ISSUES IN PRACTICAL APPLICATION OF AN ADAPTIVE INTERFACE

Beth Meyer, K. C. Burgess Yakemovic, Michael Harris

NCR Human Interface Technology Center
500 Tech Parkway, NW
Atlanta, GA 30313
(404) 853-2959
Beth.Meyer@AtlantaGA.ncr.com

ABSTRACT

The authors have been developing a prototype system for installation in an operational business environment. During the development, a number of issues have been encountered. These include:

- constraints arising from placing prototypes in operational environments
- lack of guidelines for selecting types of adaptation
- difficulty determining adaptation criteria
- difficulty obtaining necessary data from users
- lack of guidelines for making information display decisions
- problems in testing 'real world' interfaces

These issues are not readily solved with more sophisticated adaptation algorithms; rather, they point out the need for collecting more information from attempts to bring intelligent interfaces to complex business environments.

KEYWORDS: Adaptive interfaces, prototypes, commercial applications, development issues

INTRODUCTION

Since early 1991, the authors have been developing a prototype of an adaptive assistance system for retail point-of-sale operation. This prototype will be tested in an operational business environment; it is currently undergoing laboratory usability testing. While developing an adaptive interface to be installed in a business environment, we have encountered many practical issues which appear to apply to a broad class of potential applications for adaptive interfaces.

CONSTRAINTS OF OPERATIONAL PROTOTYPING

One of the most important processes in designing and evaluating a new interface is testing a prototype with real users performing real tasks. The necessity of doing this for a business-related interface can constrain prototype development. Two of the important constraint issues are negative transfer and completeness.

Preventing Negative Transfer

If an early prototype will be tested temporarily in an operational situation, people who are being paid for doing real work (rather than for performing tests) cannot afford to be slowed down either when the prototype is installed or when it is removed. This is especially true for highly procedural domains, where many actions and motions may become automatic with practice.

One way to address this problem is to specify that all existing procedures still be used to accomplish the user's goals. The prototype (new system) may differ from the existing system in the information that it provides to the user, but not in the way it requires users to accomplish their existing goals. In such a prototype, it may be reasonable to add new actions for accomplishing new goals, such as deciding whether an adaptation should take place. These actions must be ones that have no harmful effects in the existing system, so that users returning to the existing system are not affected.

This requirement restricts the ways in which the interface can adapt. For example, adapting the communication style (perhaps giving novices a menu navigation and experts a command-based system) may require the user to either learn a new way of performing tasks at the beginning of the prototype trial, unlearn the new methods at the end of the trial, or both. Similarly, dynamic task allocation could cause problems upon the removal of the prototype. The tasks the system would take upon itself during high workload conditions would be more difficult to do under the same conditions without the system.

Completeness of Prototype Implementations

When developing a prototype, one can often finish the job within cost and time constraints by limiting the range of tasks that the prototype supports. Originally, assisting retail sales seemed to be a small problem; task analysis results showed this assumption to be false. Hence, assisting every possible task with the prototype was not feasible.

However, installation in an operational environment requires either complete coverage of all tasks or an invisible transition between places where the prototype implementation is complete and places where it is not. In our case, an acceptable transition was possible. Had it not been possible, our prototype implementation would have had to be expanded to support the complete interface. Of the possible adaptive architectures, few have been tested on complete problems that are of a size similar to that of the retail point-of-sale problem.

SPECIFYING THE TYPE OF ADAPTATION

One of the fundamental steps in designing an adaptive interface is determining what aspects of the system will change in response to changing conditions. The following are some of the ways in which the system may adapt [1,2]:

- Task allocation or partitioning -- the system itself performs the complete task or part of it.
- Interface transformation -- the system adapts to make the task easier by changing the communication style and the content and form of displayed information.
- Functionality -- the system adapts the functions available to each user.
- User -- the system can help the user to adapt by determining apparent problem areas and providing intelligent tutoring for them.

Because of the constraints mentioned previously, the primary means of adaptation used by our prototype was that of changing the information displayed. If all types are equally feasible, it is not clear which types of adaptation are the most beneficial in which cases. This would be useful information for developers, since adaptation of all types at once could cause enough simultaneous change to confuse the user.

SPECIFYING THE CRITERIA FOR ADAPTATION

Another step in developing an adaptive interface is determining the conditions that should cause the system to adapt. For example, the system might adapt to any of the following characteristics of the user, task, or environment [1]:

- User experience with the task itself
- Previous user experience (e.g., familiarity with GUI or command-line interfaces)
- User aptitudes (e.g., visual acuity or spatial reasoning ability)
- User preferences
- User demographics (e.g., job title or education)
- Task complexity
- Task frequency
- Probable workload
- Physical conditions (e.g., ambient noise level)

For each criterion used, there must be at least one source of data for the system. In the absence of a tool for quickly making radical changes to adaptation algorithms and data stores, selecting the most important conditions for adaptation and the most accurate and tractable measurements of them is critical. Since a variable may be missed in the initial analysis, or not considered useful initially but shown to be important in later testing, it is necessary to have architectures which support extension.

OBTAINING DATA TO DRIVE ADAPTATION

Even if the most important and useful theoretical bases for adaptation are selected, obtaining the actual data to measure these variables is extremely challenging in an real business application.

Some of the easiest data to collect is stable user information, such as job title and education. This data only requires some provision for offline initialization and maintenance. (It would be unacceptable to require users to provide a complete background to the system when they need to use it to perform a transaction.) However, this data by itself cannot support appropriate adaptation to the changing needs of a single user, and the more useful data is also more difficult to collect.

Workload Data

Suppose that the system is designed to provide more assistance during times of greater workload. In a very general way, workload can be estimated using such factors as time of day, season, and current staffing. However, the primary factors are the number and difficulty of tasks that the user is performing simultaneously. In the retail environment, secondary tasks that increase workload may include bagging merchandise, conversing with customers, or using the telephone -- activities not readily detected by the system.

Speed Data

One of the primary measures of expertise used in human performance studies is speed. An adaptive system could measure how quickly a user accomplishes tasks and subtasks, and then adjust the level of assistance to conform with an expertise level commensurate with that speed. However, in the retail environment, operator speed is affected by many unrelated factors, such as:

- Amount and type of merchandise in a transaction
- Customer actions (talking to operator)
- System speed and hardware conditions (an operator will necessarily be much slower if the scanner is not working correctly)

Both speed and workload data, then, are significantly affected by factors external to the system. This type of issue affects any adaptive interface designed for an environment in which other people or unrelated systems may interact with the user.

Accuracy Data

Accuracy may be a more useful way of measuring user expertise and evaluating user actions, particularly since the intelligence of an adaptive interface may be more effective in preventing common errors than in speeding up correct performance. To determine whether the user's actions are correct, the system must know what the user is trying to do, how best to achieve this goal, and whether this goal is even appropriate in the first place. Obtaining this information proved to be one of the most fundamental and intractable challenges in developing our adaptive interface.

For example, one of the more common errors observed with our point-of-sale system was the omission of a special step that was necessary for sales made to store employees. Unfortunately, that step was the <u>only</u> action that distinguished an employee sale from any other sale. Hence, if the user forgot the step, the system would not be able to determine that the sale was to an employee and would not be able to detect the error.

As we analyzed the task of operating the point-of-sale terminal, we were surprised to find that many actions could be correct or incorrect, depending on conditions that the system would not be able to detect. Not even an aborted transaction is always the sign of an error -- an operator may have to abort a correct transaction because the customer had a change of heart.

When we built and tested the adaptive interface, we found that the incompleteness of the available data made the system's goal inferences a sometimes shaky basis for aiding and adaptation. During usability testing, a user might wish to perform a cash return but accidentally enter the code for a cash sale. We would then watch as the adaptive interface dutifully assisted the user in performing a cash sale, never able to determine that this was not what the user actually wanted to do.

The problem of incorrect goal inference is hardly limited to the retail environment; it is a risk in any task that does not require the user to make a complete and absolutely correct goal statement at the outset. (Few tasks, if any, have such a requirement.) It seems that a fertile area for research and development would be means for users to easily communicate their goals directly to the system at any time, in much the same way they would to a human colleague. Without such certain knowledge of the user's goal, there is always the risk of assuming correct performance when an error has been made, or vice-versa.

IDENTIFYING THE USER

There is another fundamental issue related to collecting user data. For the system to adapt based on what it already knows about the current user, it must first know who the current user is. This is less trivial than it might appear. In retail environments, identification numbers are routinely supplied. However, users sometimes enter identification numbers other than their own when performing certain tasks. (This has to do with calculating commissions.) In other environments, there may not be a specific form of identification, or the identification supplied may apply to more than one user. For example, an ATM may not be able to distinguish between the users of a joint bank account.

In other instances, the identification may be entered late in a task, so that the adaptive interface cannot use its knowledge of the user for much of the task. One answer to that problem is to change the task so that users always identify themselves at the very beginning of each session. However, in practical applications, the size and nature of the installed base of users may effectively prevent a change like this.

SPECIFYING INFORMATION PROVIDED TO THE USER

For a system that adapts primarily in terms of information displayed, the designer must determine how the users' information needs change as they gain experience and as other factors change. This information comes from a thorough and detailed task analysis.

Many task analysis methods identify all individual steps and their relationships within complete tasks, often using graphical representations such as flow charts. Such a detailed level of analysis can ensure that the interface is responsive to users' individual actions. However, this microscopic view may obscure the importance of the overall goal of the user, particularly when many steps are common to different overall tasks. System designers may miss opportunities to guide the user toward the overall goal. Therefore, it is important to reexamine the detailed analysis by reviewing several complete tasks from beginning to end.

At the same time, it is useful that adaptation be possible at the level of individual steps. We found that adapting only at the task level reduced the effectiveness of aiding. If a user made a mistake in a single step of a task, the system would provide extra assistance on *all* steps of the task, most of it unnecessary.

Some of the reasons for users' information needs to change may not be readily detected by the system. For example, the adaptive system might present a brand-new user with a motion video clip showing how to perform a group of steps in a transaction. After the novice performs the steps correctly a few times, the system may eliminate the video image and simply prompt the user to complete the steps. Several transactions later, the user might need more information about one of these steps. The system then must be able to distinguish between a need for more information about the complete group of steps and a need for different information about an individual step.

EVALUATING THE ADAPTIVE INTERFACE

In order to test whether an adaptive interface is truly adding to the usability of a product, the test must replicate the conditions under which the interface is designed to adapt. In other words, if the interface is designed to provide maximum assistance to novices and then provide less help as the user gains experience, the test subjects should actually move beyond the novice level in the course of the test. The more complex the system, the harder it is to encompass this level of training and practice in a laboratory test.

Other conditions of adaptation are also difficult to simulate in the laboratory. For example, an adaptive system might be designed to provide more help to experts when they have madé a certain number of errors with a single type of task. Testing of existing experts is usually feasible, but ensuring that they have trouble with something may be another matter entirely.

For these reasons, testing in an operational environment over an extended period of time may be necessary to prove the value of an adaptive interface.

SUMMARY

Based on our experience with building an adaptive interface for installation in a business environment, it appears there are a number of issues that are not completely addressed in the literature. These issues are not readily solved with more sophisticated adaptation algorithms; rather, they point out the need for collecting more information by bringing intelligent interfaces to complex real-world environments.

ACKNOWLEDGMENTS

Our thanks to Haila Darcy, Paula Mossaides, Clarkson Jones, Dick Henneman, Mike Donnellan, and Ed Anderson for their assistance with this project.

REFERENCES

1. Malinowski, U., Kühme, T., Dieterich, H., and Schneider-Hufschmidt, M. (In press.) A taxonomy of adaptive user interfaces. In Proceedings of HCI '92, York, 15-18 Sep. 1992, Cambridge University Press.

2. Rouse, W.B. Adaptive aiding for human/computer control. Human Factors, 30, 4 (August 1988), 431-443.