RUMU Editor: A Non-WYSIWYG Web Editor for Non-Technical Users

Eleanor Poley
Knox College
2 East South Street
Galesburg, IL 61401 USA
stellanor@gmail.com

Copyright is held by the author/owner(s).
CHI 2010, April 10–15, 2010, Atlanta, Georgia, USA.
ACM 978-1-60558-930-5/10/04.

Abstract
This paper discusses RUMU Editor, a prototype of a non-WYSIWYG web editor for non-technical users. Users often struggle with WYSIWYG web editors, and the code produced is notoriously bad. RUMU aims to improve the user experience and the resulting websites by providing a simple semantic markup language for the user, changing how styles are applied, simplifying and automating complex aspects of web design, and enabling users to make responsible choices. I conducted usability studies to compare RUMU to iWeb, a WYSIWYG editor, the results of which suggest that users are more satisfied and successful with RUMU. In a case study, two users created real personal websites using iWeb or RUMU. In a blind survey, web designers preferred the websites and source generated by RUMU.

Keywords
Usability-focused development, web editors, non-WYSIWYG interfaces, lightweight markup languages

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation]: User Interfaces—user-centered design; I.7.2 [Document and Text Processing]: Document Preparation—markup languages, hypertext/hypermedia

General Terms
Human Factors, Experimentation, Design

Introduction

It is uncommon for non-technical computer users to publish their own static websites, but those that do almost always use WYSIWYG (What You See Is What You Get) editors. The problems with WYSIWYG web editors are nothing new to the software developers [6, 10]. WYSIWYG may be familiar to users, but the web is never WYSIWYG—"what you see" in your web editor is not always "what you get" across different platforms and browsers. As Gentner and Nielson say, "The problem with WYSIWYG is that it is usually equivalent to WYSIATI (What You See Is All There Is)." [3] Users follow the cues given to them by a software's interface. WYSIWYG web editors invite users to focus on the visual presentation of the web page above all else. Another major problem with WYSIWYG web editors—which remains unseen to the user—is the poor quality of the code that they generate. It doesn't follow web standards and it is generally unreadable and unmaintainable by humans. This can also result in poor accessibility, searchability, and cross-compatibility.

Despite this awareness of WYSIWYG's drawbacks, I was unable to find a web editor that rejects the WYSIWYG paradigm. WYSIWYG may be so synonymous with usability that software developers don't trust themselves—or their users—to move past it. I hypothesize that web editor users don't need to be shielded by a WYSIWYG interface, and that a move towards a non-WYSIWYG interface can improve user experience in addition to the quality of the website and source code output by the application.

RUMU is a software prototype I developed to test this hypothesis. It organizes web pages into content areas. The content can be marked up using a simple language, which I based on Markdown [5]. The user chooses a style template for the entire site, which ensures consistent visual design and encourages the user to focus on content over presentation details. The user can click a button at any time to see a rendered preview of their work-in-progress. The code generated by the program follows web standards, is XHTML Strict 1.0 compliant, and is human-readable. The application makes efforts to simplify the web design process for the user, in part by automating complex or tedious tasks. It also encourages choices that promote good visual design, accessibility, and usability.

I also discuss the small-scale usability studies and long-form case studies I conducted to compare RUMU to iWeb, a WYSIWYG editor. These studies suggest that RUMU provides a superior user experience. Analysis by outside professionals clearly indicates that RUMU produces better source code and may encourage users to create better websites.

Related Work

In “The Anti-Mac Interface”, Nielson and Gentner challenge developers to question some of the modern axioms of usability that were inspired by Apple's Macintosh [3]. In my development, I have focused on their ideas of: representing meaning instead of WYSIWYG, reality over metaphors, richer cues instead of modelessness, and shared control between user and program. I found that this philosophy means violating some of Apple's Human Interface Guidelines [1].

I also considered existing markup languages designed for a less technical audience. Markdown [5, 6, 10] is one of several lightweight markup languages that can be converted to HTML. Wikis are a better-known

example of markup languages trumping WYSIWYG editing—allowing the user to focus on content, not presentation [8]. Researchers working with wikis in the classroom have both praised [8] and lamented [7] this alternative to WYSIWYG web content editing.

```
#First-level heading
##Second level heading
First paragraph, with an
example of *emphasis* and
**strong emphasis**.

> This is a block quote

Single linebreaks
are allowed.

Email addresses are linked
and obfuscated automatically
<sample@example.com>

Here's an internal link
[Link text](Name of Page)

-unordered list element
-another list element

{image_name: Alt Text}
```

Figure 1: Example of the semantic markup language.

Application Design

I developed RUMU for the Mac using Cocoa/Objective-C. One RUMU document represents a website, unlike other editors where each document is an HTML file. A website document consists of any number of web pages. Each page has a user-selectable layout type that dictates the page's content areas. For example, a typical two-column layout would have four content areas: header, sidebar, main content, and footer.

The user enters content in plain text fields using the markup language, which is a customized version of Markdown. The link and image syntax were changed so that users need not understand filenames, paths, and protocols—e.g., internal links are made by referring to the page's name, not its path. I made other changes to suit the needs of non-programmers and make the language simpler and more forgiving. See Figure 1.

The program has options for automation: for example, formatting the header as a heading automatically, making all footers identical, or generating website navigation links on each page. Since consistent navigation is important, and making it is tedious, the latter option is enabled by default. The user interface also includes buttons for inserting some of the more complicated markup, such as links and images. Currently, the user chooses the style for the entire website from a pre-defined list. All of the styles are built on top of the Blueprint CSS Framework [2].

Finally, the application can display a rendered preview of the website-in-progress in a pop-up window. This visual feedback allows users to experiment, learn, and gain confidence. The XHTML and CSS code generated for this preview (and for publishing) is readable, concise, and accessible. Each HTML file is generated by converting each content area's markup to XHTML and placing the XHTML into the document template for its layout. This ensures that each line of code is essentially written by a human: either by the developer who built the template, or by the user who wrote the markup.

Usability Testing and Iterative Development

I followed a usability-oriented iterative development process. Each of three cycles was followed by usability tests with 4 participants. A separate series of tests (n=9) were conducted with iWeb, a WYSIWYG editor developed by Apple, which served as the control in the experiment. I chose iWeb because it also targets a non-technical audience and attempts to simplify web design.

All participants were grouped by self-reported technical skill, and 21 total were randomly selected from a list of volunteers so that all skill groups would be equally represented in the four study groups. All participants—in the three experiment groups and the control group—were asked to complete the same task: to create a two-page website for a student organization with the provided software. All participants were given roughly 20 minutes. The users were only told that the software helps people make websites. Other than an in-program guide to the markup language, no help was given.

I also facilitated two case studies representing more realistic scenarios. Both studies gave non-technical users three hours (split across two sessions) to develop

a website for themselves. One participant used iWeb, the other used RUMU. I strove to be unobtrusive in order to make the situation as realistic as possible. However, I did occasionally help the participants with specific issues after a period of struggle.

Success Level Key

1 *= Did not succeed at task*
2 *= Somewhat succeeded at task*
3 *= Nearly succeeded at task*
4 *= Fully succeeded at task*
5 *= Exceeded expectations for task*

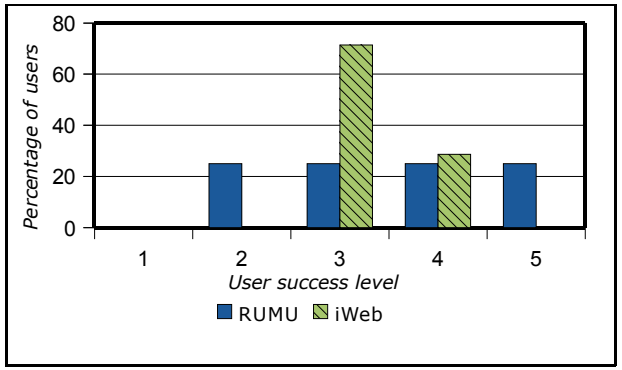

Figure 2: User success assessment.

Users are grouped into success levels based on a rubric of task milestones met. For example, users in group 3 met all but one or two requirements. Especially in iWeb, they usually failed to create a hyperlink. Both iWeb and RUMU users were asked to complete the same task, so the same success assessment was used.

Results of Usability Studies

User Success

Figure 2 graphs the percentage of users who met the requirements of the task in the short laboratory studies[1]. User success improved by the last iteration of my software. The only two iWeb users to fully succeed had exceptional technical skill or prior experience with WYSIWYG web editors. While half of the RUMU users didn't fully succeed in the allotted time, half did succeed or exceeded expectations. In the case studies, both the iWeb and RUMU participants succeeded in creating an 8-10 page static website for themselves within the three hour time limit. Both successfully familiarized themselves with their respective software.

[1] Only 19 success evaluations were conducted because two participants' video recordings were corrupted. Both used iWeb.

User Satisfaction

In the post-study satisfaction survey, users were asked to rate on a seven point scale: "How satisfied were you with the experience of making a website with this software?" and "If you were to make a website again, how likely would you be to use this software?" Figure 3 graphs responses to the first question by participants in the short laboratory studies. While RUMU users were overall somewhat more satisfied, they were slightly less likely to want to use the software again. I suspect this is because my application was a bare-bones prototype during testing; users did not perceive that there were more features worth exploring in the future. In the case studies, both users were asked the same questions. Both were surprised and satisfied to have made their own websites, but the RUMU user rated his overall satisfaction and his likeliness to use the software again higher than the iWeb user (scores of 7 and 7 versus 5 and 6.) The RUMU user was excited at the end, expressing pride in learning the software and making his own website. The iWeb user said, "It's not as complex as I thought, but I had more trouble with small things than I expected."

Observations

I observed that RUMU's interface encouraged users to focus on writing content, as I expected. Most users caught on quickly to the markup language, and the case study participant ended up very comfortable using it. Some users wanted to be able to exercise more control over the presentation, but most didn't mind, especially for such a short task. Some users didn't understand the organization, but this was resolved by later interface improvements. iWeb users struggled most with getting presentational elements, especially text boxes, to cooperate with their wishes; most also

User Satisfaction Ratings Key

1 *= Not at all satisfied with the experience of using the software*

…

7 *= Extremely satisfied with the experience of using the software*

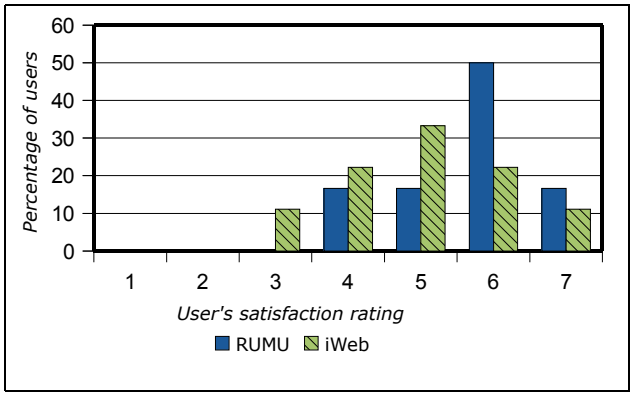

Figure 3: User satisfaction ratings from lab usability studies

struggled to make hyperlinks. In the case study, I noticed that the iWeb user was less hesitant to start experimenting, likely because the WYSIWYG interface was familiar. The RUMU user struggled more initially, but soon "got it" and became comfortable. He was very confident and efficient by the end of the session, whereas the iWeb user still had "trouble with small things." I also observed that iWeb encouraged bad design behavior, such as inserting extra whitespace to align text. RUMU tries to enforce good choices, but I discovered that the application's good intentions were sometimes limited by the user's willingness play along.

Web Designers' Analyses

I asked a small group of people proficient in HTML and CSS to examine the two websites produced in the case studies. In a blind survey, they were asked to look at three websites and their source code: the one from iWeb, the one from RUMU, and a control that I had coded by hand. For each website, the web designers were asked to rate (on a five-point scale) the readability and quality of the source code, the general adherence to web standards, the semantic quality of the source code, and the overall quality of the rendered website. They were also asked to guess how each website was created. All participants gave the RUMU scores of 4 and 5 in all categories, comparable to the scores given to the hand-coded control. The majority thought that the RUMU site had been coded by hand by a web designer. The iWeb site, on the other hand, was identified by all as the product of a WYSIWYG editor. It received mostly scores of 1 and 2 in all categories.

Accessibility

I ran both websites through an automated accessibility test [9], and RUMU outperformed iWeb. RUMU focuses on semantic code and enforces certain accessibility details (such as requiring alt text on all images). iWeb commits several blatant accessibility crimes, such as converting a page's text to an image if the font the user chooses is not web-safe. However, it has been shown that automated accessibility tests are not very comprehensive [4].

Conclusions

Based on the results of my usability studies, I have found that a non-WYSIWYG editor can compete with a WYSIWYG editor in terms of user experience. The case studies demonstrated that RUMU can be more difficult to use at first, but overall the learning curve is more straightforward than iWeb's. My data and observations suggest that RUMU users are more satisfied and successful, but with a small sample and limited tests, I cannot be certain that it provides a superior user experience. It is clear from the web designers' analyses, however, that RUMU helps users create websites with better source code. It proved worthwhile to put the user more in control of the content and semantics, less in control of complicated or common

aspects of web design, and never in control of pixel-level presentational details.

Limitations

RUMU has always been a prototype—it is not fully featured or fully stable. My sample groups were small, and all participants were college students who were more educated and computer literate than average. Finally, satisfaction surveys, success evaluations, and my observations are not perfect or unbiased measures.

Future Work

RUMU does not have all the features that I imagine for a successful non-WYSIWYG editor. Non-WYSIWYG web editors should be further explored as a way to empower more users to author websites that would satisfy the standards of web professionals. Other approaches are necessary to determine how best to approach this problem, and how to educate users about the importance of semantic markup. Lightweight markup languages like Markdown are not well-studied; developing my language further could be a research project of its own. Finally, while RUMU takes a stride towards a more semantic web, it is limited by users' understanding of semantics, and even more by XHTML's lack of semantic richness.

Acknowledgements

I am grateful to my advisors: Don Blaheta, Alex Varakin, David Bunde, and Juan Pablo Hourcade. I also thank Kyle Sibley, Matt Baker, and Dave Ross, and other web designers who participated in the survey. This project was funded by Knox College's Ford Reasearch Fellowship, Richter Memorial grants, and Ron Asplund grant.

References

[1]: Apple Human Interface Guidelines. http://developer.apple.com Apple Inc., 2009.

[2]: Blueprint CSS. http://www.blueprintcss.org

[3]: Gentner, D. and Nielsen, J. 1996. The Anti-Mac interface. *Commun. ACM* 39, 8 (Aug. 1996), 70-82.

[4]: Mankoff, J., Fait, H., and Tran, T. 2005. Is your web page accessible?: a comparative study of methods for assessing web page accessibility for the blind. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. (2005), 1-50.

[5]:Markdown. http://daringfireball.net/projects/markdown

[6]: Rankine, A. This is what you see. This is what you get. 2005. http://girtby.net/archives/2005/6/13/this-is-what-you-see-this-is-what-you-get/

[7]: Schwartz, L., et. al. (2004). Educational Wikis: features and selection criteria. *International Review of Research in Open and Distance Learning*, 5(1).

[8]: Wang C., Turner D. Extending the wiki paradigm for use in the classroom. *Proceedings of the International Conference on Information Technology: Coding and Computing.* Vol. 2, p.255, April 05-07, 2004

[9]: WAVE. http://wave.webaim.org/ WebAIM, 2009.

[10]: What it is, not what it looks like. http://www.atendesigngroup.com/blog/what-it-not-what-it-looks_Aten Design Group, 2008.