Consistency, Multiple Monitors, and Multiple Windows

Dugald Ralph Hutchings
Computer Science Department
Bowling Green State University
Bowling Green, OH USA 43403
drhutch@cs.bgsu.edu

John Stasko
GVU Center & College of Computing
Georgia Institute of Technology
Atlanta, GA USA 30332
stasko@cc.gatech.edu

ABSTRACT
We present an evaluation of mudibo, a prototype system for determining the position of dialog boxes in a multiple-monitor system. The analysis shows that, when compared to a standard approach, mudibo offered a 24% decrease in time needed to begin interaction in a dialog box. Analysis of participant behavior in the evaluation provides insight into the way users perceive and act in multiple-monitor environments. Specifically, the notion of *consistency* changes for multiple-monitor systems and the prospect of adaptive algorithms becomes further complicated and intricate, especially for window management.

Author Keywords
consistency, multiple monitors, window management

ACM Classification Keywords
H.5.2 Information interfaces and presentation: User interfaces – *GUI, Windowing systems*

INTRODUCTION
Recently HCI practitioners and researchers have become interested in design and evaluation for multiple-monitor users. A single set of input devices (typically one keyboard and one mouse) and two or more monitors driven by one or more video cards comprises a multiple-monitor computer system. Each monitor separates the physical display space yet the mouse pointer easily moves from one monitor to the other; the virtual display space is a connected entity. Different research has shown that users can expect to be more productive when using multiple monitors to complete tasks [3, 4] but that interfaces can act in unpredictable ways [4].

One of the noted problems with existing window managers for multiple-monitor systems is the inconsistent, confusing, or nonsensical handling of the display of dialog boxes [6]. Typically window managers leave the decision of placing dialog boxes to applications, which causes inconsistent behavior across applications. Applications tend to exhibit one of three major behaviors for placing a dialog box *db* with respect to a main application window *mw*: (1) put *db* in its last known location, possibly on a monitor other than where *mw* is placed; (2) put *db* on top of *mw*; (3) put *db* on a fixed monitor, regardless of the current monitor of *mw*. Notice that each of these strategies is identical for a single-monitor system (the last known location, the location of the main window, and the fixed monitor are all on the same (single) monitor). However, in a three-monitor system, each of these strategies could result in *db* appearing on a different monitor (the last known location could be on the left-hand monitor, *mw* could be on the right-hand monitor, and the fixed monitor could the monitor in the center).

Each of the three approaches is sensible, but when every application running on a system exhibits a different behavior, the result is an overall inconsistent, unpredictable method of placing dialog boxes. Some have noticed this issue and taken steps to provide the user with a consistent, predictable method of dialog box placement. For example, NVIDIA provides nView for all of its multi-display graphics adaptors (http://www.nvidia.com/object/feature_nview.html) and can be considered to be the state of the art in dialog box decision interfaces. It allows the user to select from among strategies #2 and #3 above and one additional strategy (always place the *db* under the mouse cursor). nView redirects the placement of any dialog box from any application. While nView addresses consistency, each of the strategies is susceptible to poor decisions based on user-context. Important notifications might be best placed in the user's focal area, which might not be on top of the active window, at the current position of the mouse cursor, or on some specific monitor *X*. Dialog boxes that facilitate data manipulation might be best placed to a side monitor to alleviate problems of occlusion while dialog boxes that do not interact with data might be best placed on top of the main window or under the mouse cursor, each of which might reside on different monitors. The question that arises is whether it is better to have good individual decisions but lack consistency or to exhibit consistency at a potential loss of good judgment.

It was with this question in mind that we set out to create an interface that was both consistent and led to good decision-making. *mudibo*, presented at CHI 2005, exhibits these properties by initially showing a copy of a dialog box on *every* monitor (thus on top of the main window *and* underneath the mouse pointer), waiting for the user to start interacting with one of the copies, and automatically hiding the remaining dialog box copies [7]. We argued that using mudibo would result in decreasing navigation time needed to access dialog boxes because every option was available in the smallest possible amount of time (and further could be aided by other time-decreasing techniques such as alter-

ing mousing behavior [1]). We became interested in the *degree* of savings that users might expect to enjoy and set about to determine the differences in a controlled study. Our pilot study examined the difference between mudibo and the standard behavior of applications on Windows XP. Given positive results in the pilot, our plan was to subsequently compare mudibo to nView.

We hypothesized that mudibo would provide a statistically significant savings in time to initially place a dialog box. Analysis of participant *data* upheld our hypotheses but surprisingly, analysis of participant *behavior* provided some very interesting insights into the way that people interact with multiple monitors. In this paper, we briefly describe the formal experiment and data collected to support the main hypothesis but spend the remaining time discussing participant behavior and the potentially far-reaching implications of the behavior. We discuss in particular the difficulty of defining "consistency" for multiple-monitor interfaces and the significant challenges to building appropriate, adaptive interfaces for multiple-monitor users.

USER STUDY

Since we are not focusing on the numerical results of the collected data in this paper, we provide a basic overview of the study to allow space for richer discussion of user behavior. For a full description of the study, refer to the dissertation of the primary author [8].

Method

We arranged a three-monitor system from left to right and built a simple text editor program with which participants interacted in the experiment. In a pre-set order, participants completed a set of 12 tasks, alternating among six *find* tasks and six *font* tasks. In the *find* task, participants summoned a dialog box that allowed them to search through a document for a specified word, counting the number of occurrences of the word. Whenever the dialog box appeared on top of the main editor window, it always occluded at least one piece of found text (participants also experienced this situation in practice trials). Changes occurred in the main window while the *find* dialog box was visible, which contrasts to the *font* dialog box. *Font* tasks required participants to select a specified paragraph of text, summon a dialog box, make changes to font name, size, or style, click *OK*, and verify the changes. Changes occurred in the main window only after the *font* dialog box closed. Each participant completed four sample tasks (two font tasks and two find tasks) on a single-monitor configuration to gain an understanding of the editor interface. The examiner then set the three-monitor configuration and demonstrated to the participant that windows could be dragged and positioned among the different monitors.

The study used a within-subjects design. Participants completed these sets of 12 tasks in two conditions, *M* and *N*, for a total of 24 tasks. In condition *M*, mudibo was applied to dialog box placement and in condition *N*, mudibo was not applied. In *N*, for each task type, three times the dialog box appeared on top of the parent window and three times the dialog box appeared on a side monitor relative to the parent window. Within side placement, twice the dialog box appeared on an adjacent monitor and once the dialog box appeared two monitors away from the parent window. The main window itself appeared an equal number of times on the left, center, and right monitors. Upon participant arrival, the facilitator assigned a sequential number to the participant (1, 2, 3, …). Odd-numbered participants received the set ordering *NM* and even-numbered participants received *MN*. Prior to conducting each set of tasks, the facilitator informed the participant how dialog boxes would be placed on screen, *i.e.*, both mudibo and non-mudibo were explained in detail. At the completion of both sets, participants engaged in an interview with the study facilitator.

A logger accompanied the interfaces to track each time that a user interacted with the interfaces in any way, including button clicks, window movement, window appearance and disappearance, initial selections (when mudibo was turned on), *etc*. Each log entry included timestamp information to allow analysts to recreate participants' actions. We hypothesized that log data would indicate that participants began interaction with dialog boxes in condition *M* significantly faster than in condition *N*.

Results – Data Analysis

For each of the 12 enrolled participants and each of the 24 tasks we used log data to calculate the amount of time that passed between showing the dialog box and the first interaction by the participant (clicking on a button, in a textbox, or on a list item or typing on the keyboard). Then for each participant, we summed the total times across tasks in each condition *M* and *N* and calculated the sample mean and standard deviation across participants. Finally, we conducted Student's paired-samples, one-tailed *t* test on the means and standard deviations and report the results in Table 1. On average, participants began interaction about 24% more quickly in the mudibo condition *M* as compared to the non-mudibo condition *N* for an average absolute savings of approximately 0.75 seconds per dialog box.

These results are promising and we plan a future evaluation comparing mudibo to the three nView approaches discussed in the introduction. However, our observations during the experiment (which are reflected exactly by the logged data) of user behavior offer some key insights into general user behavior on multiple-monitor systems.

Results – Behavior Analysis

Each participant adopted a *basic* strategy for handling dialog boxes in the experiment and we are able to place each participant in one of three strategy classes. Note that classes 1 and 2 correspond to two of the offered strategies

t	Cond. *M*	Cond. *N*
$\bar{x}$	27.36 secs	36.34 secs
σ	5.88	9.48
p	0.006	

Table 1: Participants were significantly faster to begin interaction in condition *M* than in condition *N*. The average savings is ((36.34–27.36)/12) = 0.75 sec per dialog box, a 24% decrease.

by nView while class 3 does not have a correspondence.

Class 1 – Move as little as possible – 6 participants

In condition *N*, participants simply used the dialog box wherever it appeared. In condition *M*, participants used the closest dialog box (since all dialog boxes were summoned from a menu bar on the same monitor as the main window, the closest dialog box was always on top of the main window).

Class 2 – Always on top – 3 participants

In condition *M*, participants exhibited the same behavior as class 1. In condition *N*, participants moved the dialog box on top of the main window before interacting with it (if it did not initially appear on top of the main window).

Class 3 – Act based on task type – 3 participants

In condition *M*, participants chose the font dialog box to be on top of the main window and the find dialog box to be on a side monitor. In condition *N*, participants used both types of dialog boxes wherever they appeared and moved the find dialog box to a side monitor only when it obscured found text in the main window.

We stressed at the beginning of this section that each participant adopted a *basic* strategy. Yet only two of twelve participants (both from class 1) followed their basic strategies throughout the experiment and we now move to discuss these variations in basic approach. We use two key terms in describing the varied behavior: *alteration*, which describes a consistent variation on the basic strategy for a certain task type or condition; and *exception*, which is behavior that further strays from either the basic strategy or the altered strategy. The noted exceptions also occur "in the middle" of exhibited behavior, which is to say that a participant followed a basic or altered strategy for one or more tasks, then behaved differently for one task, then returned to the strategy for one or more tasks.

In class 1 (*move as little as possible*) two participants exhibited an alteration where they moved the find dialog box to a side monitor but only after it obscured found text. Another participant's alteration involved moving the find dialog box to the side upon obstruction only for condition *M* and leaving it on top for condition *N*. Yet another participant showed an alteration where a side monitor was always picked in set *M* regardless of the task. All four participants exhibited one to two further exceptions from the alterations.

In class 2 (*always on top*), two of three participants followed the strategy with no alterations but exhibited one exception each. One participant's alteration was leaving the *font* dialog box to the side if it appeared there in condition *N* (surprisingly not for *find*) and exhibited one exception.

In class 3 (*based on task*), one participant followed the strategy for all tasks except three. Another followed the strategy except for the find dialog box in condition *N*, which followed no discernable pattern whatsoever (with one additional exception overall). The final participant followed this strategy but altered it such that he always put the font dialog box on top in set *N* and had one more exception.

This experiment contains the minimum number of types of tasks (2 types) to cover the space of ideal dialog box placements and repeats the specific tasks (2 tasks) several times. It is not surprising that different participants adopted different basic strategies, but it is quite interesting that each of 10 of 12 individual participants altered their own strategy based on task context, or exhibited additional exceptions from their own strategy. These behavioral patterns (or non-patterns) support an interface such as mudibo, which presents the user with an option matching any strategy at decision-time. We revisit this discussion in the next section as "simultaneous, multiple alternatives."

Results – Interview

Eleven of twelve participants indicated that they were annoyed in condition *N* when mudibo was not running and a dialog box would appear more than one monitor away from the main window, indicating that mudibo is a clear win for cases such as this. This situation occurs in everyday interaction when an application recalls the most recent position of a given dialog box and places the re-summoned dialog box there. It is clear that this approach should be abandoned for multiple-monitor systems. Interestingly in a recent upgrade to the Microsoft Windows version of the Firefox Web browser we observed that it has switched from a "most recent position" strategy to an "always on top" strategy (http://www.mozilla.com/firefox) while the Apple OS X version continues to exhibit the "most recent" strategy.

Seven participants thought mudibo was a faster approach and two thought that non-mudibo was faster, with three participants indicating that they thought there was no difference. Yet only five participants thought that mudibo was easier than non-mudibo, with three indicating that both methods were the same. Looking at overall preference, seven people preferred mudibo while five preferred the non-mudibo approach. Three of the five participants who did not prefer mudibo indicated that if mudibo allowed a user to start typing in the dialog box immediately, then they would prefer mudibo.[1]

IMPLICATIONS

The clear notion and value of consistency in the interface becomes muddied in the context of interaction on multiple monitors. As we mentioned, nView provides two options for consistent placement of dialog boxes that correspond to two classes of user behavior. Unfortunately, most of the users in each class did not themselves consistently exhibit the behavior. The implication is that while nView provides consistent and predictable dialog box placement, the placement itself might not match a user's desire for a specific task. Other times, it will be wrong simply because the user has selected a desire that varies from normal. Regardless of the desires of a user at any particular time, mudibo provides a dialog box on the chosen monitor (though it also makes the wrong choices for the rest because it is simultaneously available on all monitors). This style of consistency pre-

[1] The fact that mudibo does not allow immediate typing is an artifact of current technical limitations of the prototype; systems like Metisse that can implement mudibo overcome this limitation [2].

dictably leads to the right *user* decision every time and essentially avoids a pre-set system decision.

Indeed, there was a very interesting exchange between the study facilitator and one of the participants regarding the notion of consistency. The participant was explaining that he would prefer the non-mudibo approach but would want it to be more consistent. When asked what he meant by "consistency" he said, "[I would like the dialog box to appear] where the main window is, you know, where the mouse is" as he pointed to a window on-screen. As it so happened, the active window was on a different monitor than the one to which he was pointing and the mouse pointer was on yet another monitor! When the facilitator pointed this situation out, the participant immediately reacted to the situation and understood the purpose of the experiment, though still thought the non-mudibo approach would be better for him.

Consistency also relates to elements of the interface with which the user does not directly interact. Dialog boxes can appear without user action. Where do new instant messages, initiated by a second-party, appear on-screen? Is the decision based on the current task of the user, the sender of the message, or a combination of those and other factors? What about notifications from the operating system, window manager, or applications different than the one the user is working with or looking at (which themselves can be different interfaces or applications)? Do these items call for consistent, but different, display methods? Is mudibo a worthwhile approach here? Designers must realize that the likelihood that the user's focus is some place other than the active window dramatically increases in multiple-monitor environments [8].

Multiple monitors provide users with options to interact that were not available on single-monitor systems. It might not be enough to provide the user with a set of consistent approaches from which he or she must choose a single approach. Designers may need to work hard to provide approaches that are both consistent and provide multiple alternatives simultaneously. Giving users multiple, simultaneous alternatives has already been shown to benefit them even in single-monitor environments [9]. One major direction of future work is to continue to explore areas of general interaction that can benefit from interfaces that are not only consistent but provide multiple alternatives simultaneously, each accessible in a minimal amount of time.

Some researchers have suggested adaptive approaches to the display of information and response to user actions in the interface, especially for window management. Evaluations of window-oriented systems appear infrequently and tend to highlight the difficulties of accounting for the entire context of the user. In the earliest known assessment of an adaptive window manager for a simple, restricted multiple-monitor system, evaluators noted the difficulty with understanding user preferences and needs and concluded that they needed to build more contextual information and history information into decision-making processes [5].

We question whether these systems will ever have enough contextual information to make an appropriate decision. Further, if users fail to fall into predictable patterns, making out-of-ordinary decisions for no apparent reason for even the simplest of tasks such as changing fonts or finding words in a document, what is the value of an adaptive algorithm for richer interaction? Is the potentially huge cost of acquiring all of a user's contextual information worth the benefit provided by the adaptive approach?

We see the battle of proponents of consistent interfaces that provide multiple, simultaneous alternatives against supporters of adaptive interfaces that take user context into account as a friendly rivalry that can positively push the state-of-the-art in both fields. When multiple alternatives are equally accessible, we might have a better understanding of the frequency with which users choose one option instead of another, leading to better models of user context. When adaptive algorithms fall short of user expectations on multiple-monitor systems, we might again gain insight into key design principles for multiple-monitor interface design that can be addressed by mudibo-style approaches. What remains clear is that the notion of consistency in the interface, especially for window management and window interaction, must be carefully considered as these multiple-monitor systems enjoy heightened popularity and spread to increasingly broader user groups.

ACKNOWLEDGEMENT

We thank the National Science Foundation for their support under grant number IIS-0414667.

REFERENCES

1. Benko, H. and Feiner, S. Multi-monitor mouse. *CHI 2005 Extended Abstracts*, ACM Press, 1208–1211.
2. Chapuis, O. and Roussel, N. Metisse is not a 3D desktop! *Proc. UIST 2005*, ACM Press, 13–22.
3. Colvin, J., Tobler, N., and Anderson, J. A. Productivity and multi-screen displays. *Rocky Mountain Comm. Review 2:1 2004*, Dept. Comm. Univ. Utah, 31–53.
4. Czerwinski, M., Smith, G., Regan, T., Meyers, B., Robertson, G. and Starkweather, G. Toward characterizing the productivity benefits of very large displays. *Proc. INTERACT 2003*, IOS Press, 9–16.
5. Funke, D. J., Neal, J. G., and Paul, R. D. An approach to intelligent automated window management. *Int. J. of Man-Machine Studies 38 1993*, 949–983.
6. Grudin, J. Partitioning digital worlds: focal and peripheral awareness in multiple-monitor use. *Proc. CHI 2001*, ACM Press, 458–465.
7. Hutchings, D. R. and Stasko, J. mudibo: multiple dialog boxes for multiple monitors. CHI 2005 Extended Abstracts, ACM Press, 1471–1474.
8. Hutchings, D. R. Making multiple monitors more manageable. Doctoral Dissertation, Georgia Institute of Technology, Atlanta, GA, USA, August 2006.
9. Terry, M., Mynatt, E. D., Nakakoji, K., and Yamamoto, Y. Variation in element and action: supporting simultaneous development of alternative solutions. *Proc. CHI 2004*, ACM Press, 711–718.