oxford.

Coulmas, F. (2002). *Writing systems: An introduction to their linguistic analysis*. Cambridge University Press.

Dasiopoulou, S., Giannakidou, E., Litos, G., Malasioti, P., and Kompatsiaris, Y. (2011). A survey of semantic image and video annotation tools. In *Knowledge-driven multimedia information extraction and ontology evolution*, pages 196–239. Springer.

Désilets, A., Paquet, S., and Vinson, N. G. (2005). Are wikis usable? In *Proceedings of the 2005 international symposium on Wikis*, pages 3–15. ACM.

Ditch, W. (2007). Xml-based office document standards. *JISC Technology & Standards Watch*, 1(08).

Dix, A., Finlay, J., Abowd, G. D., and Beale, R. (2004). *Human computer interaction*. Pearson Education, 3rd edition.

Drupal website (2013). Comparison of drupal wysiwyg editors. `http://drupal.org/node/208456`. Retrieved in 30/04/2013.

Flower, L. and Hayes, J. R. (1981). A cognitive process theory of writing. *College composition and communication*, 32(4):365–387.

Friedenberg, J. and Silverman, G. (2006). *Cognitive science: An introduction to the study of mind*. Sage.

Garrish, M. (2011). *What is EPUB 3?* O'Reilly Media.

Gartner Inc. (2011). Gartner says worldwide operating system software market grew to $30.4 billion in 2010. `http://www.gartner.com/newsroom/id/1654914`. Retrieved in 16/03/2013.

Gibson, C. and Gibb, F. (2011). An evaluation of second-generation ebook readers. *Electronic Library, The*, 29(3):303–319.

Google Inc. (2009). Introducing the google chrome os. `http://googleblog.blogspot.gr/2009/07/introducing-google-chrome-os.html`. Retrieved in 16/03/2013.

Google Inc. (2012). One click to docs, sheets, and slides. `http://googledrive.blogspot.gr/2012/10/one-click-to-docs-sheets-and-slides.html`. Retrieved in 24/03/2013.

Grajek, S., Lang, L., and Trevvett, D. (2012). The 2011 enterprise application market in higher education. Technical report, EDUCAUSE Center for Applied Research.

Hall, S. P. and Anderson, E. (2009). Operating systems for mobile computing. *Journal of Computing Sciences in Colleges*, 25(2):64–71.

Hitzler, P., Krotzsch, M., and Rudolph, S. (2011). *Foundations of semantic web technologies*. Chapman and Hall/CRC.

Joomla website (2013). Editors. `http://extensions.joomla.org/extensions/edition/editors`. Retrieved in 30/04/2013.

Kellogg, R. T. (1999). *The psychology of writing*. Oxford University Press.

Khalili, A., Auer, S., and Hladky, D. (2012). The rdfa content editor-from wysiwyg to wysiwym. In *Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual*, pages 531–540. IEEE.

Laclavik, M., Šeleng, M., Ciglan, M., and Hluchỳ, L. (2012). Ontea: Platform for pattern based automated semantic annotation. *Computing and Informatics*, 28(4):555–579.

Laleci, G. B., Aluc, G., Dogac, A., Sinaci, A., Kilic, O., and Tuncer, F. (2010). A semantic backend for content management systems. *Knowledge-based systems*, 23(8):832–843.

Lamport, L. (1994). *Latex A Document preperation system*. Addison-Wesley, 2nd edition.

Lee, K.-H., Guttenberg, N., and McCrary, V. (2002). Standardization aspects of ebook content formats. *Computer Standards & Interfaces*, 24(3):227–239.

Lyx website (2013). Lyx – the document processor. `www.lyx.org/`. Retrieved in 26/03/2013.

Maass, W. (2012). *Semantic technologies in content management systems*. Springer.

MacArthur, C. A., Graham, S., and Fitzgerald, J. (2008). *Handbook of writing research*. Guilford Press.

Martin, H. (1995). *The history and power of writing*. University of Chicago Press.

Mayer, R. E. (2005). *The Cambridge handbook of multimedia learning*. Cambridge University Press.

Mayer, S. and Guinard, D. (2011). An extensible discovery service for smart things. In *Proceedings of the Second International Workshop on Web of Things*, page 7. ACM.

MediaWiki Development (2013). Xhtml. `http://www.mediawiki.org/wiki/Xhtml`. Retrieved in 13/04/2013.

Meggs, P. and Purvis, A. (2011). *Meggs' history of graphic design*. Wiley.

Meyrowitz, N. and Van Dam, A. (1982a). Interactive editing systems: Part i. *ACM Computing Surveys (CSUR)*, 14(3):321–352.

Meyrowitz, N. and Van Dam, A. (1982b). Interactive editing systems: Part ii. *ACM Computing Surveys (CSUR)*, 14(3):353–415.

Microsoft Corporation (2010). Microsoft unveils windows phone 7 series. `http://www.microsoft.com/en-us/news/press/2010/feb10/02-15MWC10PR.aspx`. Retrieved in 04/03/2013.

Microsoft Corporation (2012a). Microsoft announces surface: New family of pcs for windows. `http://www.microsoft.com/en-us/news/press/2012/jun12/06-18announce.aspx`. Retrieved in 05/03/2013.

Microsoft Corporation (2012b). Microsoft unveils the new office. `http://www.microsoft.com/en-us/news/Press/2012/Jul12/07-16OfficePR.aspx`. Retrieved in 24/03/2013.

Microsoft Corporation (2012c). Windows 8 arrives. `http://www.microsoft.com/en-us/news/Press/2012/Oct12/10-25Windows8GAPR.aspx`. Retrieved in 06/03/2013.

Milanesi, C. (2011). Ipad and beyond: The future of the tablet market. `http://www.gartner.com/DisplayDocument?doc_cd=217137`. Retrieved in 05/03/2013.

Moodle 2.4 Documentation (2013a). Dragmath equation editor. `http://docs.moodle.org/24/en/DragMath_equation_editor`. Retrieved in 10/05/2013.

Moodle 2.4 Documentation (2013b). Text editor. `http://docs.moodle.org/24/en/Text_editor`. Retrieved in 10/05/2013.

Norman, D. (1988). *The Design of Everyday Things*. Basic.

Open handset alliance (2007). Industry leaders announce open platform for mobile devices. `http://www.openhandsetalliance.com/press_110507.html`. Retrieved in 03/03/2013.

Oren, E., Möller, K., Scerri, S., Handschuh, S., and Sintek, M. (2006). What are semantic annotations. *Relatório técnico. DERI Galway*.

Ormrod, J. (2011). *Human Learning*. Pearson Education, 6th edition.

Patten, K. E. and Campbell, S. R. (2011). *Educational Neuroscience: Initiatives and Emerging Issues*. Wiley-Blackwell.

Pfaffenberger, B., Schafer, S., White, C., and Karow, B. (2004). *HTML, XHTML, and CSS Bible*, volume 145. Wiley, 3rd edition.

Polleres, A. (2010). Semantic web technologies: From theory to standards. In *21st National Conference on Artificial Intelligence and Cognitive Science, NUI Galway*.

Sauer, C. (2006). What you see is wiki-questioning wysiwyg in the internet age. In *Proceedings of Wikimania*, volume 2006.

Schunk, D. H. (2011). *Learning Theories: An Educational Perspective*. Pearson Education, 6th edition.

Shadbolt, N., Hall, W., and Berners-Lee, T. (2006). The semantic web revisited. *Intelligent Systems, IEEE*, 21(3):96–101.

Spantidakis, I. (2010). *Socio-cognitive multimedia learning environments for the production of writing: From Theory to Practice (in Greek)*. Gutenberg editions, Athens.

Spiesser, J. and Kitchen, L. (2004). Optimization of html automatically generated by wysiwyg programs. In *Proceedings of the 13th international conference on World Wide Web*, pages 355–364. ACM.

Stallman, R. M. (2002). Free software, free society: Selected essays of richard m. stallman.

Stephanidis, C. (2009). *The Universal Access Handbook*. CRC Press.

Stork, C., Calandro, E., and Gillwald, A. (2012). Internet going mobile: Internet access and usage in eleven african countries. In *19th ITS Biennial Conference, Bangkok 2012: Moving Forward with Future Technologies-Opening a Platform for All*, number 72503. International Telecommunications Society (ITS).

The Document Foundation (2013a). The document foundation. `http://www.documentfoundation.org/`. Retrieved in 24/03/2013.

The Document Foundation (2013b). Our supporters. `https://www.documentfoundation.org/supporters/`. Retrieved in 24/03/2013.

Tsien, T. (1985). *Science and Civilisation in China: Vol. 5. Chemistry and Chemical Technology. Paper and Printing: Sect. 32/by Tsien Tsuen-Hsuin*. Cambridge University Press.

Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., and Ciravegna, F. (2006). Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *Web Semantics: science, services and agents on the World Wide Web*, 4(1):14–28.

Wang, H., Shea, R., Wang, F., and Liu, J. (2012). On the impact of virtualization on dropbox-like cloud file storage/synchronization services. In *Proceedings of the 2012 IEEE 20th International Workshop on Quality of Service*, page 11. IEEE Press.

Wikimedia Foundation (2013). Wikibooks help - editing. `https://en.wikibooks.org/wiki/Help:Editing`. Retrieved in 13/04/2013.

WikiWizard website (2013). Wikiwizard. `http://www.jspwiki.org/wiki/WikiWizard`. Retrieved in 08/05/2013.

WordPress Support (2013). Wordpress support – writing and editing – editors. `http://en.support.wordpress.com/editors/`. Retrieved in 07/04/2013.

WYMeditor website (2013a). Editors. `http://www.wymeditor.org/`. Retrieved in 30/04/2013.

WYMeditor website (2013b). Wymeditor integration example - custom panel. `http://files.wymeditor.org/wymeditor-1.0.0b2/examples/08-custom-panel.html`. Retrieved in 01/05/2013.