MANUAL CODING VERSUS WIZARDS: EXAMING THE EFFECTS OF USER INTERACTION ON MARKUP LANGUAGE PERFORMANCE AND SATISFACTION

Jeffrey Hsu, Information Systems, Silberman College of Business Administration, Fairleigh Dickinson University, 285 Madison Avenue, Madison NJ 07940, jeff@fdu.edu

ABSTRACT

The role of the user interface(mode of interaction) can have an impact on both performance and satisfaction with regards to using a programming language--a well-designed user interface can improve performance and satisfaction, The impact of interaction mode, in this case command-based coding, versus using a form-fill-in wizard, is examined, with respect to performance and satisfaction while performing a survey-oriented task. Wizards brought about better performance than using the command language, and the difference between modes was far greater for novices than for experienced users. Using the wizard tended to equalize performance across skill levels. For system satisfaction, there were significant differences between interaction modes, however no differences were reported between skill levels.

Keywords: markup language, user interface, command language, wizard, performance, satisfaction

INTRODUCTION

Interaction mode has long been a key factor in user interfaces, and the choice of mode, whether command, direct manipulation, menu, or other kinds, can impact performance, learning, satisfaction, and other variables. In particular, the type of interaction mode can frequently have a significant impact on how quickly or effectively a task is completed, and also on how satisfied the user is with the language and task. There can also be significant differences when the effects of skill level are considered (17, 18). In this study, two main interaction modes are examined: command language (manual coding) and menu/form fill-in (wizard).

Command language interfaces, generally speaking, require a user to write and use specific commands, often requiring a specific syntax. Using operating systems such as DOS or UNIX, or writing programming code, are examples of command language interaction. These have tended to bring about better performance for experienced users, but can be a stumbling block for novice users due to the complexities of understanding and creating the detailed commands. As a result, novice users tend to be less satisfied with the command approach while many experienced users tended to prefer it (17, 18).

Menu/form-fill in interfaces, which are the main element of wizards, simplify the process of writing code, since users need only respond to inputs such as choosing the type of command desired and filling in the appropriate boxes, and the system will create the code (17, 18). Wizards, which are now featured on many of the software we use daily, including components of the Microsoft Office suite, for example, automate the process by leading the user through a series of input screens, after which the system will write the command or code for the user. In addition to examining the role of interaction modes, it is also important to examine what markup languages are, and why they should be the subject of further study. Markup languages have become increasingly important in the computing community, because of its wide use in both Internet web page design/development and electronic publishing. Simply put, a markup language consist of a set of tags, tokens, characters, or specialized command which are placed in a body of

text in order to provide information about the text or other data being processed (3, 7). Over the past few years, the importance of languages such as HTML (HyperText Markup Language), XML (eXtensible Markup Language), Standard Generalized Markup Language (SGML) , Dynamic HTML (DHTML) and others cannot be over-emphasized. HTML is widely used for creating hypertext-based documents which run on the World Wide Web (4, 12). SGML is an international standard for describing marked-up electronic text (1, 20). The definition of customized markup languages and tags for specific applications using XML is creating much attention and interest, for e-commerce and related applications (8, 10, 11, 19).

User background is another important factor which was examined in this experiment. The previous programming background of the subjects is an important factor which should be examined when it comes to evaluating interaction mode and performance/satisfaction. In general, experienced users tended to have a better "mental model" of a system or a language. There are a number of ways to obtain this kind of mental model--through usage, through analogy, and through training (16). A person can learn how a system is put together by using it; it is a function of the user interface, knowledge of other, prior systems, and various individual traits. Analogies are also useful to help someone develop this kind of mental model, which can be heavily influenced by their use of similar systems or by previous experiences.

Payne (15) found that one of the main differences between novices and experienced users was found in the kinds of mental models which they have, with many novices having incorrect or ill-constructed models which lead to difficulty in actually conducting the task or writing the code. Since experienced users have much better mental models of programming and the use of markups, it is expected that they would have better performance compared with novice users. In connection with this, knowledge and experience with the complexities of programming would tend to improve satisfaction for these same experienced subjects as compared with novices.

This paper describes the background, results, and summary/conclusions for a research experiment which examines performance and satisfaction as it relates to the usage of a survey questionnaire markup language. The language used, which has specific markup tag types designed for the creation of questionnaire question types, was designed to be an enhancement to markup language standards such as HTML and SGML (9).

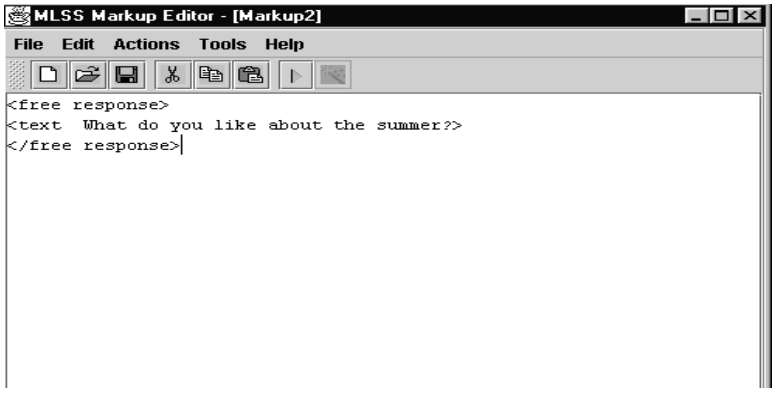

FIGURE 1. COMMAND LANGUAGE MODE

INTERACTION MODES

Command Language Mode: Subjects are asked to build the survey markup text on a text editor, which is part of the MLSS Integrated Development Environment (IDE). The subject was

provided with materials to assist them in completing the task, but it basically involved the creation of code character by character. This interaction method may provide for better performance and efficiency by experienced users, who through this mode must solve the problem with interpretive instructions and who are working with a familiar task (13).

Wizard (Menu-Form) Mode : A subject creates an online survey using a sequence of menu screens, which "walks" or "holds the person's hand" through the interaction process. This menu consists of both selection and fill-in-the-blank type items. The system correspondingly builds the markup language, line by line until the entire markup question structure is completed. The advantage of this method is that it allows users to build the markups without needing to be concerned with the syntax and details of coding each line of the markup language tags and elements. From a theoretical perspective, it could be said that the

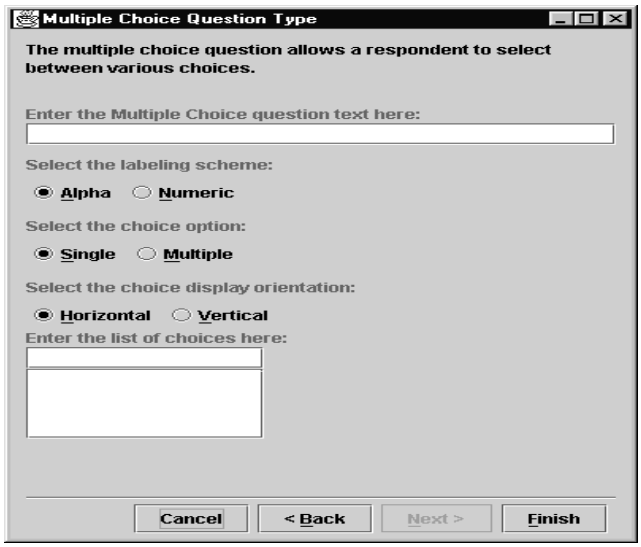

FIGURE 2. WIZARD MODE

menu-based method provides for specific instructions for a non-familiar task, which may make it ideal for novice users. On the other hand, it may result in slower, less efficient performance for experienced users (13).

MATERIALS

The following materials were used in this study:

Pre-Test Questionnaire. Filled out by all subjects who participated. A key section of the questionnaire addressed previous background, and was used to determine skill level.

MLSS Software. Each participant was provided with a customized copy of the experimental software, which was a custom-designed Java-based PC application which incorporated the following:

Editor for writing the markup code.

Interpreter which reads and creates output from the markup code.

Experimental Task . The task entailed taking a printed 15-item survey questionnaire, and coding it using the markup language. The resulting code, if written correctly, should produce an online version of the questions to be displayed on the screen. The questionnaire used is an adapted

(shortened and simplified) form of the EIES Virtual Classroom questionnaire , which has been used in previous research to tests users' reactions and responses to online Virtual Classroom courses and systems.

Post-Test Questionnaire. This questionnaire, administered after the completion of the experimental task, included items which measured the user's satisfaction with the language and the task.

SUBJECTS

A total of 268 undergraduate students participated in this experiment. The students were volunteers recruited among students majoring in computer science, engineering, architecture, and accounting. The students were given credit for participating in the study. Subjects were categorized as either a "novice" or an "experienced" subjects according to their previous background, as reflected in their responses to items in the Pre-Test Questionnaire, administered to the subjects prior to conducting the actual task. The subjects who participated in the study were primarily undergraduate students from the New Jersey Institute of Technology in Newark, NJ.

PROCEDURE

This experiment, which examined the performance, as well as system satisfaction, of users in completing a markup language task as it relates to interaction mode and user experience level, is in effect a 2x2 factorial design, with 2 forms of interaction and 2 types of experience level. The procedure for this experiment was conducted in three main stages: Pre-Test, Test, and Post-Test.

The Pre-Test stage involved the distribution of Pre-Test Questionnaires to prospective subjects, with the intention of analyzing them and categorizing them according to skill level. The determination of skill level was determined according to the following formula:

Experienced were defined as those subjects who rated themselves as having EITHER:

Medium or high or very high programming experience OR

Medium or high or very high markup knowledge/experience.

Novices were defined as those subjects who did not meet either of the above conditions.

The Task Stage required subjects to build an online questionnaire using the markup language,using the markup tags/code. In essence, the subject would build an "electronic" version of the paper questionnaire, which displays survey questions onto the screen after interpretation of the markup language code.

The Post-Test stage involved submittal of all task-related information, followed by the completion of a Post-Test questionnaire which included a quiz and items focused on subjects' perceptions of learning. The scores for the two groups (Command, Wizard) crossing with the two levels of experience (Novice, Experienced) were analyzed using GLM ANOVA, using SPSS v9.0 at a 0.05 level of significance.

RESULTS

Online time is a measure of how long a subject spent online, using the MLSS system, in working on the task. This was measured in minutes using the MLSS system's built in log file system, and was the total time spent online working on the task.

Table-1 : GLM Analysis of PERFORMANCE/ONLINE TIME: Means by Condition (Minutes)

Symbol	Condition	Novice	Experienced	All Conditions
CMD	COMMAND	216.37	137.78	177.55
WIZ	WIZARD	67.09	67.45	67.27
	All Subjects	**141.70**	**103.12**	**122.41**

GLM Results			Significance
Model	F =99.90	p = .000	***
SKILL	F = 18.97	p = .000	***
MODE	F = 154.96	p = .000	***
SKILL * MODE	F=19.33	P= .000	***

* = Significant at p < .1 ** = Significant at p< .05 ***= significant at p< .01

As clearly shown in the above table (see Table 1), there were significant differences between subjects of differing skill levels and interaction modes. Both of these were highly significant at p < .01. Command language interaction by novices clearly took the longest time (216.37), followed by command language by experienced users (137.78). However, there was little difference between the results, across skill levels, for users of the wizard (67.09 for novice and 67.45 for experienced).

System Satisfaction is a measure of how satisfied subjects were with the system they used, which in this case was the MLSS software. System satisfaction was measured using a composite System Satisfaction scale. The questionnaires and scales used in this research were generally consistent with those used in studies past NJIT research, which include both research projects and dissertation research (2,5,6,14).

Table 2: GLM Analysis of SYSTEM SATISFACTION: Means by Condition

Symbol	Condition	Novice	Experienced	All Conditions
CMD	COMMAND	24.24	18.85	21.55
WIZ	WIZARD	15.34	19.67	17.51
	All Subjects	**19.72**	**19.26**	**19.53**

GLM Results			Significance
Model	F = 238.20	p = .000	***
SKILL	F = .497	p = .481	
MODE	F = 29.01	p= .000	***
SKILL * MODE	F= 41.88	p= .000	***

* = Significant at p < .1 ** = Significant at p< .05 ***= significant at p< .01
Smaller values indicate higher system satisfaction.

The results, shown in Table 2, indicate that there were significant differences between subjects of differing interaction modes (p < .01) but that there were no significant differences between skill levels (p > .1). There was a significant difference when the interaction of skill level and interaction mode was considered (p < .01). In general, wizard users showed a higher level of system satisfaction, and while novice users preferred the wizard, experienced users preferred the command language. It appears, however, that there is a greater difference for the novice users as compared with the experienced users.

DISCUSSION AND CONCLUSION

The results presented here focus on two main dimensions: performance, as measured by online time spent on the task, and satisfaction, as measured by a System Satisfaction scale. To start, performance was generally better overall for experienced users (103.12) as opposed to novices (141.70), and the difference was significant, however there was a large and considerable difference between users of the command language (177.55) versus the wizard (67.27). Also of interest is the fact that the wizard tended to equalize performance across skill levels; in other words, both novice and experienced users spent roughly the same amount of time to complete the task. It is of course expected that experienced users would complete the task in less time than novices, and that the wizard would bring about faster performance. However it would have seemed logical that experienced users would still complete the task faster than novices using the wizard, but that did not turn out to be true.

From the perspective of system satisfaction, there was no significant difference between novice and experienced users ($F=.497$, $p=.481$) however there was a difference when it came to system satisfaction as it related to interaction mode ($F=29.01$, $p=.000$). In general, wizard users were far more satisfied overall than command language users (17.21 versus 21.55) .

In addition, it appears that there is a greater difference in satisfaction between interaction modes for novices than for experienced users. Clearly, users with experience in programming and markups would find the interaction mode to be less critical in terms of their satisfaction with the system—simply different ways of doing the same thing. However novices may find the wizard approach to be user-friendly and enjoyable, however the experience of using the command language and editor may have proved to be rather trying and difficult.

In general, it could be concluded that if the goal is to complete a task quickly, the wizard is definitely the mode to use. Both experienced users and novices were able to complete the task significantly faster than using the command mode. However, if command language coding is used, experienced users were able to complete it in twice the time needed compared with the wizard. In the case of novices, it took approximately 3 times as long.

Experienced users were roughly equally satisfied regardless of the interaction mode used, a reflection of their previous experience in programming and markup languages. Novices, on the other hand, were quite affected by interaction mode, expressing great satisfaction with the wizard and far lower levels of satisfaction with the command language mode.

System designers who are looking to design programming or web-development systems should apply these findings to systems they are looking to design or enhance. For tasks where prompt execution and fast completion is desired, the best choice would be to offer a wizard feature so that users of any skill/experience level could use it easily. The addition of the wizard feature could help to enhance acceptance of the system and encourage users to work with it frequently. On the other hand, it appears that while novices greatly appreciate wizard-based systems, experienced users do not show the tendency to the same degree, and it is possible that some experienced users would actually prefer using the command language interaction. Further research which examines the relationships between these interaction modes and learning, as well as the effects upon other measures of performance, could yield more insights into the impacts of interaction mode on using and learning markup languages.

REFERENCES

1. Association of American Publishers. (1992). Electronic Manuscript Project, Reference Manual on Electronic Manuscript Preparation and Markup.
2. Benbunan, R. (1997). *Effects Of Computer-Mediated Communication Systems On Learning, Performance And Satisfaction: A Comparison Of Groups And Individuals Solving Ethical Scenarios*, Ph.D. dissertation, Rutgers University.
3. Coombs, J.H., A.H. Renear, and S. J. Rose, (1987) Markup Systems and the Future of Scholarly Text Processing, *Communications of the ACM*, 30, 11, pp. 933-947.
4. Darnell, R., (1997) *HTML 4 Unleashed*, (Indianapolis: Sams)
5. Dufner, D., (1995) *The Effects of Group Support (Listing and Voting Tools) and Sequential procedures on Group Decision Making, Using Asynchronous Computer Conferences*, Doctoral Dissertation. Ph.D. in Management Program, Graduate School of Management, Rutgers University, Newark.
6. Fjermestad, J. (1994) *Group Strategic Decision Making in a Computer-Mediated Communications Environment: A Comparison of Dialectical Inquiry and Constructive Consensus Approaches*, Doctoral Dissertation. Ph.D. in Management Program, Graduate School of Management, Rutgers University, Newark.
7. Goldfarb, C., (1991) The SGML Handbook, (New York: Oxford University Press).
8. Holzner, S., (1997) *Xml Complete*, (New York: Mc Graw-Hill)
9. Hsu, Jeffrey and M. Turoff, (1996). A Markup Approach to Surveys and Questionnaires, *Journal of End User Computing*, Vol. 8 No. 4, pp. 20-27.
10. Leventhal, M. And D. Lewis, (1998), *Designing Xml Internet Applications (Charles F. Goldfarb Series)*, (Upper Saddle River: Prentice Hall)
11. Light, R., and T. Bray, (1997). *Presenting Xml*, (Indianapolis: Sams)
12. Mullen, R. , (1998) *The Html 4 Programmer's Reference*, (RTP: Ventana, 1998)
13. Norman, D., (1986). Cognitive Engineering, in *User-Centered System Design*, Norman, D.L. and S.W. Draper, (eds.) (Hillsdale NJ: Lawrence Erlbaum), pp. 31-61.
14. Ocker, R. (1995). *Requirements Definition using a Distributed Asynchronous Group Support System: Experimental Results on Quality, Creativity and Satisfaction*, Ph.D. dissertation, Rutgers University.
15. Payne, S.J. and T. Green, (1989). The structure of command languages, *International Journal Man-Machine Studies*, 30, pp. 213-234.
16. Sein, M.K. and R.P. Bostrom, (1989). Individual differences and conceptual models in training novice users, *Human Computer Interaction*, 4, pp. 197-229.
17. Shneiderman, B. (1997) *Designing the User Interface 3/E* (Reading MA: Addison Wesley).
18. Shneiderman, B., (1997) *Designing the User Interface 2/E* (Reading MA: Addison Wesley).
19. St. Laurent, S., (1997). *XML: A primer*, (IDG, 1997)
20. Wright, H. (1992). SGML Frees Information, *BYTE* pp. 263-268.