Proceedings of 11th International Conference on Human-Computer Interaction, Volume 7, July 2005.

Deciding Layers: Adaptive Composition of Layers in a Multi-Layer User Interface

Bryan Clark

Red Hat
10 Technology Park Drive
Westford, MA
bclark@redhat.com

Jeanna Matthews

Clarkson University
8 Clarkson Avenue
Potsdam, NY 13699
jnm@clarkson.edu

Abstract

We discuss the composition of layers in a Multi-Layer User Interface. Specifically, we define Features Layers to be layers composed of features from a single category (e.g. photo editing features might make up one feature layer and text formatting features another). We contrast this with Mixed Layers or layers that contain a mix of features from many categories. The finer-grained Feature Layers give users greater control over the exact features added to the application. We also present two new methods for automatically adding new layers to an interface based on the user's prior experience with similar applications and based on how the application is launched. We describe a blogging application we have implemented to explore these concepts.

1 Introduction

Feature-rich applications often accomplish more tasks than any single user requires. Although users approach such an application with many different goals, the application is still presented to each user with the same interface. Exposing every feature to all users produces a "bloated" collection of menus and toolbars. By making it possible to do many things, the designers make it difficult to actually do any one thing!

Interface designers attempt to tailor these complex applications to suite the majority of users. This process leaves an application that is not designed for the first time user nor for the advanced user, but rather for the mid-level user. First-time users are left struggling to understand the basic features they need to accomplish simple tasks. First-time users of an application do not want to spend the time to learn how to use the software properly; they are most concerned with getting their task finished. A recent study has shown that those who are new to computers (Kraut, Scherlis, Mukhopadhyay, Manning & Kiesler, 1996) are not taking advantage of all that computers and the Internet have to offer because of frustration with complicated software.

Studies have shown that even advanced users waste 45% of their time with frustrating interface experiences (Ceaparu, Lazar, Bessiere, Robinson & Shneiderman, 2004). Problems reported by advances users include indecipherable error dialogs and hard to find functions and menus.

It is important to note that "advanced" users of complex applications are often advanced in a single category of the applications features and not others. For example, a photographer using presentation software may desire more photo editing options from their interface, while an editor using the same software may desire text-formatting options. Advanced users often need only a subset of functions, while applications display the complete set of functions in a maze of menus to search through.

Interface designers target the mid-level user because it is simpler than designing custom interfaces for each user group. Creating custom interfaces for first-time users, mid-level users and users advanced in every subset of an application's functionality would be overly time-consuming and costly.

Multi-Layer Interface design (Shneiderman, 2003) is an Adaptive User Interface strategy for satisfying the needs of first-time through advanced users all within the same application. Designers create a single interface, but group features into layers. First-time users begin at this first and simplest layer where the core functionality of the application is clearly exposed without many additional options. Then as users gain experience with the application and desire additional features, they can move up to a more advanced layer. Each new layer can add new features and/or present different existing features in a new way corresponding to the users' understanding of the application. The highest layer provides the most features and functionality possible and is designed with the most experienced user in mind.

We begin with Shneiderman's description of Multi-Layer interface design and further define two distinct categories of layers within a Multi-Layer system – Feature Layers and Mixed Layers. We define Features Layers to be layers composed of a single category of features (e.g. photo-editing features in one feature layer and text formatting features in another). We define Mixed Layers to be layers that present advanced levels of features from multiple categories in the same layer (e.g. a layer that adds a few advanced photo-editing features together with a few text formatting features).

Mixed Layers correspond most directly to the examples presented in (Shneiderman, 2003) where applications have a slider bar by which users can advance through a hierarchy of layers. We argue that Feature Layers can provide a more natural user interface by allowing users to advance farther in some feature sets than others. Mixed layers require users to advance on all fronts and make it impossible for users who are advanced in one aspect to avoid cluttering their interface with every possible option.

In addition to this definition of Mixed and Feature Layers, we present two new methods of automatically adding layers to a user interface – specifically progression based on how the application is launched and progression based on assessing a user's prior experience with other similar applications. If the application can detect previously created content on its first run, the application can assume user is experienced with that type of content and will add interface layers to complement the users' experience. A second method of progression is done from application launch. As the user launches the application from different file types, via a right click menu option, the interface adapts to show features relevant to the type of file used to launch the application.

To demonstrate the new definitions of layers and the new methods of dynamically adapting those layers in a multi-layer interface we have built an example implementation. The example application is a Blog-Editor, an application for posting entries into a person's web-log

(Wentworth, 2003). The application is a simple text editor with features to change text alignment, add styled text, links and images.

2 Feature Layers and Mixed Layers

The original work on Multi-Layer Interfaces was presented with a sketch of a two different application designs, given to promote the idea and encourage the adoption of the system. For example, one sketch showed a simple word processor with 8 layers gaining more and more features as the user increased the slider attached to the windows right-hand side to a higher level.

To highlight the difference between the original multi-layer design ideas and ours we introduce two new definitions, Mixed Layers and Feature Layers. In Mixed Layers, dissimilar features are grouped together in one layer. With each layer added to the interface, the user is presented with some features they want and some they may not be ready for. For example, a user of a word processing application may be presented with advanced printing options at the same time they are presented with advanced document editing features like table editing. If the user has only the need for advanced document editing features but advances on features like printing, then they will have to learn both at the same time, when this may not be their goal.

Table 1 describes a sample set of features that might be present in a blog editor with a Mixed Layer interface. Each layer may consist of font style features, text alignment features, list features, link features and picture features. For example, at the lowest layer (layer 1), users have several font features including bold, italic and underline. They also have a list feature, bulleted lists and a link feature, creating links in a blog. Users have no photo handling features at layer 1.

At layer 2, users gain both additional font editing features (underline and strike through) as well as text alignment features (left, center, and right justification). They do not gain additional list or link features relative to layer 1 and they still have no photo handling features.

Notice that these layers seem somewhat arbitrary. Layer 2, for example, was constructed by the designer with the assumption that editing or adding pictures to a web-log is an advanced task that would be most understood by a user who understands how to work with styled text, aligned text, lists, and links first. However, there is no reason to force a user to add font style and text alignment features before picture manipulation. In this Mixed Layer approach, the user is forced to follow the designer’s concept of which features are easy and which are advanced.

Mixed Layer	Features and Feature Levels in Layer				
	Font Style	*Text Alignment*	*Lists*	*Links*	*Pictures*
1	**Bold**, *Italic*, Underline	None	Bulleted List	Create Links	None
2	**Bold**, *Italic*, Underline ~~Strike-Through~~	Left, Center, Right	Bulleted List	Create Links	None
3	… Program Code	Left, Center, Right, Justified	Bulleted, Numerical List	Create Links, Anchors	Add Pictures
4	… Program Code	Left, Center, Right, Justified	Bulleted, Numerical List	Create Links, Anchors	Add, Edit Picture Sizes
5	… Program Code	Left, Center, Right, Justified	Bulleted, Numerical List	Create Links, Anchors	Add, Edit, Align Pictures

Table 1 - Mixed Layers groups multiple feature sets together at different levels to create a single layer of the interface

Feature Layers, on the other hand, consist of similar features (e.g. all the photo editing features). Within a feature layer, there can be multiple levels. Similar features that not easily learned together are placed in a higher level of the same feature layer. An advantage that Feature Layers have over Mixed Layers is the amount of control given to the user instead of the application designer. Feature layers are smaller pieces of functionality that allow the user to choose the exact set that they would like. Mixed layers require the application designer to know what features would best suite the user, with each layer the user advances on all fronts rather than just a single group of similar features.

Table 2 shows the layout of the Feature Layers in our Blog-Editor with features layers on the left column and the available advanced levels in the columns to the right. Feature Layer levels can have numbers associated with them, however this is simply for easy reference and carries no actual meaning to the numerical values of the level. Recall that a fundamental Feature Layer idea is that any layer can be added with any combination of levels, not just linearly. For example an interface may contain the highest level, 'Program Code', of the feature layer 'Font Style' and not contain any 'Text Alignment' or 'Links' features.

Feature Layers	**Layer Levels 1 →**	**2 →**	**3 →**
Font Style	**Bold**, *Italic*, Underline	~~Strike-Through~~	Program Code
Text Alignment	Left, Center, Right	Justified	
Lists	Bulleted Lists	Numbered Lists	
Links	Create Links	Create Anchors	

Pictures	Add Pictures	Edit Picture Size	Align Pictures

Table 2 - Feature Layers and Layer levels are individually chosen by the user

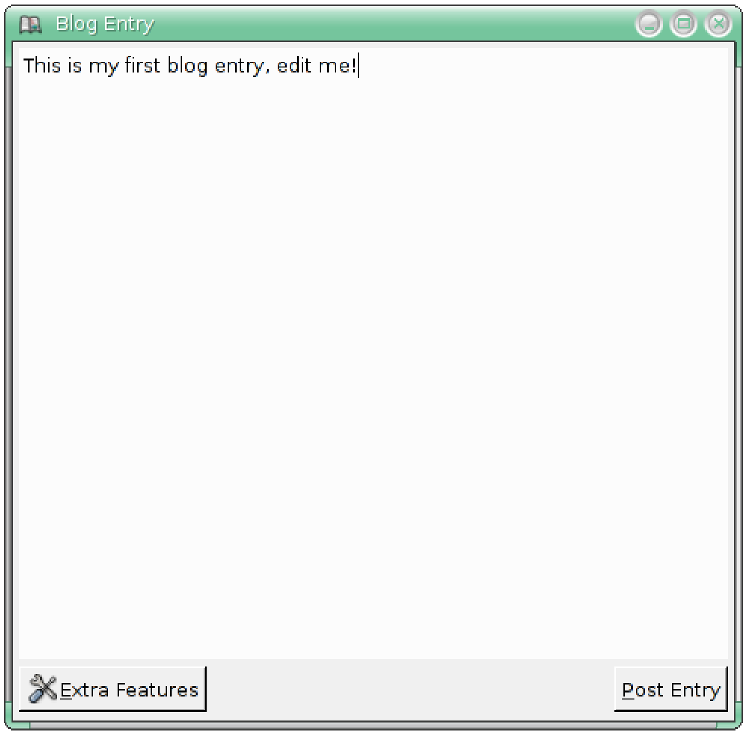
By having layers of similar features the user has more control over what features are present in their interface as new layers are added. Font styles bold, italic and underline were grouped together into a layer because they are all very similar and easily learned at the same time. However the style option of "pre-formatting", often used to show program code or other text with mono-spaced format needs, is not always understood by beginning users but is useful to advanced users so this was placed at level 3 of the font Feature Layer.

The feature layer system imposes fewer unnatural restrictions on the user. The designer still decides with a group of similar features which are more advanced. However, designers do not choose which sets of dissimilar feature are more advanced nor do they force users to acquire all advanced features when they only want a specific set of advanced features.

3 Progressing through Layers in our BlogEditor

3.1 Starting at a Layer 1

The first layer of our interface is the simplest and easiest layer to understand and use. With this simple interface the user is able to do the most basic posting into their web-log. There are no advanced features to distract a first time user from accomplishing the essential task.



This is my first blog entry, edit me!

Figure 1 - The first use interface with no Feature Layers added

Immediately, there are only the basic components of a web-log interface, a text entry area and a button to post the web-log entry. There is a text entry area that accepts any kind of text the user wishes to enter and displays in a WYSWIG form. Initially the text area is filled with the text "My first web-log entry" with which the user can replace or add to on their first post. When the user pressed the "Post Entry" button to post their entry the application sends their text to be posted to the web-log. All additional features of the Blog-Editor are hidden at first use to make the basic functionality of this application clear.

3.2 Adding Layers By Assessing Prior User Experience

Adaptive user interfaces traditionally change in response to the experience the user has with only the existing application. Experience users may have with similar applications is ignored. For example, an adaptive user interface would treat a user with extensive experience with another word processor the same way they treat a user who has never used any kind of word processor.

We propose changing the interface in response to the prior experience the user has with the task the application performs. On first use of our application, the interface attempts to adapt to the users experience with the task the application performs as they have no experience yet together. The analysis of the users' previous experience with the task the application performs gives the application clues towards what kind of experience the user has even though the experience isn't with this application.

For example, on first use, our Blog-Editor asks the user for their web-log site URL, username, and password. With this account information the application accesses the users' web-log through a standard XML-RPC (Winer, 2003) method and analyzes the users' previous entries if any exist. If the application finds image tags in the users previous web-log entries, the imaging interface layer will be added to the interface. Likewise if other items are detected in the web-log entries, like text-styles or text-alignment the application adds those layers to the interface at first use. If no entries are found the user is assumed to be a first-time web-logger and no automatic action is taking to add new layers to their interface.

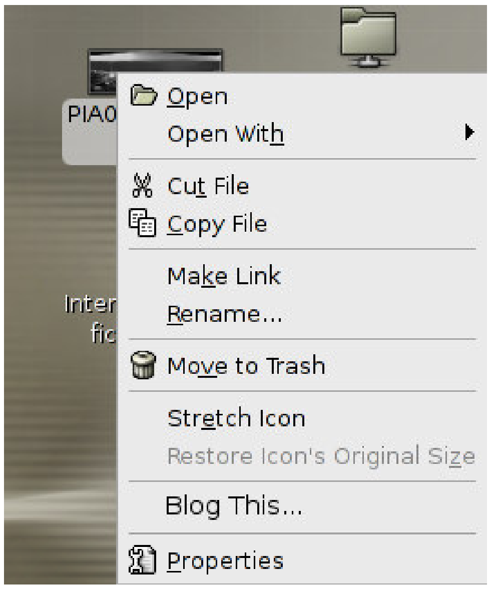


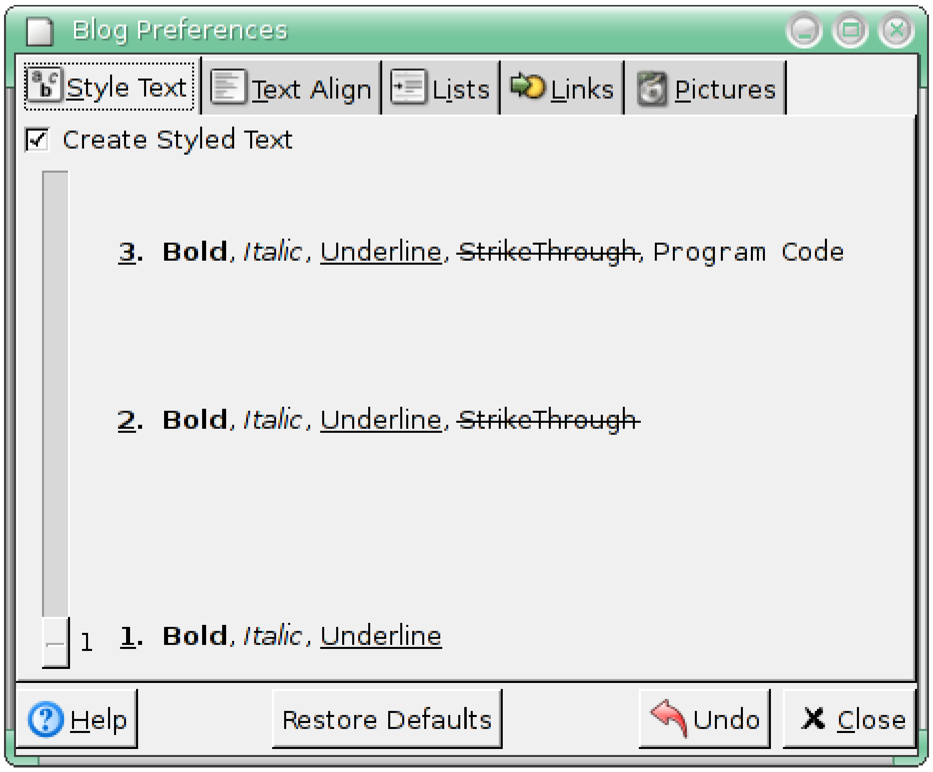
Figure 2 - Alternate Program Startup, right clicking image shows menu item for opening the Blog-Editor with this image

3.3 Adding Layers through Alternative Application Startup

It can be difficult to judge when a user is ready for the addition of a new layer to the interface. We propose taking clues from how the user starts the application. For example, users may click on a document to launch an application. This document can be examined for clues to the features required by the user.

Figure 3 shows an example of the dynamic menu for different image types like jpg, png or gif. The user has right-clicked on an image and the popup menu gives the option to “Blog This…”. The “Blog This…” option can be placed on any type of file the application has registered to handle, our Blog-Editor registered images, source code, and links or text within the web browser. The popup menus give quick access to the Blog-Editor and acts as the launch context provider for the automatic discovery of the extra features the user desires in our application.

When a user who has never used phot-handling features chooses to “Blog This...” on a photo, the application adds the image feature layer at the default basic level; this layer gives the user the ability to edit and add images inside the Blog-Editor.
This technique adds new features only an application launch. This is good because users may complain that the interface appears too dynamic or uncontrolled if features are added during the runtime of the application.



☑ Create Styled Text

3. **Bold**, *Italic*, Underline, ~~StrikeThrough~~, `Program Code`

2. **Bold**, *Italic*, Underline, ~~StrikeThrough~~

1 1. **Bold**, *Italic*, Underline

Figure 3 - Layers and Layer Level Preferences

3.4 Switching Layers through Manual Control

In our Blog-Editor, we also preserved a manual control for adding layers. At anytime, a user may select the "More Features" option and is taken to a preferences window where they can decide if they would like more features shown. Each of the tabs in the preferences window represents a feature layer the user may wish to change. Each tab also has different levels within the Feature Layer, which the user can adjust by moving the slider up to gain more advanced features in that layer or down for less.

Each of the changes made through the Preferences window are instantly applied to the interface the user can see the effect that adding a layer has. The user can experiment with the different layers and levels of layers until they find what they are comfortable with. Experienced users may find this the quickest way to go from the simplest interface to the more advanced interface.

Because layers were chosen in the relative groups of features it has simplified the choice of adding layers in the Preferences dialog. Previous attempts at providing the user with a manual method of changing layers gave a single slider bar with numbers assigned to each layer level. We felt that the single slider approach didn't give enough indication of what features would be present at each layer. Instead, each level of a Feature Layer has a number attached to it but is described by the additional features that come with that level so the user knows what changes will take place with the new level.

4 Conclusions

In this paper we have designed and implemented an example Multi-Layer User Interface to encourage the adoption of this technique. Our implementation details new design concepts of a Multi-Layer User Interface while also demonstrating our methods for deciding when to switch layers.

By defining Feature Layers and Mixed Layers in Multi-Layer Interfaces, we have shown a better way to organize Multi-Layer User Interfaces into a finer grained layer approach. With our Blog-Editor we explored several systems for deciding when to add new layers to the users interface.

Analyzing the users' previous use of a similar application allowed our interface to determine the users' experience even though the experience wasn't with our application. Explicit interactions such as alternative application startup gave our system the ability to add new layers to the interface based on how the user launched the application. While automatic methods of adding layers to the interface can improve adaptability of the application we continue to provide a manual adjustment interface accessible at anytime.

References

Ceaparu, I., Lazar, J., Bessiere, K., Robinson, J., & B. Shneiderman. (2004). Determining causes and severity of end-user frustration. *International Journal of Human-Computer Interaction*, 2004.

Clark, B. (2004). PyBlog. Retrieved May 1, 2004, from http://www.clarkson.edu/~clarkbw/thesis.

Kang, H., Plaisant, C., & Shneiderman, B. (2003). New approaches to help users get started with visual interfaces: Multi-layered interfaces and Integrated Initial Guidance. *Proc. of the Digital Government Research Conference*, Boston, MA, May 2003.

Kraut, R., Scherlis, W., Mukhopadhyay, T., Manning, J., & Kiesler, S. (1996). The HomeNet field trial of residential Internet services. *Communications of the ACM* , 39 (12) 55–63.

Langley, P. & Fehling, M. (1998). The experimental study of adaptive user interfaces. Technical Report 98-3, Institute for the Study of Learning and Expertise, Palo Alto, CA, 1998.

Shneiderman, B. (2003). Promoting universal usability with multi-layer interface design, ACM Conference on Universal Usability, Vancouver, British Columbia, Canada, November 2003.

Wentworth, D. (2003). Definition of a weblog. Retrieved February 1, 2003, from http://blogs.law.harvard.edu/about#whatIsAWeblog.

Winer, D. (2003). RFC: MetaWeblog API, Retrieved Auguest 1, 2003 from http://www.xmlrpc.com/discuss/msgReader$2198?mode=topic.