Dynamic Pull-Down Menu Organization in Run-Time by Analyzing Menu Selection Behaviors

YoungJin Ro, Ho-Jin Choi, Dan Hyung Lee, and In-Young Ko
School of Engineering, Information and Communications University, Daejeon, Korea
{ryj8516, hjchoi, danlee, iko}@icu.ac.kr

Abstract

Event-based self-adaptation of user interface (UI) is introduced in this paper as an approach to solving problems in current static and dynamic UI approaches. Our approach does not require extra information about the user or the context, but uses events which can be gathered easily. Our approach adopts the concept of an agent to make externalization of UI adaptation, that is, no need to make intensive modification of the internal logic of an application. At the current stage of research, this paper focuses only on the menu organization aspect of UI design, where menu categorization and ordering can be accomplished dynamically based on user's menu selection behaviors. The outcome of the approach is to show enhanced job performance by reducing time for menu search and occurrence for false selection. It is expected that many menu-driven systems can improve the usability by adopting our approach.

1. Introduction

Nowadays most software organizations recognize the importance of the user interface (UI) design in making software products. No matter how good 'zero defect' software products they are, it should be regarded as the failure if users do not wish to use them because of bad design. There are several ways to improve the UI of a software product, and evolutionary prototyping has been one of the popular traditional methods to get a relevant UI. However, this approach has limitations due to its "static" nature, and more "dynamic" approach to UI design has been introduced recently to overcome the limitations of the traditional approach.

Dynamic UI represents a new paradigm of UI design, that is, a concept of such UI that can be changed (semi-)automatically while the software is in use based on the analysis of user's pattern of actions. In short, different users may see different UI's adapted to each individual. This paper investigates an approach to dynamic UI based on events. In our approach, events are extracted to recognize user's behaviors and, based on the information with these events, the UI adapts itself to the user dynamically. This research is in its infancy, hence in this paper we focus only on the menu organization aspect of dynamic UI.

The paper is organized as follows. The motivation to and the basic concept for dynamic UI are described in sections 2 and 3, respectively. Our approach is introduced in section 4, where the aim is to provide a guideline towards dynamic menu organization. Section 5 evaluates the potential usefulness of this approach, and section 6 concludes.

2. Motivation

There are many dialog styles such as filling-in-forms, questioning and answering, and command languages. Each dialog style is used differently and each one has its own strengths and weaknesses – for example, menu-driven interfaces are easy to learn, but inefficient and inflexible. In this paper we begin our research towards dynamic UI by focusing on menu organization since menus provide a convenient UI even for novice users. By allowing the menu organization to enhance itself dynamically, this approach can be used to improve a menu-driven system's usability substantially.

When using pull-down menus, menu selection errors can degrade the performance of a job. Menu selection errors are known to be caused by problems such as improper categorization, overlapping categories, and vague category titles. Some research reports that search time was saved by 10% to 37%, and task errors were reduced up to 53% by correcting selection errors in the UI.

Figure 1 shows examples of improper categorization errors of Microsoft Word 2003 and Microsoft Power Point 2003. Three examples are shown in the figure. First, users should select header

and footer in View category instead of Insert category although users select this menu to insert header and footer. Second and third examples are similar with the first example. It is difficult to decide that page setup is categorized as File or Format and Master is categorized as View or Format. Since users cannot easily recognize correct categorization, users would make mistakes and it can degrade performance. However, it is not appropriate to claim that improper categorization errors are made by every user because some people agree with the original categorization.

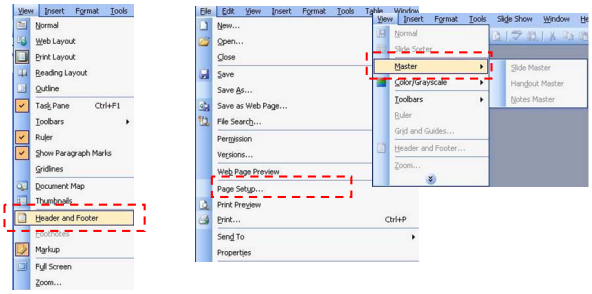

Figure 1. Examples of improper categorization errors

Categorization is not a simple task for a UI designer as he/she should provide a reliable menu categorization for many different users. One popular approach to categorization of menus is Hayhoe's approach. In this approach, at least 50 users make grouping items into self-defined categories. Then the frequency with which users place pairs of items together in a group is calculated, resulting in similarity measures. However, this approach has weaknesses. First, too many users are needed. Hayhoe argued that at least 50 subjects are needed. Research with 50 subjects is not an easy work for open software developers. Second, it does not guarantee to produce optimal menu categories. In fact, it is not possible to make categorization that all people satisfy. For these reasons, it seems difficult to reach a good solution for static categorization of menus, hence we aim to seek solutions using dynamic UI approach.

3. Dynamic UI

Dynamic UI is a new paradigm. This approach was introduced to overcome limitations of static UI, but still dynamic UI is not used for many applications. In dynamic UI, system tries to predict user's immediate actions and change the dynamic part of the UI frequently in the anticipation of the next action. Also, it is expected that less effort will be required to communicate with users compared with static UI since system automatically changes UI based on user's acting patterns. Users can get specialized UI without continuous complaining to designers about the uncomfortable UI if system automatically reflects user's patterns of actions. For example, system can find why a user has some difficulty to accomplish a specific task and fix it until the user does not show any problem. Therefore, it is able to make user centered design without high cost as like static UI.

The approach of dynamic adaptation based on user information takes information from users then tailor UI by this information. An example of this approach is dynamic interaction generation (DIG), which was introduced to make dynamic user interaction. The DIG research prototype, called DIGBE (DIG for building environments), was also implemented. DIGBE converts a building management configuration database into a dynamic, adaptive user interface. The framework takes the user privileges, the roles of the interaction participants, the specific task, the objects of interest, and the values and types of data for making dynamic UI adaptation.

While Penner and Steinmetz [2] focused on the adaptation based on user information such as operator role or task model, Menkhaus and Pree [3] concerned about multi platform access and contexts. Due to appearance of new devices, which are able to access Web applications, such as mobile phones and handheld computers, making interactive services becomes more difficult. As an example for dynamic adaptation based on contexts, MUSA (multiple user interfaces, single application) prototype was introduced to support multi platform and tailor UI properly. First, it can be easier to develop UI for a variety of computing devices by making an abstract description of a UI. Also, MUSA tries to adapt to the context in which it is used. Tailoring of the interface design includes the capability of adaptation of content delivery to various devices, while preserving consistency and usability of the service.

Although the approaches above seem convincing in special circumstances, they have weaknesses to be used widely. First, they usually support only pre-defined user types or contexts, limited for supporting various and anonymous users in the Web environment. Second, adaptation cannot be made flexibly because these approaches provide multiple UI for single application,. Third, they have weaknesses in adapting to existing applications since UI should be made from the start. Finally, they require user's intervention to provide specific information such as the user's role and access level to make dynamic UI adaptation.

4. Event-based dynamic pull-down menu organization

4.1. Event-based self-adaptation

Our approach uses an agent. An agent perceives its environment using sensors and acts on the environment using actuators. Figure 2 shows the schematic diagram of a simple reflex agent which is used in our approach.

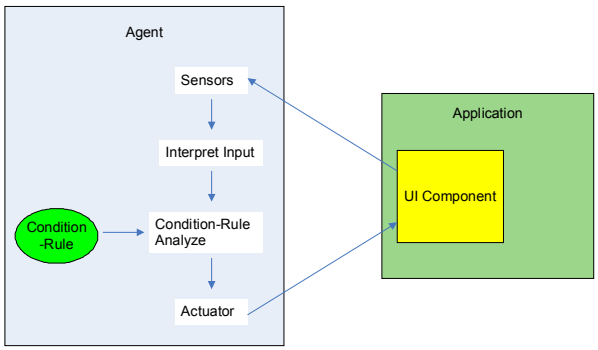

Figure 2. Schematic diagram of the system

The concept of agent is useful for self-adaptation of UI because the structure is appropriate for monitoring UI events and acting to change UI element properties externally. An agent is placed in the outer side of the application, then it will be possible to adapt in any application since the internal logic of applications does not depend on the agent. By externalization of UI evolution, there will be no need to make application-specific agent, and adaptation of existing applications will require less cost. Therefore, the agent approach can be a solution to overcome weaknesses of current dynamic UI approaches.

Using UI events for self-adaptation has strong points. First, in most applications, UI events can be extracted easily. There are several applications that log events (e.g., Web applications) and UI events can be gathered without additional equipments. Moreover, gathering UI events does not depend on user types or contexts. Especially, menu selection events can be gathered easily when the system has hierarchical menus.

4.2. Preparation of designers

First, designers make basic menu design. For menu organization, for example, designers should allocate area for pull-down menu and menu bar using common sense. At least, designers can make basic menu design with random order or alphabetic order.

Second, designers set up conditions and rules for dynamic adapt ion of UI. In case of menu organization, designers can decide such parameters following:

- When does a sub-menu need to be moved for modifying categorization?
- When does a sub-menu hide?
- Which is more important between frequency of use and order of use?

4.3. Sensing events

The event-based self-adaptation approach allows us to collect usability information by analyzing UI events. Although the agent only perceives low-level events such as shifts in input focus, key or mouse events, the events can be abstracted and filtered. By analyzing each event, it is possible to make counts and statistics to provide measurable data to the system. Later the counts and statistics can be used for usability improvement. For example, if Nielsen's model is used, which provides usability factors such as learnability, efficiency, memorability, errors, and satisfaction, the measurement of usability factors can be accomplished like table 1. Although satisfaction is one of the most important factors of usability, there is no way to record satisfaction automatically without any user intervention. Therefore, without user intervention to give scoring about satisfaction, system cannot automatically analyze satisfaction information from events. The table shows an example to gather usability information from counts and statistics, but by characteristics of systems, the way of gathering usability should be tailored.

Table 1. Gathering usability from counts and statistics

Learnability	$\frac{\text{Current Total Time}}{\text{Past Total Time}}$ $\frac{\text{Current Time / Task}}{\text{Past Time / Task}}$ $\frac{\text{Current Time / Task}}{\text{Past Time / Task}}$
Efficiency	Total Time Time / Task
Memorability	$\frac{\text{Current Idle Time}}{\text{Past Idle Time}}$ $\frac{\text{Current Time for Filtered Events}}{\text{Past Time for Filtered Events}}$
Errors	Number of Errors

These steps of events extraction and interpretation help gathering usability information for the system to handle the data automatically as above. When a system has complex functionalities and UI, however, it is not simple to extract events correctly. Accordingly, it is not easy to determine how lower level events are

abstracted and how to decide filtered events since the system cannot know user's objectives.

While this paper focuses on menus, the challenge becomes much easier. It is clear to recognize user's objectives since selecting appropriate menus are the final goals of users always. For example, users do not select font menu for printing, inserting a picture, or drawing table. Users only click font menu to adjust font format. Also, it is easy to determine filtered events since menu selection events are simple enough. Assume that font is categorized as format. Then, if a user select help category first, and select format to select font later, selecting help category can be classified as a filtered event. By recording what menus are selected finally, system can recognize filtered events.

In this paper, mainly three different data can be extracted by the system. In addition, first two counts and statistics can be classified as menu selection events. First, the framework accepts selection events such as moving mouse pointer to specific menu or category. Also, when a user clicks a menu or category, then it would be recorded as a selection event. Second, search time can be used meaningfully. Search time means spent time for selecting a menu. These events can be sensed when any menu search event is made. Finally, the last data is frequency of use of menu. It concerns how often a user uses the menu. It could be calculated by the delay between two identical menu search events. For example, if user A attempts to select menus in every 1 minute while user B attempts to select menus in every 30 seconds, A can be evaluated with low frequency of use. The usage of different counts and statistics will be explained in next sections.

4.4. Dynamic menu categorization

The menu categorization can affect the performance. However, since the paper deals a menu which can be classified as 'Sets of Lists', next section also describes about the dynamic menu ordering; sets and lists are that sets do not much concern about the order, but in a list, specific order is often meaningful, so for lists, dynamic menu ordering is necessary.

The basic approach for the dynamic categorization is as follows. If a user selects a category more frequently and spent more time for the category than other categories to select a menu, then system modifies menu category. For the dynamic menu categorization, the system should keep data such as category selection information, spent time, number of categories. As an example, a scoring for each category can be following:

*C[i] * Time[i] / NumMenu[j]*
C: Category selection
Time: Spent time
NumMenu: Number of menus

After the system makes scoring for every category as above, if one score exceeds other categories' scores at least as much as defined threshold, then the menu categorization should be modified.

This initial approach has a problem. That is, order of selection path does not affect to categorization. For example, among selected categories, first selected category should be treated with bigger score than later selected categories. Therefore, the scoring should be modified as following:

- Original score = C[i] * Time[i] / NumMenu[j]
- Adjusted score = Original score * (TotalPath – PathOrder[i] + 1 / TotalPath)

By this adjustment, each category can have different weight although several categories are selected. But there remain more problems. First, the approach assumes that user makes blind moves to find a menu. Second, the approach ignores learning ability of user. After user learning menus and categories, changing categorization can make confusing.

In fact, users do not make blind move when they cannot find appropriate menus. Users find next category by seeing title of categories. Then, dynamic categorization should consider about influence of each title. That is, certain weight should be multiplied with scoring. Here, designer's common sense can be used to establish influence of each title. It is not recommended however because dynamic menu organizations focuses on the user-oriented menu design. Instead of designer's common sense, user's grouping research information can be used for setting up the weights since personalized grouping is often inferior no matter who makes the grouping. Table 2 represents an example of weighting for each category.

Table 2. Example of weighting

	File	Format	Tools
No. of users select to make grouping	50	30	20
Weight	50 / 100 = 0.5	30 / 100 = 0.3	20 / 100 = 0.2

The second problem was ignoring user learning. In fact, users can learn where a menu is located after several trials and errors. Users can also disable adjusting categorization. If a user disables dynamic menu categorization, there will be no confusing anymore due to unwanted changing. When users

disable dynamic categorization, all menus cannot be categorized anymore. It could be a problem if users can want to configure only few menus. Therefore, the approach should consider frequency of use of menu that can be gathered automatically. The system records previous selection and current selection of menu and when frequency of use of a certain menu is high, then it can have higher threshold to adjust categorization. Especially, clear information about categorization of menus can be remembered about 30 seconds since human short term memory lasts less than 30 seconds. Therefore, if a menu is selected in every few seconds, it is not effective to modify categorization.

4.5. Dynamic menu ordering

Dynamic menu ordering is simpler than dynamic menu categorization. It uses relative scoring for order of use and frequency of use. Remaining challenge is how to make a combination of order of use and frequency of use. For considering two ordering ways together, scores are adjusted for fair comparison. In this paper, all values should be adjusted to be located in the range from 0 to 1. For example, 3 can be adjusted to 0.875 since the original range of scoring is from -4 to 4. Figure 3 shows the example of adjusting scores. In this figure, left table is used for scoring order of use and right table is used for scoring frequency of use. When the order of use is counted, a positive number means column menu is executed former than row menu. Also In the scoring order of use, a positive number means column menu is executed more time than row menu while a negative number indicates column menu is executed less than row menu.

	Menu1	Menu2	Menu3
Menu1		+3	+4
Menu2	-3		0
Menu3	-4	0	

	Menu1	Menu2	Menu3
Menu1		+1	0
Menu2	-1		0
Menu3	0	0	

	Menu1	Menu2	Menu3
Menu1		0.875	1.000
Menu2	0.125		0.500
Menu3	0.500	0.500	

	Menu1	Menu2	Menu3
Menu1		1.000	0.500
Menu2	0.000		0.500
Menu3	0.500	0.500	

Figure 3. Example of adjusting scores

Finally, the order of use and frequency of use can be combined with two weights: p, and q. p indicates order of use weight that a user or designer gives, and q indicates frequency of use weight. The default values of p and q can be 0.5 and 0.5.

5. Evaluation

For evaluation of the proposed approach, 5 different subjects were chosen. Then, a set of 25 menus and 5 different categories were established. Each subject could make grouping with the menus and categories, then a final categorization was made based on summarizing 5 different categories of each subject. Then, the hit ratio of each user's categorization compared with generated menu categorization was calculated as figure 4.

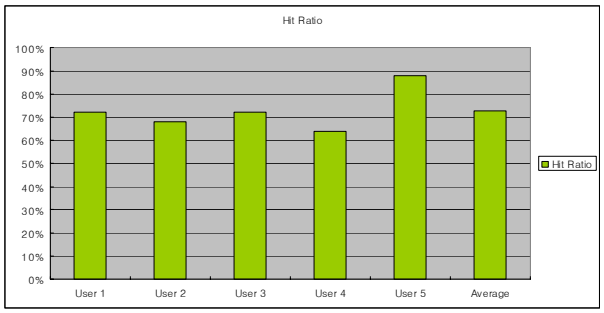

Figure 4. Hit ratio of each user

In this figure, the average hit ratio is about 73%. When a user tries to select a menu, the user has 27% probability of miss selection. In this case, the frequent miss selection can bring huge performance degrading. The hit ratio of categorization will be much higher in general cases, since the menus which are used for this experiment were examples of ambiguous menus such as page setup. Since the objective of the evaluation could be confirming the performance improvement when users make selection errors, it is more effective to use ambiguous menus to see clear performance improvement.

Next, 25, 50, and 100 random menu selections are generated. If there is an expert who knows exact locations of all menus, then the expert can make only 50 selection events for accomplishing 25 menu selections because it requires 25 category selections and 25 menu selections since sub-menus cannot be selected without selecting specific categories. In the best case, only 100 and 200 menu selection events are needed for executing 50 and 100 random menus. When a user makes blind moves, the user cannot find the exact category that menu is contained. In the worst case, the user can select a menu after select 5 different categories. Therefore, the number of selection events was much bigger than the best case. Even the gap between the best case and the result became bigger when more random menu selections were made. Although the experiment deals the title-based move, there is a gap between the best case and the result. By adopting dynamic menu categorization, the number of false selection events is reduced dynamically. The

performance of searching menu is increased. Figure 5 and 6 show the average number of selection events with blind moves and title-based moves in both of static and dynamic menus.

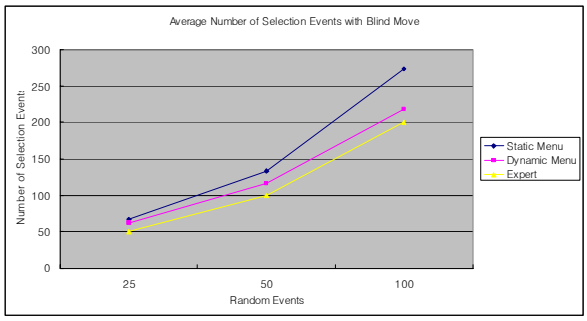

Figure 5. Average number of selection events with blind moves

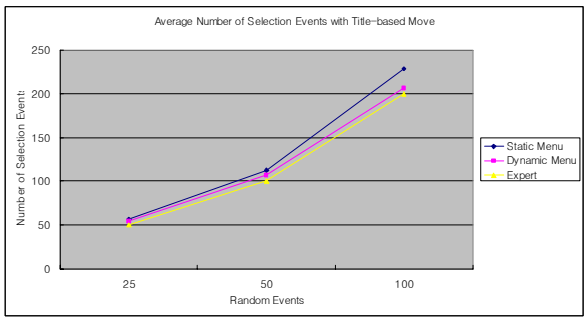

Figure 6. Average number of selection events with title-based moves

6. Conclusion

This paper proposed an event-based self-adaptation of UI as an approach to solving problems in existing dynamic UI approaches. At the current stage of research, our work focuses only on the menu organization aspect of UI design, where menu categorization and ordering can be accomplished dynamically based on user's menu selection behaviors. Our approach does not require extra information about the user or the context, but uses events which can be gathered easily. Our approach adopts the concept of an agent to make externalization of UI adaptation, that is, no need to make intensive modification of the internal logic of an application.

To show the validity of the proposed approach, we conducted a preliminary experiment for enhanced job performance by reducing time for menu search and occurrence for false selection. It is expected that many menu-driven systems can improve the usability by adopting our approach.

7. Acknowledgement

This research was supported by the MKE(Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Advancement). (IITA-2008-C1090-0801-0032)

8. References

[1] K. Gajos, R. Hoffmann, and D.S. Weld, "Improving User Interface Personalization", *Proceedings of UIST*, Vol. 4.

[2] R.R. Penner and E.S. Steinmetz, "Dynamic User Interface Adaptation Based on Operator Role and Task Models", *IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 2, 2000.

[3] G. Menkhaus and W. Pree, "User Interface Tailoring for Multi-platform Service Access", *Proceedings of the 7th international conference on intelligent user interfaces*, ACM Press, New York, USA, 2002, pp. 208-209.

[4] E.S. Lee and D.R. Raymond, "Menu-Driven Systems", *Encyclopedia of Microcomputers*, Vol. 11, 1993, pp. 101-127.

[5] D. Bäumer, W.R. Bischofberger, H. Lichter, and H. Züllighoven, "User Interface Prototyping—Concepts, Tools, and Experience", *Proceedings of the 18th international conference on software engineering*, IEEE Computer Society Washington DC, USA, 1996, pp. 532-541.

[6] D.J. Mayhew, *Principles and Guidelines in Software User Interface Design*, Prentice-Hall, NJ, USA, 1991.

[7] S.J. Russell and P. Norvig, *Artificial Intelligence: a Modern Approach*, Prentice-Hall, NJ, USA, 1995.