Published in P. Isaias and M.B. Nunes (Eds.) Proceedings of the IADIS International Conference WWW/Internet 2005, Volume II, pp. 247-254. IADIS Press, ISBN 972-8924-02-X.

HEURISTIC EVALUATION OF CONTENT MANAGEMENT SYSTEMS: CMS SPECIFIC HEURISTICS

Ronald Bos
Institute of Information and Computing Sciences, Utrecht University
P.O. Box 80089, 3508 TB Utrecht, the Netherlands
rbos@cs.uu.nl

Jilles van Gurp
GX Creative Online Development B.V.
Wijchenseweg 111, 6538 SW Nijmegen, the Netherlands
jilles@gx.nl

Jan Herman Verpoorten, Sjaak Brinkkemper
Institute of Information and Computing Sciences, Utrecht University
P.O. Box 80089, 3508 TB Utrecht, the Netherlands
{janherm, s.brinkkemper}@cs.uu.nl

ABSTRACT

Heuristic evaluation is a cheap and effective method to find usability problems in a user interface. However, a common set of heuristics does not cover domain specific characteristics. In this paper, the heuristics used in this method are extended by presenting twelve content management system (CMS) specific heuristics. These domain specific heuristics will help evaluators better during a heuristic evaluation of a content management system than existing sets of heuristics. Furthermore, these heuristics can be followed as a "rule of thumb" during the design of the user interface of CMS.

KEYWORDS

WWW/Internet Applications, Web Software, Human Computer Interaction, Usability.

1. INTRODUCTION

Heuristic evaluation is a cheap and effective method for finding usability problems in a user interface. However, existing heuristics are aimed at user interfaces in general. A content management system (CMS), being a specific instance of a web application, is restricted in the way it can interact with its users. Consequently, Nielsen's (1994) set of heuristics – the most pervasive set – is most likely not the most appropriate and successful one to use with a heuristic evaluation of a CMS. In this research a set of CMS specific heuristics is developed and validated that can better guide evaluators in finding usability problems while performing a heuristic evaluation of a CMS.

The rest of this paper is organized as follows. This section introduces the need for CMS specific heuristics. Section 2 describes the method used to develop and validate the heuristics. In section 3 the results from the study and the CMS specific heuristics are described. In section 4 the conclusions are stated.

1.1 Heuristic Evaluation

The most widely used method to improve software usability is evaluating the system in order to find its usability problems. Heuristic evaluation (Nielsen and Molich, 1990) is a usability evaluation method which can diagnose potential usability problems in a user interface. It can be described as "having a small set of evaluators examine the user interface and judge its compliance to recognized usability principles (the heuristics)" (Nielsen, 1992). It is low cost in terms of time and resources; it can be completed in just a few

hours per evaluator. In theory, only 3–5 experienced evaluators are needed to identify ~75–80 percent of the usability problems in the interface (Nielsen, 1992). The heuristics that Nielsen (1994) proposed are based on a factor analysis of 101 usability principles accounting for 249 usability problems and have evolved from the set of heuristics originally posted by Nielsen and Molich (1990). In contrast to user testing, heuristic evaluation does not involve end-users of the system.

The main advantages of heuristic evaluation over other usability evaluation methods can be summarized as: It has the best cost/benefit ratio (Jeffries et al., 1991); it is intuitive and it is easy to motivate people doing it (Nielsen and Molich, 1990); it has a high success rate: only a few evaluators are necessary (Danino, 2001).

1.2 Content Management Systems

Organizations struggle to maintain their increasingly complex websites (Browning & Lowndes, 2001). They contain more and more content, and technical possibilities are increasing (e.g., connections to other systems). This necessitates a good organization of the management of the website's content. A CMS does just that. Content management can be defined as "the process of managing electronic content through its lifecycle – from creation, review, storage and dissemination to destruction" (Parapadakis, 2000). A CMS is a system that supports this process, or: "A system that supports the creation, management, distribution, and publishing of corporate information, covering the complete life-cycle of a website" (Robertson, 2003, adapted).

The majority of CMSs are a specific instance of web applications, defined as "any software application that depends on the Web for its correct execution" (Gellersen & Gaedke, 1999). They are delivered over the internet through a web browser, posing a number of restrictions on the way the application can interact with the user. For example, direct manipulation is hard to realize. In general, web applications are limited to types of interaction that current web browsers offer. But web applications are also different from normal websites, being more interactive, and consisting of more complex and unique interactions (Wroblewski & Rantanen, 2001). Because of the specific restrictions on the way a CMS interacts with its users, Nielsen's (1994) set of heuristics is most likely not the most appropriate one to use with a heuristic evaluation of a CMS.

1.3 Development of Domain Specific Heuristics

The assumption is made that a set of CMS specific heuristics can better help evaluators find usability problems of CMSs during a heuristic evaluation. This assumption was validated by performing a heuristic evaluation of GX WebManager, a CMS developed by GX Creative Online Development. This CMS is one of the top CMSs in the Dutch market; many users use the system, and some of the largest Dutch websites are maintained with GX WebManager. No significant usability issues were known for the system. If the assumption is correct, evaluators would find a relevant number of usability problems that were insufficiently addressed by Nielsen's (1994) heuristics. This relevance was inspected both quantitatively – the number of usability problems reported per heuristic – and qualitatively – the nature of the problems reported.

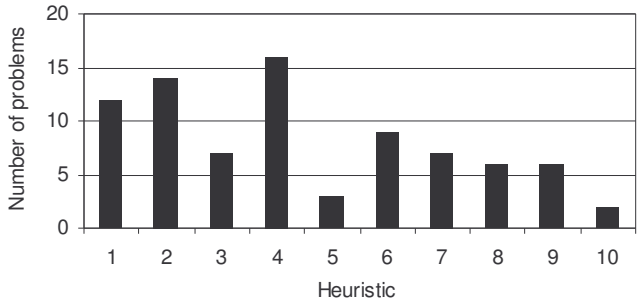

Figure 1. Number of usability problems reported per heuristic

The heuristic evaluation (n = 7) yielded 82 unique usability problems. The number of problems reported per heuristic is depicted in Figure 1. As this figure indicates, some of Nielsen's (1994) heuristics yielded more usability problems than others (M = 8.2; sd = 4.3). If all heuristics were equally useful in finding

usability problems, the number of usability problems found per heuristic would have been more balanced. Of course, this data is based on a heuristic evaluation of just one CMS. Since one system is not likely to contain a balanced number of usability problems per heuristic, no final conclusions can be drawn from this data. It is, however, an indication that not all the heuristics are equally successful. After all, GX WebManager is a widely used system, but no significant usability issues were reported by its users.

Qualitative analysis also indicated that Nielsen's (1994) heuristics did not suffice for evaluating CMSs. For example, heuristic 3 (user control and freedom) yielded several problems caused by the fact that the CMS runs in a web browser. A new heuristic might be added that specifically addresses this kind of problems, thereby better helping evaluators finding usability problems of that nature.

2. METHOD

The two main steps in this research are the development of the CMS specific heuristics and the validation of these heuristics.

2.1 Development of Heuristics

The following sources were used to develop the set of CMS heuristics:

1) The results of the heuristic evaluation were quantitatively and qualitatively analyzed. These results gave an indication of which of Nielsen's (1994) heuristics were useful in helping evaluators find usability problems of a CMS. The nature of the problems reported was used to determine which heuristics could better be recategorized or adjusted, or what heuristics should be added to the set;
2) Results from a task analysis were used to determine specific users' needs and specific characteristics of CMSs. This task analysis consisted of semi-structured interviews with four CMS end-users, artifact analysis (i.e., manuals, training documents, the CMS itself), and user observation (i.e., observing users while they are working with the system);
3) Usability guidelines available in the literature (e.g., Smith & Mosier, 1986; Mayhew, 1992; Wheinshenk & Yeo, 1995) that were applicable to CMS were used to adjust the heuristics.

2.2 Validation of Heuristics

Before domain specific heuristics can be used for a heuristic evaluation, they need to be validated. In several studies heuristics are validated by comparing the performance of evaluators using the new set of heuristics against evaluators using another set of heuristics (e.g., Baker et al., 2002; Mankoff et al., 2003; Sutcliffe & Gault, 2004). Even though at first sight this is the best method, but as Nielsen (1994) indicates, it is impossible for the evaluators to not use their additional usability knowledge. So in reality evaluators would apply certain heuristics they are not supposed to use. This would bias the results.

Another method to validate domain specific heuristics is expert validation. This method is also used in several studies (e.g., Mankoff et al., 2003; Grenville, 2001; Somervell & McCrickard, 2004). In this study we also used expert validation to validate – and when necessary adjust – the heuristics. Expert validation is chosen in favor of comparing the performance of evaluators using this set of heuristics against evaluators using another set of heuristics, because of the potential bias when using the latter method. Practical constraints were also in favor of expert validation

Nine experts participated in the validation of the set of heuristics. Six also participated in the heuristic evaluation of GX WebManager. They are competent in evaluating theheuristics since they have experience with performing a heuristic evaluation of a CMS. Therefore they are able to rate the applicability and relevance of the heuristics in finding usability problems in CMSs. Three experts are CMS user interface designers. They do not have experience with performing a heuristic evaluation, but they are experienced in designing the user interface of a CMS and can therefore provide valuable feedback on the heuristics.

The participants were surveyed using an online questionnaire. This questionnaire listed all the CMS specific heuristics, and participants were asked to rate the items on the following 5-point scales:

1) Relevance: How relevant do you think this heuristic is for the discovery of usability problems during a heuristic evaluation of a CMS user interface? (not relevant at all – very relevant);

2) Importance: How important do you think this heuristic is to follow as a "rule-of-thumb" during the design of a CMS user interface? (not important at all – very important);
3) Violation: How often do you think this heuristic is violated in CMS user interfaces? (almost never – very often).

Also, participants were asked to provide comments on each of the heuristics and to mention heuristics that were missing from the set.

3. RESULTS

Twelve CMS specific heuristics resulted from this research. These heuristics are listed in Table 1 and are explained and related to Nielsen's set of heuristics below.

Table 1. CMS specific heuristics

1: Visibility of system status. The system should always keep the user informed about what is going on, providing feedback about the fulfillment of every action (users should not have to rely on the browser's status and progress bar). Provide real-time status indicators when executing actions. Feedback messages should state (1) what the computer has done and (2) whether it was successful. Options that are unavailable at that time or on the selected object should be greyed out. Indicate the progress throughout a task (e.g., "step 4 of 6") whenever appropriate.
2: Match between system and real world. The systems should speak the user's language. Follow real-world conventions, making information appear in a natural and logical order. Data flows should support and follow the user's task flows and goals. Menus should be organized in a logical way, and be phrased as commands to the computer.
3: Consistency. Be consistent through all aspects of the interface; the user should not have to wonder whether different words, situations, or actions mean the same thing.
4: Recognition rather than recall. Objects, actions, and options should be visible. The user should not have to remember information from one part of the dialogue to another. Default values should be used whenever a likely default value can be defined. Hyperlinks and controls should be easy to distinguish.
5: Flexibility and efficiency of use. Accelerators –unseen by novice users– should be used to speed up the interaction for the expert user. Industry-standard shortcuts should be used. Consider implementing direct manipulation/drag and drop functionality and context-menus wherever a user might expect this. Provide the possibility to edit in WYSIWYG-mode whenever possible and appropriate. Make sure the web application loads quickly enough to keep response time acceptable.
6: Aesthetic and minimalist design. Dialogues should not contain irrelevant or rarely needed information. Limit the use of windows, but use pop-ups when appropriate. Overuse of features could cause a decrease of usability. When appropriate, complicated dialogues should be hidden from casual users by providing a "beginner" and "expert" version of the interface. If a display contains too much information, partition it into separate displays.
7: Error prevention and recovery. Provide the user with an "emergency exit" to leave any unwanted state. Support undo and redo for every action. Allow users to revert to previous versions of pages/data. Error messages should be expressed in plain language (no code; hide technical information), precisely indicate the problem, and constructively suggest a solution. Protect users from actions with drastic consequences (ask for confirmation). Provide feedback on invalid input and suggest subsequent actions. Do not let users execute actions that lead to errors.
8: Provide help and instructions. Provide online and context-sensitive help. Rather integrate instructions into the application (e.g., through the use of labels or tooltips) than relying on help and documentation. Descriptive instructions, labels, and tooltips should be available whenever appropriate.
9: Conformance to other applications. The system should be consistent with other applications (both desktop and web applications) and not contain conflicting elements. Industry standard controls and task flows should be used. When controls are copied from desktop applications, they should work in the same way as they do in the desktop application (and include all its features). If necessary to overcome the browser's shortcomings, use plug-ins.
10: Follow web application conventions. Avoid the use of double clicks. Use roll-overs to display additional data in order to reduce screen clutter, but never hide primary controls or content. Use standard web controls (i.e., radio buttons, checkboxes, etc.) and flows (e.g., the deletion of elements from a list). When using text as a hyperlink, underline it.
11: Browser controls and navigation. The use of the browser's controls (e.g., the back button) or (accidentally) closing the browser should not result in data loss (save changes when appropriate). Make a clear distinction between the browser's and the web application's menus and controls. If the browser is not opened full-screen, the browser's controls should not cause errors or confusion.
12: Allow easy data entry that minimizes the chance of errors. Various types of input should be recognized (e.g., "1", "Jan", and "January" for January, "555-5555" and "5555555" for telephone numbers, etc.). Provide easy means for the entry of restricted data (e.g., a calendar view for the input of dates, a color picker for the selection of colors, etc.).

Provide a direct manipulation interface for novice users, but make sure it works as the user expects. Always show the data a user has entered. Indicate which data fields are required. Use appropriate data entry controls for the task at hand.

3.1 Omitted Heuristics

Two of Nielsen's (1994) original heuristics were omitted from the set of CMS specific heuristics or incorporated into other heuristics. Nielsen's heuristic 3 (user control and freedom) had problems reported related to the fact that a CMS runs within an internet browser. This is now covered by a specific heuristic regarding the browser's windows and controls (see heuristic 11). Issues regarding emergency exits, undo, and redo that this heuristic covered are incorporated into a heuristic regarding error prevention (heuristic 7). Few problems were reported in reference to Nielsen's heuristic 5 (error prevention). As error prevention and error recovery are closely related, this heuristic is merged with Nielsen's heuristic 9 (see heuristic 7).

3.2 Adapted Heuristics

Eight of Nielsen's (1994) original heuristics were adapted or combined to form CMS specific heuristics.

3.2.1 Heuristic 1: Visibility of System Status

Nielsen's heuristic 1 (visibility of system status) yielded relatively many usability problems. It is elaborated with details on feedback on the fulfillment of actions (Smith & Mosier, 1986; Microsoft, 1995) and on the contents of feedback messages (Weinshenk & Yeo, 1995). Results indicated that users cannot rely on the browser's status and progress bar, and that unavailable options should be greyed out (cf. Apple, 2004; Mayhew, 1992). Also, as task analysis indicated, system response time could increase unexpectedly due to high network traffic. Therefore, real-time status indicators and progress throughout a task should be provided when appropriate (Smith & Mosier, 1986).

3.2.2 Heuristic 2: Match between System and Real World

The nature of the – relatively many – problems reported with Nielsen's heuristic 2 (match between system and real world) indicate that speaking the users' language is an important part of this heuristic. Two related items were added to improve detection of problems. Data flows should support the user's task flows (e.g., Mayhew, 1992). Also, statements on the organization of menu items are added (e.g., Apple, 2004; Mayhew, 1992); not all evaluators detected such problems, possibly because the heuristics did not explicitly mention it.

3.2.3 Heuristic 3: Consistency

Nielsen's heuristic 4 (consistency and standards) yielded the most usability problems, nearly all regarding internal consistency. Consequently, this heuristic is slightly adjusted to cover only internal consistency. Conformance to other applications (i.e., external consistency), is included as a separate heuristic (heuristic nine) to improve the chance of evaluators finding such problems. Task analysis showed that external consistency is important, because most CMS users also intensively use other software like Microsoft Word. For clarity, the phrase "be consistent through all aspects of the interface" (Microsoft, 1995) is added.

3.2.4 Heuristic 4: Recognition Rather Than Recall

Nielsen's heuristic 6 (recognition rather than recall) is only slightly adjusted. Users should not have to remember information between dialogues (cf. Smith & Mosier, 1986; Koyanl et al., 2003). Statements on instruction are moved to heuristic 8, as instructions are more related to help than to this heuristic. The suggestion to use default values (e.g., Mayhew, 1992; Smith & Mosier, 1986) and the idea that hyperlinks and controls should be easily distinguishable were added. Some problems related to hyperlinks as controls were reported. Explicitly mentioning this improves the finding of such problems.

3.2.5 Heuristic 5: Flexibility and Efficiency of Use

Nielsen's heuristic 7 (flexibility and efficiency of use) is shortened. Because accelerators are often shortcuts, it is explicitly stated that shortcuts should be industry-standard, which evaluators reported to not always be the case. The evaluators also mentioned that somtimes they missed direct manipulation features. Therefore it

is stated that direct manipulation should be considered. Last item added regards system response time, important because web applications load via a network, which can cause long loading and response times.

3.2.6 Heuristic 6: Aesthetic and Minimalist Design

Some related items were added to Nielsen's heuristic 8 (aesthetic and minimalist design) to accommodate web application specific characteristics. The use of windows should be limited (Wroblewski & Rantanen, 2001), but when appropriate pop-ups should be used (cf. Smith & Mosier, 1986). Overuse of features could cause screen clutter and decrease the system's usability. Complicated dialogues should be hidden from beginner users if possible and appropriate (Mayhew, 1992), and information-excessive screens or dialogues can often better be separated into several screens (Smith & Mosier, 1986; Koyanl et al., 2003).

3.2.7 Heuristic 7: Error Prevention and Recovery

Nielsen's heuristics 9 (help users recognize, diagnose, and recover from errors), heuristic 5 (error prevention), and parts of heuristic 3 (user control and freedom) were merged, combining into one heuristic on error prevention and recovery. For CMSs a special form of undo is to revert back to a previous version of a page, as indicated by the task analysis. Users should be protected from actions with drastic consequences by asking for conformation (Mayhew, 1992; Smith & Mosier, 1986). Results showed that, upon invalid data entry, feedback and suggestions should be provided (cf. Smith & Mosier, 1986; Microsoft, 1995). Actions leading to errors should be unavailable (Apple, 2004; Mayhew, 1992).

3.2.8 Heuristic 8: Provide Help and Instructions

Nielsen's heuristic 10 (help and documentation) is adjusted to facilitate web application specific demands and characteristics. In web applications, context-sensitive and especially online help are probably the best choice. Even better is to integrate instructions into the application, by using descriptive labels and tooltips whenever appropriate (Wroblewski & Rantanen, 2001; Mayhew, 1992; Smith & Mosier, 1986).

3.3 Extra Heuristics

Besides the adaptations of Nielsen's heuristics, five specific heuristics are added to the set.

3.3.1 Heuristic 9: Conformance to Other Applications

This heuristic regards the consistency (or conformance) of the system to other applications. As Nielsen's heuristic 4 (consistency and standards) mostly yielded problems regarding internal consistency, a heuristic regarding the conformance to other applications was desirable. It states that the system should be consistent with other – both desktop and web – applications the user might be used to work with. It should certainly not contain conflicting elements (e.g., Mayhew, 1992). Results indicated that industry-standard controls and task flows should be used. Plug-ins can overcome browser's shortcomings (Wroblewski & Rantanen, 2001).

3.3.2 Heuristic 10: Follow Web Application Conventions

Although covered by Nielsen's heuristics ("follow platform conventions", heuristic 3), this heuristic is included as a separate heuristic. Since CMSs run in an internet browser and most users are accustomed to working with web applications and interactive websites, their conventions should be followed: single rather than double clicks; use roll-overs to display additional data and to reduce screen clutter; use standard controls (e.g., radio buttons, checkboxes) and flows (e.g., deletion of elements from a list) that are familiar to users (Wroblewski & Rantanen, 2001). Last, when text is used as control or hyperlink, it should be underlined to make it easily recognizable (Wroblewski & Rantanen, 2001).

3.3.3 Heuristic 11 Browser Controls and Navigation

This heuristic is also related to the fact that web applications run in an internet browser. It is important to keep in mind that the browser, with its menus and navigational controls, takes up a prominent space of the screen. Browser's controls and navigation (e.g., back button, menus) can cause data loss in web applications. This should be prevented, and the browser's controls should be easily distinguishable from the web application's controls (Wroblewski & Rantanen, 2001), especially if the browser is not opened full-screen.

3.3.4 Heuristic 12: Allow Easy Data Entry That Minimizes the Chance of Errors

Data entry is a significant part of a CMS. Even though to some extend covered by Nielsen's heuristics, it is not explicitly mentioned, and hence added as an extra heuristic. Various types of input should be recognized by the system to prevent errors (Mayhew, 1992), and for entry of restricted data easy means should be provided (e.g., calendar view for date input) (Smith & Mosier, 1986). When appropriate, a direct manipulation interface should be provided (Apple, 2004; Microsoft, 1995). Entered data should always be visible (Kayonl et al., 2003), and data fields that are required should be clearly indicated as such (Koyanl et al., 2003; Smith & Mosier, 1986). Lastly, appropriate data entry controls should be used for the task at hand (e.g., Apple, 2004; Microsoft, 1995; Smith & Mosier, 1986).

3.4 Results Expert Validation

The average ratings the experts gave for each of the heuristics are listed in Table 2. As this table shows, on average the experts gave a high relevance, importance, and violation score to all the heuristics. Agreement between experts was moderate (Cronbach's $\alpha = 0.55$).

Table 2. Ratings from expert validation (*sd* in parentheses)

Heuristic	Relevance	Importance	Violation
1	4.33 (0.87)	4.22 (0.67)	3.89 (1.05)
2	3.89 (1.05)	4.22 (0.83)	3.67 (1.00)
3	4.67 (0.50)	4.33 (0.50)	3.33 (1.12)
4	4.00 (0.87)	3.89 (0.78)	3.56 (0.73)
5	3.56 (0.73)	3.78 (0.97)	3.33 (1.12)
6	3.78 (1.09)	4.00 (1.12)	3.78 (1.30)
7	4.00 (1.00)	4.11 (1.05)	3.89 (1.45)
8	3.78 (0.67)	3.67 (0.87)	3.67 (1.00)
9	3.11 (0.78)	2.89 (0.93)	3.67 (0.87)
10	3.11 (1.45)	3.11 (1.17)	3.11 (0.78)
11	3.22 (1.20)	2.89 (1.27)	3.67 (1.22)
12	3.33 (1.00)	3.22 (0.97)	3.22 (1.20)

One heuristic was adjusted after analyzing the results of the expert validation. Heuristic 10 originally was "Follow web (application) conventions". This was the most controversial heuristic and scored relatively low with regard to relevance ($M = 3.11$; $sd = 1.45$). Experts giving a low relevance score commented on the fact that the heuristic referred to web conventions while a CMS is a much richer application. It was therefore decided to remove the parentheses around the word "application", changing the focus to web application conventions and not specifically mentioning web conventions anymore.

The experts found heuristic 11, "Browser controls and navigation", the least important to follow as a "rule of thumb" during the design of a CMS user interface, albeit with little agreement (2.89; $sd = 1.27$). However, when further analyzing the results, it appears that the CMS user interface designers gave a lower importance rating to this heuristics than other, albeit not significant (t (7) = 0.919; $p = .388$). One of them commented that following this heuristic would cost quite a lot of resources. So it appears that the CMS user interface designers see this as a trade-off, and consequently following this heuristic as less important. Nevertheless, this heuristic is left unaltered, since the cost of solving found problems is irrelevant to the finding of them during heuristic evaluation.

In general, the experts gave all heuristics a satisfying score on relevance (3.73), importance (3.69), and violation (3.56).

4. CONCLUSION

CMSs differ from desktop applications in several ways. Most importantly, because they are a specific instance of web applications, they are restricted in their interaction with the users. Also, as CMSs have to be

delivered via the internet, system response time can increase unexpectedly. Next, CMSs have a strong focus on data entry, and the user's mental model shows similarities to other text editing environments.

Because of these differences, original sets of heuristics are not the most applicable. Therefore, in this study a set of CMS specific heuristics has been developed and validated.

The usefulness of the set of CMS specific heuristics that have been developed in this study is twofold:

1) It forms the basis of a cheap and effective usability evaluation method. It better helps evaluators in finding usability problems during the heuristic evaluation of a CMS. There is considerable potential for using the method to improve the usability of CMSs (cf. Baker et al., 2002);
2) It can be used as a set of "rules of thumb" to be followed during the user interface design of CMSs. By following these heuristics, usability problems can be prevented.

The CMS heuristics were validated by means of expert validation. Experts rated the heuristics developed in this study as: Relevant for the discovery of problems during heuristic evaluation of CMSs; important to follow as a "rule of thumb" during the user interface design of a CMS; and often violated in CMS user interfaces.

Practical application should further prove the value of these heuristics – this is beyond the scope of this research.

REFERENCES

Apple, 2004. *Apple Human Interface Guidelines*. Available from http://developer.apple.com/documentation/UserExperience/Conceptual/OSXHIGuidelines/ [Accessed 22 March 2005].

Baker, K. F. et al, 2002. Empirical development of a heuristic evaluation methodology for shared workspace groupware. *Proceedings of ACM CSCW'02*, pp. 96-105.

Browning, P., & Lowndes, M., 2001. *JISC TechWatch Report: Content Management Systems*. Available from http://www.jisc.ac.uk/uploaded_documents/tsw_01-02.pdf [Accessed 8 May 2005].

Danino, N., 2001. *Heuristic Evaluation – a Step by Step Guide*. Available from http://www.nickydanino.com/articles.shtml [Accessed November 30, 2004].

Gellersen, H. W., and Gaedke, M., 1999. Object-oriented web application development. *IEEE Internet Computing 3*(1), pp. 60-68.

Grenville, N. D., 2001. *Developing Heuristics to Optimize the Configuration of the Video-Mediated Environment*. Thesis. (PhD) Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Jeffries, R. et al, 1991. User interface evaluation in the real world: a comparison of four techniques. *Proceedings of ACM CHI'92*, pp. 119-124.

Koyanl, S. J. et al, 2003. *Research-Based Web Design & Usability Guidelines*. Available from http://usability.gov/pdfs/guidelines.html [Accessed 22 March 2005].

Mankoff, J. et al, 2003. Heuristic evaluation of ambient displays. *Proceedings of ACM CHI'03*, pp. 169-176.

Mayhew, D. J., 1992. *Principles and Guidelines in Software User Interface Design*. Englewood Cliffs: Prentice-Hall.

Microsoft Corporation, 1995. *The Windows Interface Guidelines for Software Design*. Redmond: Microsoft Press.

Nielsen, J., 1992. Finding usability problems through heuristic evaluation. *Proceedings of ACM CHI'92*, pp. 373-380.

Nielsen, J., 1994. Enhancing the explanatory power of usability heuristics. *Proceedings of ACM CHI'94*, pp. 152-158.

Nielsen, J., and Molich, R., 1990. Heuristic evaluation of user interfaces. *Proceedings of ACM CHI'90*, pp. 249-256.

Parapadakis, G., 2000. What's in a name? *AIIM E-DOC Magazine*, november/december 2000. Available from http://www.edocmagazine.com/archives_articles.asp?ID=18350 [Accessed March 23, 2005].

Robertson, J., 2003, June 3. So, what is a content management system? *KM Column*. Available from: http://www.steptwo.com.au/papers/kmc_what/pdf/KMC_What.pdf [Accessed 30 October 2004].

Smith, S. L., and Mosier, J. N., 1986. *Guidelines for Designing User Interface Software*. Bedford: MITRE Corporation.

Somervell, J., and McCrickard, D. S., 2004. Comparing generic vs. specific heuristics: illustrating a new UEM comparison technique. *Proceedings of HFES'04*, pp. 2480-2484.

Sutcliffe, A., and Gault, B., 2004. Heuristic evaluation of virtual reality applications. *Interacting with Computers (16)*, pp. 831-849.

Weinshenk, S., and Yeo, S. C., 1995. *Guidelines for Enterprise-Wide GUI Design*. New York: John Wiley & Sons.

Wroblewski, L., and Rantanen, E. M., 2001. Design considerations for web-based applications. *Proceedings of HFES'01*, pp. 1191-1195.