Adaptive Toolbars: An Architectural Overview

T. Miah, M. Karageorgou and R.P. Knott

LUTCHI Research Centre
Department of Computer Studies
Loughborough University
Loughborough, Leicestershire LE11 3TU, UK.
Email: [t.miah, m.karageorgou, r.p.knott]@lboro.ac.uk

Abstract. Applications today are 'Packed' with toolbars. If we take a count of the number of toolbars in some of today's application, we see that MS-Word has 9 toolbars, MS-Excel has 13, and MS-PowerPoint has 7. One obvious question to ask is; do users really use all of the toolbar items available in each application, or are they just cluttering the screen?

This paper presents a technique for automatically adapting the toolbar to user needs. For example as users work through a document in MS-Word, they may select BOLD from the format menu and not use the toolbar, or the toolbar with the BOLD icon is not displayed. As the user continues to use the command repeatedly from the menu bar the software should detect this usage and display the appropriate toolbar.

A system for adapting the toolbars for MS-Word has been designed, implemented and preliminary evaluation carried out. The result of the evaluation showed that on average most subjects prefer the adaptive version of Word. These subjects were mainly from the novice and expert group of users.

1. INTRODUCTION

Current software systems are rather complex, offering the user many choices, configurations and options; but they often leave the users to handle this complexity on their own. One of the main objectives of system design is to make it easier for the users to use or operate the device, whether this is a large computer system, a video recorder or a programmable calculator. We believe the system should be relatively simple to use and the complexity hidden from the user. One way of achieving this is by effective user interfaces, which utilise appropriate metaphors.

Each user is different and has different conception of complexity. Although groups of users may have similar views, the exact nature of their view may vary between the individuals within the group.

Some common problems of user interfaces are:

1. The number of different menus and toolbars with various options and icons may clutter the workspace, and it appears that considerable learning is required before someone is ready to use a specific system. Even if users are trained to use the system, it still remains complicated and requires the user to have comprehensive knowledge of the application for it to be used efficiently.
2. The majority of users do not know all the possible options that the system can support. This may be due to presentation of the options, users limited short term memory, infrequent use of certain functions or options, or a combination of these factors. Users tend to limit themselves to a few basic options or functions that are familiar to them.
3. Often it is not easy to understand the function provided by a button simply from its icon. This may lead to a wrong choice of button, and may have an adverse effect on the user, resulting in them preferring menus rather than toolbars.

To reduce the complexity of an interface while at the same time catering for this individuality of each user, commercial software allows user customisation of the interface. Users are allowed to change menus, add macros, assign buttons to toolbar, and rearrange elements on the screen. However, the problem is not totally solved by allowing users to customise their software but transferred from the designer of the system to the end user. Apart from the time required to customise, adapt or adjust the interface, the task of customising is not as simple as it seems. The user cannot have sufficient knowledge to be able to start customising unless the user is competent on that particular application and the task.

One plausible solution is to give the system the ability to decide and change the interface for the user automatically according to the user needs. Here, the main concern from a design viewpoint is how does the system decide what the user needs. Obviously it must learn from the users work patterns or habits or employ the use of a user model to determine the needs of users (some examples of user model based systems include; Benyon and Murray (1993), Sukaviriya and Foley (1993), Crow and Smith (1993).

However, adaptive systems are not with out problems; they may cause the user to feel a loss of control or distract the user from the primary task. These systems some times may make wrong judgments about the users needs and adapt the system to completely different or undesired functions. This has an adverse effect on user performance.

These problems must be considered when designing interfaces and in particular when designing systems that adaptively changes the interface.

The problem of loss of user control in an adaptive system is not an easy problem to solve. Obviously, to give the user some degree of control the system could ask the user whether to accept or reject the adaptation suggested by the system. Examples of such systems are; Adaptive Forms-Malinowski (1993); Flexcel - Thomas and Krogsæter (1993); Eager - Cypher (1991). However, in doing so, the dialog between the user and the system is extended and can impede on users who work under time pressure. Even for a simple adaptation, the user has to respond by saying, “yes, please adapt the system”. This can get quite annoying and cumbersome. There must be a trade off between the user feeling a loss of control and the user not being frustrated by repeatedly being asked whether to accept or reject adaptation. It is highly plausible for the system to do some limited adaptation automatically, especially for adaptation that is reversible, as the user can always undo the adaptation, should it not fulfill the needs of the user.

A good system must encompass a combination of features from adaptable and adaptive system components. This will give the users the option to make their own changes according to their needs, as well as the system adaptively changing the interface automatically and offering advice to the user at other times instead of adapting automatically.

2. ADAPTIVE TOOLBARS

The primary motivating factor of this work, is to overcome information overload caused by displaying too many toolbars and at the other extreme, displaying none or inappropriate toolbars. For example, as a user works through a document in MS-Word, the user may select **BOLD** from the format menu, either instead of using the toolbar or because the toolbar with the **BOLD** icon is not displayed. As the user continues to use the **BOLD** command repeatedly from the menu bar, the software should detect the usage and display the appropriate toolbar. Similarly, if the displayed toolbar is not be utilised by the user then the toolbar should be removed.

Should the toolbar be removed is a very interesting question, we can not simply start adding more and more toolbars as the user makes use of these commands, until all the possible toolbars are displayed but this defeats the objective of this work. However, we can not simply start removing toolbars, when the system feels like it. How does the system really know that the toolbar is not going to be required subsequently?

To overcome this problem the system sets a limit on the number of toolbars that can be displayed at any one time. For example if this limit is set to three, this effectively means that there are three slots available in the display area for toolbars. If all display slots have been used and a fourth toolbar is required to be displayed, then one toolbar must be remove. How do we, and more importantly how does the system decide on which toolbar to remove? The system must decide on the relative '*importance*' of each of the toolbars as deemed by the user and remove the least important toolbar.

2.1 How do we determine the "Importance" of a Toolbar?

Not all toolbars available in an application are equally valuable to the user. The value of a toolbar is dependent on the task being performed. Without employing heavy use of a task model, an estimation of the importance of a toolbar as deemed by the user can be calculated from a number of variables and is referred to as *Toolbar Importance*. The calculation of toolbar importance has been adapted from Funke et. al. (1993).

The toolbar importance variable is used to determine which toolbar will be removed from the toolbar display area. If the maximum allowable toolbar limit is reached and a new toolbar is to be displayed, then space must be created for the new toolbar. The system calculates the importance of each of the toolbars currently displayed and removes the one with the least important value, as determined by this variable.

Toolbar importance is based on information displayed on the toolbar, it's relative value for the undertaken task, and the extent of the user's focus on the toolbar while performing the task. For implementation, these criteria are to be determined from the following variables: time of creation (i.e. time at which the toolbar was displayed); time elapsed since last interaction; frequency of interaction.

A toolbar interaction occurs whenever the user presses a toolbar button. Therefore, the importance of a toolbar can be expressed as:

$$\boxed{\mathit{Toolbar}\ \mathrm{Im}\,\mathit{por}\,\mathrm{tan}\,\mathit{ce} = K_1\left(\mathit{TimeOfCreation}\right) + K_2\left(\mathit{TimeSinceLastInteraction}\right) + K_3\left(\mathit{FreqOfInteraction}\right)}$$

Equation 1: Calculating a Value for the Importance of a Toolbar

The coefficients of the terms indicate the relative weight assigned to the respective factor of toolbar importance. For the present system it has been decided to keep the coefficients for each term the same, until further investigations are carried out, regarding the influence of the relative weighting on toolbar importance and the accuracy of this parameter in predicting the intentions of user needs. Also, once the relationships between these variables, the user and the accuracy have been formulated, further adaptation can be applied on these variables so the system adapts to individual users and modifies it's values over time.

Time of Creation This variable represents the importance based on how recently it was created. The importance, as perceived by the user, decreases with time. This may be derivative of the findings that information in short term memory tends to decay exponentially [Kyllonen and Alluisi (1987)]

$$\boxed{\mathit{TimeOfCreation} = e^{\frac{-\left(\mathit{CurrentTime} - \mathit{TimeCreated}\right)}{C}}}$$

Equation 2: Toolbar Creation Time

The value of *TimeOfCreation*, (Equation 2), is an exponential function of time (i.e. the current time minus the time at which the toolbar was displayed divided by a constant C). The exponential power is divided by C to obtain a smooth and reasonable decrease in the value of *TimeOfCreation*.

Time Since Last Interaction. This measure the importance based on how recently the toolbar item has been used by the user. The more recently a toolbar has been utilised, the more important it is considered by the system.

$$\boxed{TimeSinceLastInteraction = e^{-(CurrentTime - TimeAtLastInteraction)}}$$

Equation 3: Time Since Last interaction

Frequency Of Interaction This is also a component in assessing toolbar importance. The *FreqOfInteraction* represents the importance of a toolbar based on how often it is used. The more often a toolbar is used, the more important it is considered by the system.

$$\boxed{FreqOfInteraction = \frac{NumberOfInteractionSinceToolbarCreation}{CurrentTime - TimeOfCreation}}$$

Equation 4: Calculating the Frequency of Interaction with a Toolbar

This factor balance against the weight assigned to the *TimeSinceLastInteraction* (see above). For a toolbar that has been used frequently, if the *TimeSinceLastInteraction* indicates that it should be removed, *FreqOfInteraction* says, “let’s not be too quick to reach a conclusion.” It causes toolbars that have been important, in terms of their frequency of use, to be removed only after more careful consideration of their ongoing use. For an alternative algorithm see Debevc et. al. (1996).

2.2 System Architecture

The architecture shown in Figure 1 shows a generalised concept for adapting toolbars of many applications. The same technique can be used for each individual application. However, an environment like Microsoft office provides for an integrated adaptive toolbars system to be developed taking advantages of user command actions made in one application to infer adaptation for other applications. Although there may be subtle differences in user command action between different applications, there are also, similarities as there are similar functions available. For example, copy/paste is available in almost all applications written today.

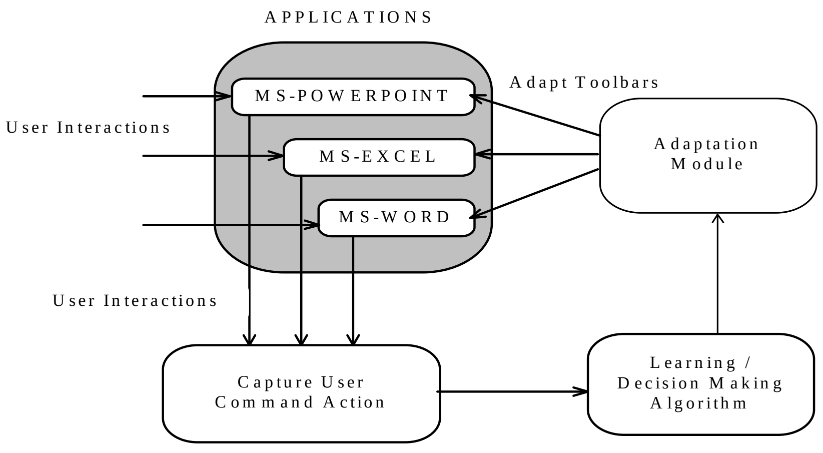

Figure 1: Architecture for an Adaptive Toolbar System

2.2.1 Detailed Architecture of Adaptive Toolbars

The heart of the system is the *Decision Module,* which uses the algorithm detailed above to reach a decision as to which toolbar/dynamic button to add/remove. User actions are captured via the use of macros. Each MS-Word command can be converted to a macro and additional functionality added. The additional functionality added to each command/macro is simply to record facts to a file (Dynamic Knowledge Base). These facts relate to; the command the user executes; the time of execution; and whether execution is from the menu bar or the toolbar. If the command is used from the toolbar then it records the interaction with the toolbar. It records all the parameters that are required to calculate the importance of a toolbar.

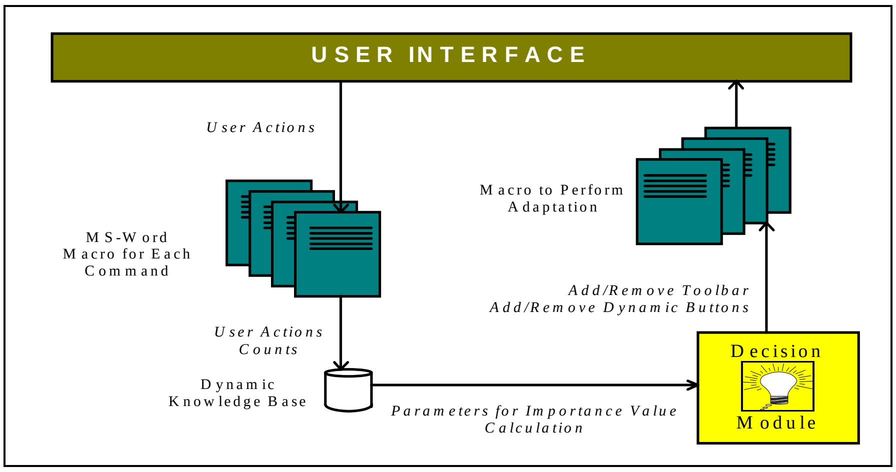
USER INTERFACE
User Actions
Macro to Perform Adaptation
MS-Word Macro for Each Command
Add/Remove Toolbar
Add/Remove Dynamic Buttons
User Actions Counts
Decision
Module
Dynamic Knowledge Base
Parameters for Importance Value Calculation

Figure 2: Data Flow for Adaptive Toolbar System

The decision module uses the parameters to calculate the importance of each of the displayed toolbars. When a decision is made, Word macros add/remove toolbars or dynamic buttons from the interface. These macros also provide some user feedback as to which toolbar/button is being removed and which is being added.

2.3 The Dynamic Knowledge Base

A simple knowledge base is created to hold information on user action. Because the knowledge required for the system is simple in nature, there is no need to implement a very sophisticated database to store the information. The knowledge base is implemented as a structured file.

The file contains the following information:

- ***A count of menu utilisation for a toolbar command.*** Counts the number of times the menu is used in preference to the toolbar.
- ***Data associated with each of the toolbars.*** Each toolbar has two data field associated with it: time created and time at last interaction. These values are used later by the decision module to determine the importance of the toolbar, which is subsequently used to determine which one, if any of the toolbars should be removed from the currently displayed toolbars.
- ***Data associated with each of the commands.*** This is similar to that held for the toolbars. Namely the time of creation and the time of last interaction. This applies only to the commands on the dynamic toolbar. This data is used to calculate the importance of the button.

3. SYSTEM OPERATION

The system provides two adaptation features; the dynamic toolbar buttons, similar to Debevc et. al. (1996) and the adaptive toolbar.

The dynamic toolbar buttons has a simplistic learning algorithm, it creates an icon for every command executed by the user, if it does not already exists on the toolbar (see Figure 3). If it exceeds the limit of the number of icons allowed to be displayed then it removes the least important icon and the new icon is inserted.

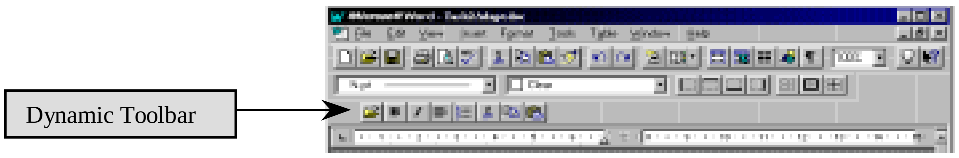

Figure 3: Dynamic Toolbar in Operation

The adaptive toolbars feature removes or adds a toolbar as and when the system thinks the user needs a particular toolbar. Users are made aware of an insertion or removal of a toolbar by flashing an icon. Shneiderman (1992) detailed a number of techniques for getting the user's attention one of which is blinking/flashing and suggests a frequency of 2 to 4 hertz. An icon in the form of an arrow pointing to the right is used to indicate an outgoing toolbar and an arrow pointing to the left to indicated an incoming toolbar (see Figure 4).

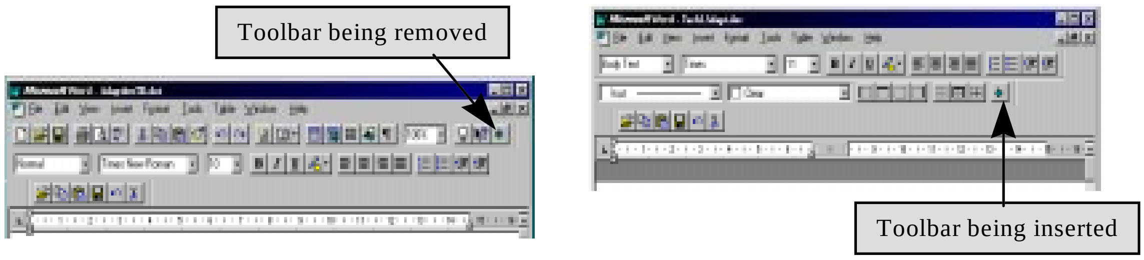

Figure 4: Adaptive Toolbar in Operation

A controversial issue on positional consistency of icons on the dynamic toolbar and possibly the adaptive toolbar comes into questions. In a study in which a pull-down list of food items was resequenced to ensure the most frequently selected items moved toward the top, Mitchell and Shneiderman (1989) found that users performed better using the static menu as opposed to the resequenced menu. However, evidence in favour adaptation was found in a study of a telephone book menu tree that had been restructured to make frequently used telephone numbers more easily accessible (Greenberg and Witten 1985). This shows that there are some applications where adaptation works and others where it hinders. We need to identify these applications and the criteria under which adaptation works and does not work.

4. CONCLUSIONS

A system based on the architecture described in this paper has been implemented and some preliminary evaluation carried out. The preliminary evaluation was in the form of subjective comments of users after a demonstration of the system. Most of them liked the idea of adaptive toolbars and were especially in favour of the dynamic toolbar. Most of them saw the potential of the dynamic toolbar as most of their commonly used commands would be placed on to this toolbar and the standard toolbar provided by MS-Word could be removed.

The technique of adaptive toolbars becomes more important on smaller screen devices. With a smaller screen, the screen real estate is already limited and valuable space can be taken up by toolbars. These techniques can be applied on applications developed for PDA's (Personal Digital Assistant).

5. FUTURE WORK

Develop Improved Learning Algorithm

The system developed so far uses a simple algorithm to decide when a toolbar is required by simply counting the number of times menu command are being used when a toolbar could have been used but not displayed. However, this does not take into consideration the time lapsed between such interaction. One possible extension to the algorithm is to decrease the counter as well as increasing it. The counter begins to decrease after a time out period, should no interaction take place.

Investigation into the Accuracy of the Learning Algorithm

A formal and thorough investigation into the accuracy of the learning algorithm can be carried out. A standard measure for the accuracy can be developed and utilised to compare several different algorithms. These can form the basis for a standard way of benchmarking learning algorithm for applications of such nature.

REFERENCES

Benyon D. and Murray D. (1993). *Developing Adaptive Systems to Fit Individual Aptitudes.* Proceedings of the 1993 International Workshop on Intelligent User Interfaces. W.D. Gray, W.E. Hefley & D.Murray (eds.). Orlando, Florida. New York. ACM Press (1993)

Crow D. and Smith B. (1993). *The Role of Built in Knowledge in Adaptive Interface Systems.* Proceedings of the 1993 International Workshop on Intelligent User Interfaces. W.D. Gray, W.E. Hefley & D.Murray (eds.). Orlando, Florida. New York. ACM Press (1993)

Cypher A. (1991). *EAGER: Programming Repetitive Tasks by Example*. Proceedings of CHI ’91, ACM, New Orleans, May 1991, pp. 33-40

Debevc M., Meyer B., Donlagic D., and Svecko R. (1996). *Design and Evaluation of an Adaptive Icon Toolbar*. User Modeling and User-Adapted Interaction 6: 1-21,

Funke D.J., Neal J.G., Paul R.D. (1993). *An Approach to Intelligent Automated Window Management*. International Journal of Man-Machine Studies 1993, Vol.38, pp. 949-983.

Greenberg S. and Witten I.H. (1985). Adaptive Personalised Interfaces: A question of viability, Behaviour and Information Technology 4, 1 (1985), 31-45.

Malinowski U. (1993). *Adjusting the Presentation of Forms to Users Behavior.* In Proceedings of the 1993 International Workshop on Intelligent User Interfaces. Orlando, Florida. Hefley W.E. and Murray D.(eds.), New York: ACM Press (1993).

Mitchell J. and Shneiderman B. (1989). Dynamic verses Static Menus: An experimental comparison, ACM SIGCHI Bulletin 20, 4 (1989), 33-36.

Shneiderman (1992). *Designing the User Interface: Strategies for effective Human-Computer Interaction (2nd Edition).* Addison-Wesley Publishing Company. ISBN 0-201-57286-9.

Sukaviriya P.N. and Foley J.D. (1993). *Supporting Adaptive Interfaces in a Knowledge-Based User Interface Environment.* Proceedings of the 1993 International Workshop on Intelligent User

Thomas C.G. and Krogsæter M (1993). *An Adaptive Environment for the User Interface of Excel*, Proceedings of the 1993 International Workshop on Intelligent User Interfaces. Orlando, Florida. Hefley W.E. and Murray D.(eds.). New York: ACM Press (1993)