The WYSIWYG Principle in Word Processing

Samuel Rupprechter
Samuel.Rupprechter@student.uibk.ac.at

May 24, 2014

Contents

1 Introduction

This term paper will give a short overview of (software centered) word processing and compare the *"What You See Is What You Get"* principle with the alternative approach of using a document markup language in Section 2. The main focus of this text will lie on criticism in Section 3.

I won't really explain the history of word processing, so if you are more interested in that, please refer to [**?**].

2 Word Processing

This section will introduce two different approaches to word processing, which describes the writing, editing and often printing of documents through the use of a computer system which was designed to support this composition.

2.1 Markup Languages

Markup languages make use of the so called WISIWIT principle (*"What I See Is What I Type"*), which characterizes systems where all you see is plain text, sprinkled with special commands. A compiler will use these to format the text after the author is finished writing.

Many such languages exists—for example HTML or LaTeX.

2.2 The WYSIWYG Principle

WYSIWYG is an acronym for *"What You See Is What You Get"* and means that a computer screen displays content exactly the way it will look like in its finalized form. Most often this content consists of text with some graphics and its finalized form will be a web page or in our case a printed document[1].

This principle was invented to simplify the tedious process of creating documents by compiling them every time a change was made and to save users from having to remember a lot of special code syntax. Instead, the user of such a system gets immediate feedback on all his actions and is therefore always aware of how the final result will look like. He also doesn't need to know any special commands—all he has to do is click at some buttons in the graphical user interface to change the format. If he does not like the way it looks, there is always the back button. Especially for a document of which the exact layout is of critical importance, for example a poster or brochure, this is extremely helpful. Examples of WYSIWYG editors are *Microsoft Word* and *Libre Office*.

3 WYSIWYG Criticism

The WYSIWYG principle often gets criticized for a variety of reasons and I will list and explain some of them. For a more detailed description of these

[1] Note that this does not necessarily imply a hard-copy but could also describe a digital copy of a document.

claims and more please refer to Conrad Taylor's paper [**?**] or Allan Cottrell's elaboration on that subject [**?**].

3.1 Quick and Dirty

Donald E. Knuth created TEX, the typesetting program LATEX uses for formatting its output, because he wanted a higher print quality for his books. This processing is rather complicated, can take quite some time and is not even guaranteed to always completely succeed[2], but if it does, normally the quality of the document leaves nothing to be desired.

WYSIWYG word processors on the other hand use their own typesetting algorithms, but often sacrifice quality for speed, because by their nature they have do display "what you get" in real time. Users would not accept an error message, "just because" one line is a little bit longer then another one. Because of this focus on speed, these editors also cannot use complicated algorithms for things like proper hyphenation of words. Anybody who tries to get correctly justified text in such a program can see the effects of this, as the result is often a rather gruesomely formatted text with lots and lots of non-uniform spaces like this.

3.2 The Illusion of Control

Another reason WYSIWYG gets criticized by writers is the claim that it often does not actually give you enough control, but rather an illusion of it.

An example of this would be the formatting of section titles: At first, it is very easy to make all the titles look nice and alike but if at the end of a 900 page report with maybe 40 titles and 100 subtitles the author suddenly comes to the realization that he wants a different font for all of them, he is faced with a massive problem, because he has to edit all of them by hand.

If he had written the same text and some additional LATEX code in a simple text editor, a single additional command at the beginning would solve his problem. Of course, it has to be mentioned that good modern WYSIWYG editors are catching up and are also providing simplified versions of this feature by now. An example would be *styles* in *Microsoft Office*, although that is nowhere near as advanced as its LATEX-counterpart and more complicated.

[2]The compiler may spit out warnings or errors.

3.3 Distraction

It has also been argued that WYSIWYG word processors, instead of helping the user be more productive, distract users too much and actually make them less productive. Since the editor immediately gives feedback on how the writing will (supposedly) look like when printed, the author is always urged to format the text right away instead of focusing on the words. The appearance of the document becomes more important then its logical structure.

Note that although typesetters like LaTeX do a better job at keeping the structure of the text at the center of attention they can suffer from a similar sort of distraction-introduction: Users still get interrupted by switching between their natural language and the markup language. Finding a complete solution to this problem is probably impossible.

4 Conclusion

Although the WYSIWYG principle certainly has its place in word processing, it is not the be all and end all for any written document and markup languages still have their place in the world. Several attempts to combine the best of both principles have been made, for example in [**?**].