Planning for the User-Interface: Window Characteristics

Boris Kerkez and **Michael T. Cox**
Department of Computer Science and Engineering
College of Engineering & CS
Wright State University
Dayton, OH 45435-0001
{bkerkez;mcox}@cs.wright.edu

Abstract

While manipulating an interface, users often generate a proliferation of windows that overlap with each other and cause disorganization. Given a cluttered screen, many actions are possible to reorganize the configuration including moving and minimizing windows. This paper claims that three types of knowledge exist to aid the user and the system in intelligently reducing the clutter. They are the knowledge of window characteristics, window content, and user problem-solving context. We present a system called P4P. It implements a planner that currently uses the first and second knowledge types to achieve a state of reduced screen-clutter given arbitrary window configurations. Two examples illustrate the process. We conclude that, while necessary, knowledge of geometrical window characteristics alone is not sufficient to effectively manage the user interface.

Introduction

One of the basic decisions within the problem of assisting a user to dynamically manage an interface layout is the choice of window configuration. While using the interface, windows often proliferate and overlap with each other thereby creating disorganization or *screen clutter*. Given a cluttered screen, many actions are available to reorganize the configuration including moving a window, forcing an overlapping window to the background, and minimizing (iconifying) a window. At least three types of knowledge exist to aid both the user and the system in intelligently reducing the clutter by applying such actions.

- Knowledge of a window's characteristics
- Knowledge of a window's content
- Knowledge of a user's problem-solving context

To enhance user performance, an intelligent user-interface (IUI) can exploit such knowledge in order to organize the presentation of information. Conversely the user can employ this knowledge to guide the actions that a IUI chooses.

The knowledge dealing with the window's characteristic is the lowest level of knowledge available for IUI in our taxonomy. This aspect of IUI knowledge deals with the physical positioning of the windows in the user interface (spatial characteristics), including position of windows with respect to both each other and the screen display area. The knowledge that deals with the spatial characteristics of windows can be obtained solely from the geometrical window characteristics of the user interface. Characteristics of the windows also include the recency and frequency of use. In order to obtain the knowledge about the frequency of use, IUI must be able to monitor user's activity in terms of the mouse clicks and keyboard inputs. For instance a window that has not been "touched" by the user for a significant amount of time (i.e., recently) may be considered irrelevant to the user's goals (cf. Miah & Alty, 2000). Such knowledge can be useful for re-organizing the user interface. For example "Iconify the window least recently touched" is a tempting heuristic.

Unfortunately the knowledge about the windows' characteristics alone is not enough for the problem of managing the user-interface. Knowledge about window content disambiguates many inferences. For example a window that displays incoming email messages to a user may be very relevant, although, it may not have been used recently or frequently. From strict window characteristics such as inactivity, a system may infer that the window is not being used. From this lack of use, a system may additionally infer that the window is not relevant to the user's goals. Subsequently it may choose to iconify this relevant window. Therefore any solution to the problem of managing a user

interface must take into consideration not only the recent usage of windows, but also, the content and the purpose of the windows. [1]

In order for a IUI to be able to produce not only consistent, but also complete plans to manage the user interface, the knowledge about the user's problem solving context must be addressed as well. IUI must know about its user's tasks, goals, and preferences in order to assist him or her better. At each step in the planning process, IUI must tailor the information presentation with respect to user's goals and intentions. Therefore, IUI must inevitably know and be able to recognize what kind of a task the user is trying to accomplish with the user interface. This very important knowledge aspect will be a focus of our future research efforts.

We present a system called P4P (Planning for Planners) that implements a planner that currently uses the first and second knowledge types to achieve a state of reduced screen-clutter, given arbitrary window configurations. The system consists of two parts: the first part is an underlying planner called PRODIGY (Veloso, *et al.*, 1995). It produces the plans to reduce the screen clutter. The second part is a Java application that executes the plans and allows users to see how the executed plan reduces the screen clutter. We also present two examples that illustrate the difficulties and challenges when dealing with the screen clutter problem.

This paper examines issues that concern geometrical interface features (a subset of all window characteristics) and the impact they have on a system's plan for a user interface. Although the screen clutter problem should actively involve both the system and the user, the focus of this paper is on the system. The subsequent section discusses the PRODIGY component of P4P, the domain representation of the micro-window domain, and briefly discusses an example in this domain. The third section presents a more detailed example of planning for clutter free screens with P4P. The possible window states that P4P must consider in the domain are expanded so that we can illustrate the complexity of the problem. The next to the last section projects our future research efforts. The conclusion section briefly summarizes the paper and then asserts that, while necessary, the geometrical characteristics of the user interface alone are not sufficient for effective plans for the user interface.

[1] For additional details about the use of window content knowledge, see Kerkez, Cox, & Srinivas (2000). In this paper an automated planner plans for its own user-interface, hence the name of our IUI system; that is, the acronym of P4P is "Planning for Planners" and reflects the introspective nature of this particular planning problem. For a brief description of the larger interface system in which P4P plays a role, see Cox (2000).

The Micro-Window Domain

The PRODIGY system employs a state-space nonlinear planner and follows a means-ends analysis backward-chaining search procedure that reasons about both multiple goals and multiple alternative operators from its domain theory appropriate for achieving such goals. A domain theory is composed of a hierarchy of object classes and a suite of operators and inference rules that change the state of the objects. A planning problem is represented by an initial state (objects and propositions about the objects) and a set of goal expressions to achieve. Planning decisions consist of choosing a goal from a set of pending goals, choosing an operator (or inference rule) to achieve a particular goal, choosing a variable binding for a given operator, and deciding whether to commit to a possible plan ordering and to get a new planning state or to continue subgoaling for unachieved goals. Different choices give rise to alternative ways of exploring the search space.

The Micro-Window, or μWindow, domain of P4P explores the problem of alleviating window clutter using a simple world composed of a screen and an arbitrary number of windows. The screen is divided into four area quadrants to render windows and a horizontal rectangle or bar across the bottom to hold icons. When not iconified, a window is contained in exactly one quadrant and fills the entire area. When iconified, the window is moved to the icon bar. Operations on windows are a subset of normal window operations: move a window; minimize a window; and restore a window. A window cannot be resized in this domain. Partial ovcerlap does not exist. Only total window overlap exists such that a lower window is fully ocluded by any window above it.

Several relational states exist for windows and quadrants. First, a window can be an icon (minimized), or not an icon. By definition an *iconified* window does not contribute to window clutter. Second, a window can be *on-top-of* another window (i.e., one window can entirely cover another window in the same area). Therefore a window can be on top (visible on the screen), or not be on top (there is some other window on top of it). Whether a window is on top or not is inferred. Third, an area can be *clear* or not clear. A clear quadrant indicates that there are no windows currently in the area.

Brown and Cox (1999) present an early example from the μWindow domain. The problem definition includes an initial state consisting of four windows, W1-W4, stacked in quadrant A1 with W1 on the bottom and W4 on top. The goal is to achieve *screen clarity* by creating a clutterless state in all quadrants. A clutterless state is achieved in a given quadrant by not allowinging windows to overlap (i.e., be one on-top-of another). The solution is a simple plan of first moving W4 off of W3 and into A2, W3 to A3, and W2 to A4. However, the simplicity of the problem and solution belies the complexity of the process that is necessary to produce this example. The domain requires multiple inference rules, operator constraints,

and search control rules to instantiate the subtle heuristics that managed the decision choices.

Detailed Planning Episode

In order to further represent interface management as a planning task, this paper examines a more realistic problem situation under a more complex domain than the one presented by Brown and Cox. Moreover, the new example will require more than just knowledge of window characteristics. Even though the plans produced in this example effect only the geometrical and spatial characteristics of an interface, the goals of the problem require information from both window context and user context.

To the μWindow domain, we add a fourth relational state to the three states defined earlier. A window can be either *active* or not. An active window is in the set of all windows used by the user during the current task. All windows in this set will be considered active. All windows not active are considered to belong to a suspended user task. Furthermore by definition, an iconified window is never a part of the active set. Since now we have a concept of the activity of windows, we redefine the concept of a clear area. An area, in this example that introduces the window activity, is clear if it does not contain any active windows. Notice that an area may be clear if it is occupied by one or more windows, as long as these windows are not a part of the set of active windows.

The goal of achieving the screen clarity can be accomplished if the planner is able to maintain a minimal percentage of window overlap. One obvious solution to this problem is to specify a threshold for the number of different windows that can be present on the screen at any given time. When this threshold is exceeded, all last recently used windows can be iconified. However, this solution does not easily scale to a more complex problem that involves a user's preferences and the content of the windows them. Windows that have not been used for some time may indeed be relevant for the task in hand.

As discussed in the introduction, a window that displays incoming email messages to a user may be very relevant, although it may not be subject to user's mouse clicks and the keyboard input for quite some time. However, this inactivity will render the window as not being used, and therefore, the window will be a candidate for iconification when the given threshold is exceeded. Therefore, the solution that will achieve the window clarity must take into consideration not only the recent usage of windows, and the physical positions of windows with respect to the screen and other windows; it must also include the knowledge about the content of the windows. In the previous example, knowing the context of a window (an email application) can provide additional knowledge that will enable IUI to infer the relevancy of a window.

This second example considers a problem of repositioning the windows, so that the windows that are relevant for the task (i.e., active) in hand are brought to the top, while irrelevant windows are moved away (iconified). The situation presented here has six windows in its initial state: two windows are email application windows; another is a word processor window; and the other three windows are not relevant to this example.

The initial state of the μWindow world is graphically drawn in Figure 1. In our example the user wishes to send an email message to someone; the context of the message will contain portions of the text that need to be copied in the message from a word processing application. Therefore, a word processing window (W1), and two email application windows (W2 and W5) will need to be active, because they are the relevant windows for this task. Notice that here we are using the knowledge about a user's problem-solving context, which is explicitly given to our planner. Our future research efforts will address the issues involved in gathering the knowledge about the context of a task from implicit application models (Gorniak & Poole, 2000). In the initial state, only W2 is visible on the screen in area A1, while other three visible windows are not relevant. Furthermore, the word processing window W1 is below the email window W2, and W5 is covered by another window in area A4, so W1 and W5 will have to be made visible for them to be indeed relevant, given the task in hand.

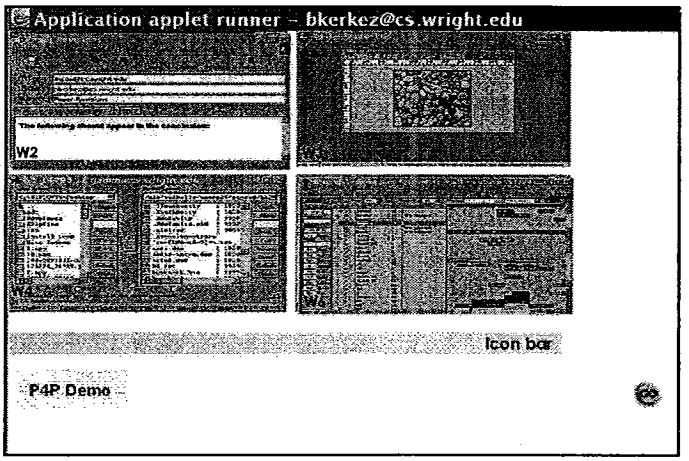

Figure 1. The Initial μ-Window World State has Six Windows (Two in the Upper Left and Two in the Lower Right)

Figure 2 shows the problem as defined in PRODIGY including the given initial and goal states. After inferring that

the windows W3 and W4 are not relevant, the goal will be to iconify them so that they are out of the way and do not contribute to the window clutter. The goal also specifies that all windows relevant for the current task (W1, W2 and W5) need to be active.

```
OBJECTS:

 Type = WINDOW:
 (W1 W2 W3 W4 W5 W6)

 Type = AREA:
 (A1 A2 A3 A4)

 States:
 ((in W1 A1)      (in W2 A1)
  (in W3 A2)      (in W4 A3)
  (in W5 A4)      (in W6 A4)
  (active W4)     (active W3)
  (active W6)     (clear A1)
  (on-top W2)     (on-top W4)
  (on-top W6)     (on-top W3)
  (on-top-of W2 W1)
  (on-top-of W6 W5)
  )

 Goals:
 (no-clutter,
  icon W3,
  icon W4,
  active W2,
  active W5,
  active W1
  )
```

Figure 2. PRODIGY initial state

In P4P the underlying PRODIGY planner maps the window and area objects and states of the plan into the visual representation of the problem. Figure 3 represents the initial status configuration for specifying the number of windows, by enumerating the windows in each area, and by specifying the name of the file into which PRODIGY will save the plan solution it generates.[2] The numbers "1 2" in the box labeled A1 mean that W2 is on-top-of W1 in the A1 area.

Figure 3. Setting the Initial State in P4P

[2] A P4P user can also use the status bar to manually setup and test various problems.

Figure 4 shows the plan generated by PRODIGY. To reduce the initial clutter and free up the quadrant areas that will be needed for active windows, the planner chooses to first iconify windows W3 and W4. Figure 5 shows the animation of the minimize action (PRODIGY operator application) for W4. One of the email application windows (window W2) is already on top in quadrant area A1, therefore, the window is made active and the area A1 is no longer clear.

Solution
- <infer-no-clutter>
- <minimize w3>
- <minimize w4>
- <make-active w2>
- <minimize w6>
- <make-active w5>
- <move w1 a3>
- <make-active w1>

Figure 4. PRODIGY Solution State

In this plan window W1 is moved from quadrant A1 to quadrant A3. Note also that both windows W3 and W4 have been minimized by the first two actions[3] in the plan sequence and are therefore represented as labeled icons on the icon bar.

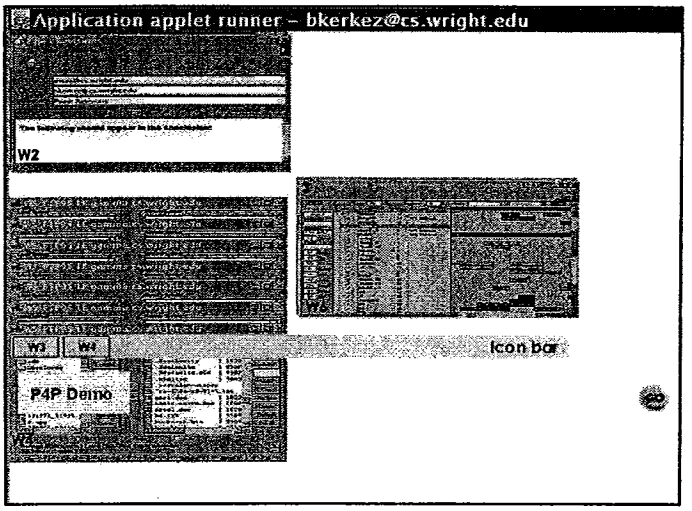

Figure 5. Animation of plan step <minimize w4>.

Now, the other email application window, W5, is covered by W6 in area A4. Although W6 is not relevant for the current task, it is currently active, and, the planner chooses to iconify

[3] The <infer-no-clutter> step is not actually an application of a plan operator or action. Instead it is an inferred state change.

it, so that the area A4 where window W5 is positioned is now clear. Therefore, now that the area A4 is clear, W5 is made active. Now, the word processing window W1 is below already active (and currently relevant) window W2, so it is moved to the area A3 and then made active. Figure 6 shows the final window configuration from which the goal states are achieved.

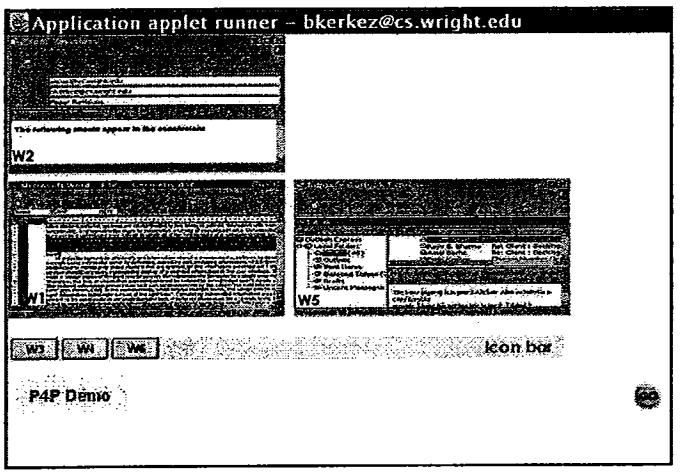

Figure 6. Final solution state

Future Research

Due to the preliminary state of the implementation of this research, many avenues exist toward which we may proceed. Certainly the μWindow domain must be made even more realistic. For example partially overlapping windows dominate most window clutter problems of actual user-interfaces. To discretize a screen layout into 4 areas, and further, to constrain window size to be equal to any area excludes domain operators such as window *resize*. However because our main research goal is not to simulate a real-world window reconfiguration problem in all its detail, but rather, to better understand the problems of intelligent management of window configurations, we designed a rather simple domain abstraction. Only future changes that promise to increase our understanding of how knowledge of window characteristics, window content, and user context (along with the interaction between each) impacts the designand execution of IUI's are of real interest.

A major improvement to the research is adding adaptability or learning to the planning process in a IUI. For example in the previous scenario, we can deduce that the email message window should not be iconified if it is visible on the screen at any time (i.e. it is not overlapped by any other windows). Without knowing about the content of windows, the decision about iconification of the email window may not be known *a priori*. One way to cope with this problem would be to provide the planner with the ability to learn from its mistakes. For instance, if the planner iconifies a relevant visible window, user should be able to correct the mistake made, and planner must remember user's decision in its future plans. To facilitate the learning component of the planner, the planner must be able to monitor interaction between users and the system. Fortunately all user's inputs are given to the system through the mouse and the keyboard clicks (excluding recently very popular speech interfaces). As Liebermann (1999) points out, by using "the marionette strings", which are strings that represent the mouse and the keyboard interactions of a human user with a software application, the planner is able to capture the interaction, and consequently, decipher the interaction to be able to learn by observing users.

Conclusion

Most modern software systems are characterized by the ability to provide their users with a very high level of functionality, and to enable the users to accomplish a variety of specialized tasks. Unfortunately it is very often the case that users find it difficult to utilize all of the functionality a software system offers. The cause of this condition stems from the complexity of system's user interface. It has been shown that, although almost every new version of a software system introduces added functionality to the user interface, most users choose to utilize those parts if the user interface that they are accustomed to, and spend very little time learning new user interface features (Lewis, 1998).

In order for humans to use information intensive environments effectively, software systems (and in particular IUI's) should provide enough assistance to the user by collecting, extracting and analyzing information, which then can be abstracted to include only those aspects that are relevant for the information presentation. Furthermore, a system should help users manage the display of presented information at the user interface. There does not exist a unique notion of the best way to present the information to the users in a single user environment. Therefore, the task of information presentation must be tailored to an individual user's needs and preferences. The tailored information presentation task can be summarized by the adage "the right information at the right time" (Brown & Cox, 1999).

Completely automated information presentation facilities prove to be inadequate for accomplishing the task of tailoring information for a specific user, because, they failed to successfully learn and capture the needs of individual users (Mitchel & Sundstrom, 1997; Woods, 1987). We believe that

the solution to this problem is a mixed-initiative approach of planning for a user interface, in which information gathering and presentation tasks are distributed between a system and a human user. For example any undesirable change by an IUI to the interface should immediately be over-ruled by a user, and, the IUI should adjust its future plans accordingly (but see Miah & Alty, 2000, for an interesting alternative). Most important however, is that, by providing a system with a human-in-the-loop it will enable it to account for specific needs of individual users. This is in addition to learning and revising its presentation planning approaches in the face of user feedback. The mixed-initiative approach to planning for a user interface will be another focus of our future research.

This paper describes some issues surrounding a set of window characteristics such as the geometrical aspects of the problem of planning for a user interface, and discusses the implementation of one solution to this problem in a toy window domain. However it is clear that knowledge of window characteristics alone is not sufficient to solve many of the complicated situations that arise in common application tasks. For a more complete solution, an intelligent user interface will require knowledge of window content and the problem-solving context of users. This research represents a small step toward a theory of intelligent interfaces that actively assist the user to visualize information by tailoring it to the individual's task.

Acknowledgements

The authors' research was supported by the Dayton Area Graduate Studies Institute (DAGSI) under grant number HE-WSU-99-09 and by Wright State University.

References

Brown, S., & Cox, M. (1999). Planning for Information Visualization in Mixed-Initiative Systems. In M. T. Cox (Ed.), *Proceedings of the 1999 AAAI-99 Workshop on Mixed-Initiative Intelligence* (pp. 2-10). MenloPark, CA: AAAI Press.

Cox, M. T. (2000). Interfaces for Mixed-Initiative Planning. In C. Rich & C. Sidner (Eds.), *Proceeding of the IUI'2000 Workshop on Using Plans in Intelligent User Interfaces* (pp. 32-34). Cambridge, MA: MERL.

Gorniak, P, & Poole, D. (2000) *Modeling Use of Unmodified Applications from Observed Interactions.* In C. Rich & C. Sidner (Eds.), *Proceeding of the IUI'2000 Workshop on Using Plans in Intelligent User Interfaces* (pp. 1-3). Cambridge, MA: MERL.

Kerkez, B., Cox, M. T., & Srinivas, C. (2000). *Planning for the User-Interface: Window Content.* Unpublished.

Lewis, M. (1998). Designing for Human-Agent Interaction. *AI Magazine, 19*(2), 67-78.

Lieberman, H. (1999) *Attaching Interface Agents to Applications.* MIT Media Lab. Unpublished. URL http://lieber.www.media.mit.edu/people/lieber/Lieberary/Attaching/Attaching/Attaching.html

Miah, T., & Alty, J. L. (2000). Vanishing Windows – a technique for adaptive window management. *Interacting with Computers, 12*, 337-355.

Mitchel, C. M., Sundstrom, G. A. (1997). Human Interaction with Complex Systems: Design Issues and Research Approaches. *IEEE Transactions On Systems, Man and Cybernetics*, 27 (3), 265-273.

Veloso, M., Carbonell, J. G., Perez, A., Borrajo, D., Fink, E., & Blythe, J. (1995). Integrating planning and learning: The PRODIGY architecture. *Journal of Theoretical and Experimental Artificial Intelligence*, 7(1), 81-120.

Woods, D. D. (1987). Cognitive Technologies: The design of joint human-machine cognitive technologies. *AI Magazine*, 86-91.