An Adaptive Window Management System

Stefan Stille, Shailey Minocha and Rolf Ernst

Institut für Datenverarbeitungsanlagen
Technische Universität, Braunschweig
Hans–Sommer–Str. 66, D-38106 Braunschweig, Germany
stille@ida.ing.tu-bs.de

ABSTRACT Modern complex and information intensive computer applications involve multi-window operations. To simultaneously visualize all the necessary information for a task requires time-consuming window management operations by the user, like opening/closing or moving/resizing the windows. Since these window interface housekeeping activities are not directly related to the user's task domain (Shneiderman, 1992), time spent on window management results in loss of user's mental context and an increase in the actual task completion time and hence, reduced user throughput.

We propose the concept of Adaptive Automatic Display Layout (A^2DL) which aims to automate and adapt the layout of the windows on the display screen according to the current user and his current task domain. This conception, realized in terms of an automated window management system, will help free the user's cognitive and temporal resources to work on his actual application domain activities rather than managing the multi–window desktop, thereby enhancing the overall human–computer system performance.

KEYWORDS Adaptive Window Management, Adaptive User Interfaces, Human Factors, Context Determination, User Modeling

1. Introduction

With an ever–increasing rate of advances in computers and display technology, a user is burdened with a cluttered multi–window desktop and has to deal with an annoying overhead of rearranging the display layout (that is, positioning and sizing of windows) quite often to suit his current task or information requirement. As researchers of HCI[1], it is important that the outputs of our research should be in the direction of benefitting humanity and should lead to the development of more usable and humanly acceptable information technology. Keeping this in view and in order to bring about a fusion between cognitive psychology and computer science, we are currently developing an Adaptive Window Manager (AWM), a realization of the idea of Adaptive Automatic Display Layout (A^2DL), where our objective is to relieve the user from the burden of managing the window–based interface.

In our earlier communication (Stille et al., 1996), we have treated in detail the motivation, software ergonomic requirements, the architecture and working of the proposed AWM. In this paper, after a brief overview of AWM, we focus on the discussion of our theoretical and empirical analyses of user interaction patterns with a User Interface (UI) and its representation in the form of a user–interaction model that has directed the architectural design and working of AWM.

[1]This research work is being supported by German Research Foundation under grant no. Er 168/7-2

2. Previous Efforts and New Directions

In our previous work (Lüders and Ernst, 1994), (Lüders and Ernst, 1995a), (Lüders and Ernst, 1995b) we have proposed an approach of Automatic Display Layout (ADL), wherein it has been suggested that certain layout algorithms could be used to automatically organize the windows on the screen to present the currently needed information in order to assist the user in his current activity. The problem of ADL has been formulated as a graph layout problem and treated as a combinatorial optimization problem (Lüders et al., 1995). We have proposed the usage of the simulated annealing (Otten and van Ginneken, 1989) and the p–f (pressure–force) directed algorithms (Fruchtermann and Reingold, 1991) to manage the display layout. The approach to ADL is not only applicable to windows or icons on the screen, but can also be applied to graphical objects within windows. This feature is especially important for hypertext or hypermedia applications. In order to demonstrate and test the suitability and effectiveness of these algorithms to the display layout problem, we have developed two experimental systems: GraphADL and HyperADL (Lüders and Ernst, 1995a). GraphADL is a graph browser that uses the proposed layout algorithms to enable the display of, and navigation in, a graph. HyperADL is a hypertext editor that computes layout for the hypertext cards that are to be displayed on the screen. Low computation times of the layout calculations in these systems indicated that the principle of ADL could be applied to an interactive real-time use.

In our current work, we have propounded the concept of A^2DL where our aim is to extend the usage of the algorithms, as proposed in ADL, for automated window management by applying them to a real–time situation of human computer interaction in a multi–window environment. In the GraphADL and HyperADL systems (Lüders et al., 1995) the user has to specify many input parameters for the layout algorithms, like the optimization criteria, upper bound of the computation time or the maximum number of iterations for the iterative refinement process of the layout to terminate, etc., and the user is thus, expected to know about the concept of ADL, working of these systems and about the layout algorithms. With the basic objective of minimizing the user's involvement in window management activities, our emphasis, in the notion of A^2DL, has been to relieve the user from explicitly specifying any parameters for the layout algorithms. This is because a user is not expected to know the intricacies of the layout computations that take place in the background. To practically implement and demonstrate the proposed idea of A^2DL to organize the presentation of the windows to adapt to the user's current task, we are currently developing the AWM over the public domain window manager - fvwm2 (Nation, 1996) under UNIX/X11.

3. AWM: An Overview

As we prove later in this paper through our user-behavior analyses, a user, during a user–computer interaction, tends to divide his activities in terms of tasks or working situations, which we refer to as "working contexts". Each working context is identified in terms of the applications constituting that task and their states. For example, an editor window may be used in the LaTeX mode for entering text in one of the user's tasks or it may be used in C/C++ mode for entering C/C++ code in another context or it may be unused/iconified in yet another context.

AWM is an auto–adaptive layout system, which would be able to identify the user's current working context, on the basis of each working context's distinguishing constituents and their states, and would be able to adapt the layout characteristics to suit the current user and his current working context. It is *auto–adaptive* in the sense that during the user–computer interaction, it *learns* about the user's layout preferences for his various working contexts from his actions, *updates* this 'learnt' information with the change in user's preferences, and uses this 'learnt' knowledge to *adapt* the layouts to suit the working contexts of the user in his future interactions.

The design and working of AWM is, though, based on a *single model* for all users, as we discuss in the next section, but to cater to an individual user's layout wishes, AWM maintains a profile that is specific to each user which contains the learnt contexts and layout features for each context as per the user's preferred layout characteristics for that context. The layout features consist of the window/icon (object) attributes and the layout parameters, where the possible object attributes are size, width/length ratio, etc. and the layout parameters determine the characteristics of the display layout, like, homogeneous distribution of objects. The user desired layout

features are inputs to the layout algorithms for generating the layout. Thus, when the user is in a same or similar working context that has been learnt by AWM and has been incorporated in the profile for that user, the layout algorithms would take into account the layout features for that context to present the layout. In future sessions of the user, AWM not only learns new contexts and the associated layout features but also updates the information for the previously learnt contexts through feedback (layout corrections or modifications made by the user) and thereby, presents layouts for the learnt contexts which are appropriate for the user and his tasks.

As we intend to apply the notion of A^2DL to a real–time use and also because the display layout determines the UI's look and feel, it is important that AWM meets the software ergonomic requirements from human factors perspective (Stille et al., 1996). The AWM–system should facilitate the ease of handling of the UI, increase user productivity, decrease user mental and physical workloads and, thereby, increase user satisfaction and acceptance. Also, the layout computations should not be time–intensive as the purpose of AWM is to support rather than to interfere with the user's tasks. We have described the architecture and working of AWM in our previous communication (Stille et al., 1996).

In this paper, we discuss our results of the interpretation of user's actions and action patterns with a UI. The findings have been incorporated in a user model that reflects the user's behavior and layout wishes for the windows while performing his tasks. The information contained in this model is a significant input to the efficient designing of AWM to fulfill user's layout wishes optimally resulting in an effective, improved and satisfying human–computer interaction.

4. Analysis of User Behavior

An adaptive user interface system is able to adapt itself to the tasks and wishes of the user (McTear, 1993). To provide this adaptivity and an individualized user interface, it must possess knowledge about the user, such as his preferences, cognitive and learning capabilities, expertise with computers, goals, intentions and expectations (Schneider-Hufschmidt et al., 1993). This knowledge source is a user model and its appropriateness is crucial for user acceptance of the adaptive system as it represents the information that is the basis for adaptation (Kobsa and Wahlster, 1989).

In most of the research prototypes illustrating different adaptive systems (Schneider-Hufschmidt et al., 1993), (Kobsa and Wahlster, 1989), (Oppermann, 1994), a user model is built by classifying the user's expertise on the basis of his behavior and action patterns with the computer system for a particular task, or by querying the user. These user models are application domain–specific and focus on providing adaptivity in various forms to assist the user in the completion of his current task or activity. None of the user models is generic that takes into account how, in general, a user interacts with a UI comprising of multiple windows in a user's task.

There are, however, some results that we have come across during our literature survey on user behavior in a multi-window setup. In (Henderson and Card, 1986), it is described that users normally work with a small set of windows (5 or fewer) for a while, then delete or shrink most of them, and begin building another set of windows. To capture this user behavior of window usage for his activities, windows in the environment have been programmed to "gray out", becoming barely readable after 10 seconds. The windows are instantly restored if user selects them with the mouse or the system uses them. These findings, though illustrate user behavior with a group of windows, do not discuss the aspect of window layout preferences of a user for his tasks.

We, now, describe our approach to the construction of a user model.

4.1 User–Interaction model for AWM

The design and implementation of AWM is based on a "user–interaction" model that has been derived from an analysis of user behavior with a UI in a multi–window setup. The two specific requirements for the user–interaction model of AWM were that it should be universal and not application specific as AWM is proposed to be application independent. Secondly, the results should be detailed enough to define a concrete strategy for controlling the layout parameters.

The model, that we have developed, is non-specific as it does not pertain to each *individual* user or does not model a group of users with common characteristics. Rather it is a *single* user model for all users, a generalized knowledge base, describing how users perform their activities in a multi–window environment.

As per the classifying nomenclature given in

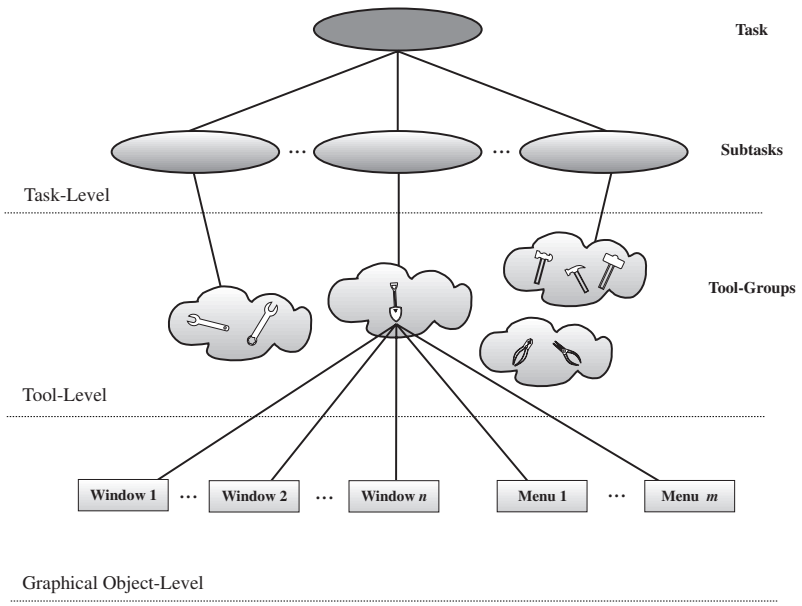

Figure 1: Partitioning of user tasks

(Beale and Finlay, 1992), we have used the approach of *static user modeling*, that is, building a model or profile of a typical user based on the information contained in records or protocols of user's actual interaction with a computer system. The analysis constitutes the study of the way in which a user performs his tasks while interacting with the UI, combination of software tool groups that he uses, approximate number of windows that a user normally uses in his one task, and his manipulation of windows to obtain a preferred layout for a task. The theoretical results have been augmented by empirical observations and we discuss our analyses in the following sub-sections.

4.2 Theoretical Analysis of User Behavior

An interactive system aids a user in accomplishing his tasks. An observational study of user–computer interaction indicates that a user views his work as composed of tasks to achieve certain goals and a goal is the desired output from a performed task. Typical tasks of users could vary from writing a document with illustrations, debugging a program or selecting data from a database to view its graphical representation.

A user divides his job into several tasks and each task may constitute of several sub–tasks. Figure 1 illustrates how a user's task can be hierarchically organized into several sub–tasks achieving certain sub–goals. For each sub–task, the user may choose a software tool from tool group, a group of software tools of the same functionality, that supports his current task in the best possible manner. On a UI level, each tool constitutes of several graphical objects, like windows, menus, list boxes, combo boxes, etc. His task or sub–task–status can be determined by observing the states of the active software tool and the graphical objects of this tool.

The layout wishes of a user strongly depend on his tasks that he is currently performing and on his current working step in the task. To illustrate this, we consider two different tasks that a user may perform on an interactive system where no automatic display layout mechanisms are used.

Let us suppose that the user wants to debug a program. He uses an editor, a debugger and an x–term window in which a compiler can be started. Figure 2 shows a display layout arranged by the user after manipulation of windows. The editor for the source code is placed on

Figure 2: Layout during software development

the left part of the screen and covers a major part of the screen as editing the source code is the most important sub–task to achieve the main task of debugging a program and is therefore, used most often. The debugger–window is placed to the right of the editor window and is also relatively big so that the user can debug the program. The x–term window is used to start the compiler after the sources have been modified and hence, the window is relatively small because while debugging compilation error messages are not expected. Also, it may be noted that the user does not expect any overlapping of windows because access to all windows is needed.

The second task that we examine here, is to write a document using the LaTeX –system and to examine the layout with a previewer and to improve / modify it. Figure 3 illustrates a possible display layout of the windows that are being used for this task. The editor window is on the right top most corner and is not as big as in the first task. The x–term window is placed below the editor window and is bigger than in the previous illustration, as the user would like to view and analyze the layout relevant messages of the LaTeX –system. The previewer occupies most of the screen space as the user wants a view a complete page of the document one by one. If the previewer is not being used, the user might iconify it or put it onto the background in order to have direct access to other windows. In the background, lies the debugger window which might be up because of its usage in a previous task. Since it is not being used for the current task, it lies in the background.

From these two illustrations, we conclude the following: A user performs tasks to achieve certain goals on a computer system. To accomplish a task, a user chooses a set of software tools or applications. Each tool comprises of certain graphical objects. A set of windows and associated graphical objects used in a task comprise a working set. Each task is, therefore, associated with a distinct working set. The state of the tool, or in particular, the state of the graphical objects in the tool indicate the current activity or working step of the user. The user manipulates the layout of windows on the screen (opening, closing, resizing, moving, etc.) to suit his current

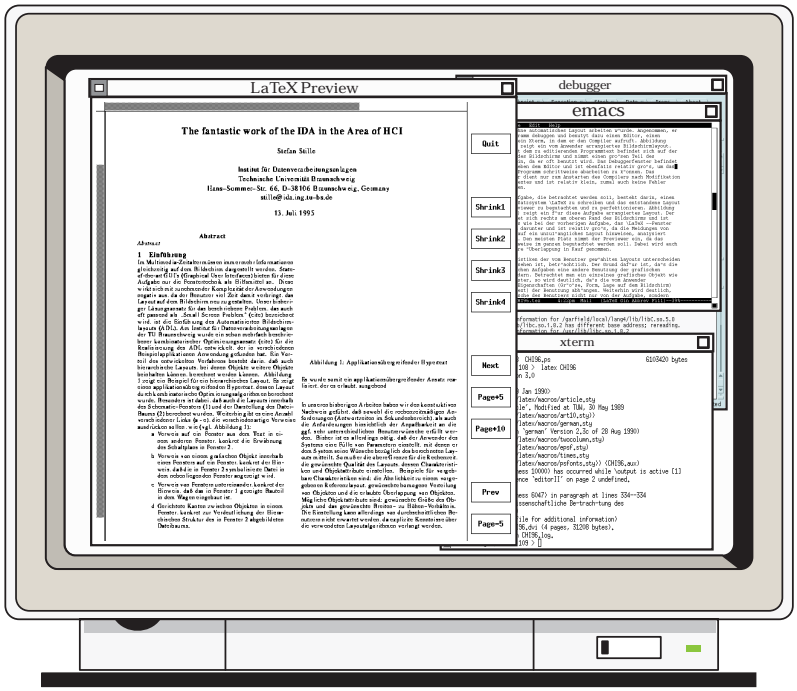

Figure 3: Layout during document creation

context and the current activity within the context. The layout characteristics may be different for every user task and also may vary with the current activity within a task. For example, the user may accept the overlapping of windows while viewing the text in the previewer window in the LaTeX task discussed above but an overlapping may be unacceptable while entering text in the editor window. However, each task or working context of the user is distinguishable in terms of the working set and the state of the constituents of this working set.

4.3 Empirical Analysis of User Behavior

We have performed an empirical study to reinforce our theoretical analysis of user behavior. Our source of information has been records or protocols of user interaction with the computer system. Such protocols can involve employing a number of recording media, including audio and video tape, or system recorded traces of activity of the users. We have used the logged information of the users' interaction under UNIX/X11 over fvwm.

The aim of our study was to verify our theoretical finding that a unique set of windows comprise a working set. This was empirically determined on the basis of the usage of windows in a working context. We considered those windows as a part of the working set, which the user history indicated, were being used to solve the user's current task and had been used over a significant amount of time. The window getting the keyboard focus for a considerable amount of time is an indication of its usage. The instances where a window was chosen for a short while moving the mouse / focus over the window were ignored because this window was not actually being used.

In Figure 4, the total number of windows on the screen and the number of windows that are actually a part of the user's working set are plotted for all the 17 test users based on our observations over the users' sessions. This figure indicates that none of the users use all the open windows on the display screen at one time and, therefore, not all open windows are a part of the working set. For all the test users and over all the sessions, we have computed that 4.09 windows form a part of the working set when about 15.59 windows are open.

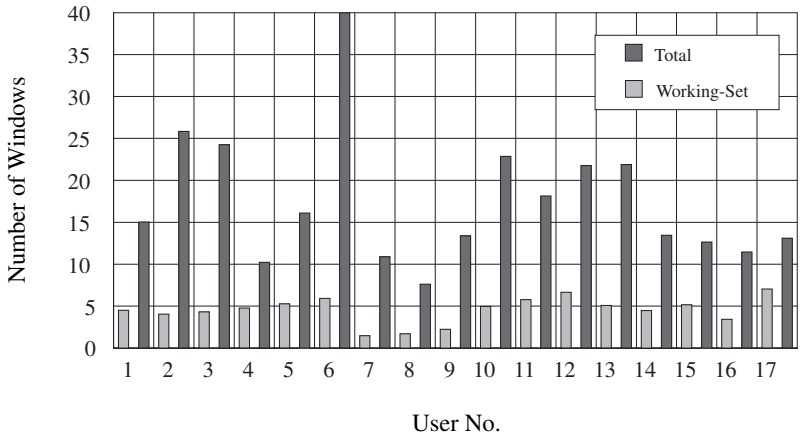

Figure 4: User number vs. number of used windows

Our theoretical analysis indicates that to accomplish his tasks or goals, a user works in terms of distinct contexts and a unique set of application windows comprise a working set for a context which we confirmed empirically as well. A user switches between windows with a change in working context and performs layout actions to suit his current context.

In order to provide a layout that conforms to the user's working context, AWM must have a mechanism to identify and classify the contexts and to learn the user's layout wishes for each context. The classification and learning of the working contexts of the user in AWM is through neural networks as neural networks are useful for real–time pattern recognition. We have used the Adaptive Resonance Theory–(ART)– neural networks (Stille et al., 1996) which are self-organising networks that have the ability to create a new pattern category when presented with a novel stimulus while preserving their previously learnt knowledge (Carpenter and Grossberg, 1987), (Carpenter and Grossberg, 1988).

After the identification of context, it is necessary to match the object/window attributes stored in the reference database of AWM (Stille et al., 1996) for that context to the windows on the screen for performing layout calculations and to bring about a change in the layout for that context. We are currently facing the difficulty of an exact matching due to insufficient identifiers associated with the windows of the applications. At present, we can only approximately relate the object on the screen with the stored object attributes. We are exploring this aspect to find out unique object identifiers for accurate recognition.

5. Conclusions and Future Work

In this communication and in (Stille et al., 1996), we have addressed the problem of automating the task of managing the user interface in order to assist the user and have proposed an automated display layout system which would organize the presentation of the windows to conform to the user's current task.

The concept of the A^2DL to adapt the display layouts as per the user's current task is based on our theoretical and empirical analyses of user behavior with an interactive system in a multi-window setup. In this communication we have discussed our analyses that has led to the formulation of a "user–interaction" model which has directed the design of AWM. Our current work status and detailed analysis of each of the architectural components will be presented shortly in our other communications.

We intend to design and implement a UI for AWM. The UI would enable the user to initiate or inhibit the layout actions performed by the system. It would also allow the user to specify any distinct identifier, for example, a context name, to his working contexts. When the user is in the same or similar working context, the user may give this identifier to obtain the corresponding layout of

the windows on his display screen. The display of this identifier on the screen would also be an indicator of the layout that has been performed as a result of layout calculations. Through the UI, the user would also be able to specify any special requirements or layout preferences which would then be taken into account by the system and reflected during the user's future interactions.

Our current efforts are aimed to have a working prototype of AWM ready to evaluate the system by performing usability studies with actual users through observations, questionnaires and interviews to discover whether the proposed system can really increase the convenience of use and enhance the user's efficiency while working with a multi-window system aided by AWM.

On the performance front, we aim to use the multithreading feature of the present-day operating systems to take advantage of modern multiprocessor workstations. This would not only speed up the layout computations but also enable them to be performed in the background resulting in improved response times of AWM to user's actions without degrading the performance of user's actual application domain activities.

References

Beale, R. and Finlay, J. (1992). *Neural Networks and Pattern Recognition in Human Computer Interaction*. Ellis Horwood Ltd.

Carpenter, G. A. and Grossberg, S. (1987). A massively parallel architecture for a self–organizing neural pattern recognition machine. *Computer Vision, Graphics and Image Processing*, 37:54–115.

Carpenter, G. A. and Grossberg, S. (1988). The art of adaptive pattern recognition by a self–organizing neural network. *IEEE Computer*, pages 47–88.

Fruchtermann, T. M. J. and Reingold, E. M. (1991). Graph drawing by force–directed placement. *Software Practice and Experience*, 21(11):1129–1164.

Henderson, D. A. and Card, S. K. (1986). Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention in a Window–Based Graphical User Interface. *ACM Transactions on Graphics*, 5(3):211–243.

Kobsa, A. and Wahlster, W., editors (1989). *User Models in Dialog Systems*. Springer Verlag, Berlin, Heidelberg.

Lüders, P. and Ernst, R. (1994). The dynamic screen – beyond the limits of traditional graphical user interfaces. In *Proceedings of 13th IFIP World Computer Congress, Hamburg, Germany*.

Lüders, P. and Ernst, R. (1995a). Improvement of the user interface of multimedia applications by automatic display layout. In *Proceedings of Multimedia and Networking Conference, San Jose, CA*.

Lüders, P. and Ernst, R. (1995b). Improving browsing in information by the automatic display layout. In *Proceedings of the IEEE Symposium on Information Visualization, Atlanta Georgia, USA*, Los Alamitos, California, USA. IEEE Computer Society Press.

Lüders, P., Ernst, R., and Stille, S. (1995). An approach to automatic display layout using combinatorial optimization algorithms. *Software Practice and Experience*, 25(11):1183–1202.

McTear, M. F. (1993). User modelling for adaptive computer systems: a survey of recent developments. *Artificial Intelligence Review*, 7(3-4):157–184.

Nation, R. (1996). *fvwm – F (?) Virtual Window Manager for X11*. obtained by ftp - ftp://ftp.hpc.uh.edu/pub/fvwm/version-2/.

Oppermann, R., editor (1994). *Adaptive User Support–Ergonomic Design of Manually and Automatically Adaptable Software*. Lawrence Erlbaum Associates, Publishers.

Otten, R. and van Ginneken, L. (1989). *The Annealing Algorithm*. Kluwer Academic Press.

Schneider-Hufschmidt, M., Kühme, T., and Malinowski, U., editors (1993). *Adaptive User Interfaces-Principles and Practice*. North-Holland Elsevier Science Publishers B.V.

Shneiderman, B. (1992). *Designing the user interface: strategies for effective human–computer interaction*. Addison–Wesley.

Stille, S., Minocha, S., and Ernst, R. (1996). A^2DL–An Adaptive Automatic Display Layout System. In *Proceedings of the IEEE 3rd Annual Symposium on Human Interaction with Complex Systems (HICS 96)*, Dayton, OHIO, USA. IEEE Computer Society Press.