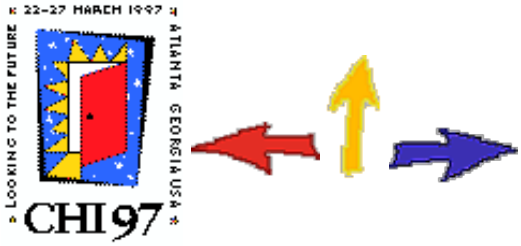
CHI 97 Electronic Publications: Late-Breaking/Interactive Posters



Participatory Adaptation*

Elizabeth Sklar Rozier, Richard Alterman
Computer Science Department
Brandeis University
Waltham MA 02254 USA
+1 617 736 2700
rozierb@cs.brandeis.edu, alterman@cs.brandeis.edu

ABSTRACT

Expert users of programs that handle complicated data management problems develop methods for coping with data overload, multi-user cooperation, and real-time situations. These expert methods incorporate domain and/or user interface knowledge. If such methods were inherent in a system, then novice users could benefit from the expert's experience, the learning curve would be shortened and a more effective system would result. Defining and implementing a complete set of expert methods at design time is a daunting task. Collecting such information from a system's usage, after it has been deployed, should provide a more accurate database of expert methodologies. Current adaptive systems attempt to capture and automate such features during run-time. However, these systems can never evolve very far beyond their original design, since the adaptations occur within the scope of that design. Our method is to offer the expert's usage database as input to the designer, re-introducing the designer in the development cycle after a system has been deployed initially, so that a more effective system can be produced in the next generation.

Keywords

Usage, expert, adaptive system,design

INTRODUCTION

Human interaction with a computer program involves knowledge of two types: domain knowledge and user interface knowledge. Domain knowledge is important for developing problem solving techniques. User interface knowledge is important for working with a computer tool effectively. Expert users of particular computer programs combine the two types of knowledge to perform tasks successfully.

The system design process can receive input from the user population before and after system deployment. For example, participatory design [3] includes the work practices of the targeted user population as a part of the design process. Alternatively, other systems respond to the practice of users at run-time, i.e. during deployment, either by adapting automatically to the habits of the user (adaptive systems) or directly by the user (adaptable systems) [2].

Our method of system development is to gather input for system re-design both during and after deployment. We want to take advantage of the notion that a computer tool can record a database of its usage which can then be used as input to designers in the re-design phase. This data is consulted during re-design to provide guidance for system modification. We refer to this technique as *Participatory Adaptation*. During run-time, we collect the domain and user-interface usage data of experts. It is often hard to classify usage data as either domain- or user-interface-specific, for example when a domain event triggers a user-interface action. Allowing the two types of data to intermix has the potential to be a powerful technique of system adaptation, cutting across both the sign and

tool functions of the system. **

Typical HCI systems address the sign function -- the user interface -- while traditional AI systems examine the tool function -- the user-domain relationship. One of the underlying propositions of our work is to build systems that jointly adapt both the sign and tool function of a system.

SYSTEM DESIGN / TESTBED

Our test domain is called *Mover's World*; three users play different roles in a moving company (two lifters and one handtruck operator) and attempt to move objects from a house onto a truck. Constraints are built into each situation, so that cooperation is required for the completion of a problem. For example, a lifter is only allowed to lift objects with weight less than or equal to his strength, thus heavy objects require two lifters to lift together.

The user interface for our system, called Interactive Mover's World (**imw**), is shown in Figure 1.

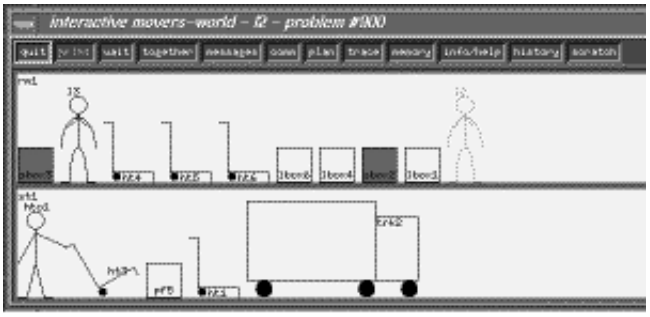


Figure 1

The graphics area illustrates the problem state. The row of buttons above the graphics area activate a series of pop-up windows that the user can manage in order to access various features of the interface. Experts exhibit patterns of usage through popping up and moving around windows, triggered by domain and interface events.

The manipulation of the interface is purposefully complex. Users are intentionally bombarded with sources of information and choices to make. The idea is to simulate an environment that is rich in data, both stagnant and changing in real-time, and to require cooperation between users in order to solve a problem.

The user is required to manage the pop-up windows in order to plan moves, send plans to an executive process that synchronizes actions among users, communicate with other users, examine a database of object properties and access a help facility. All user communications occur via **imw**'s communications facility, which provides a set of canned messages pertinent to the various situations a user might encounter.

A second user interface is being developed for comparison, as shown in Figure 2. The main difference between the two interfaces is evident in the graphics area: the first interface is more intuitive, using squares and stick figures to represent objects and movers. The second interface is entirely symbolic and requires the user to learn a mapping between colored symbols, objects and movers.

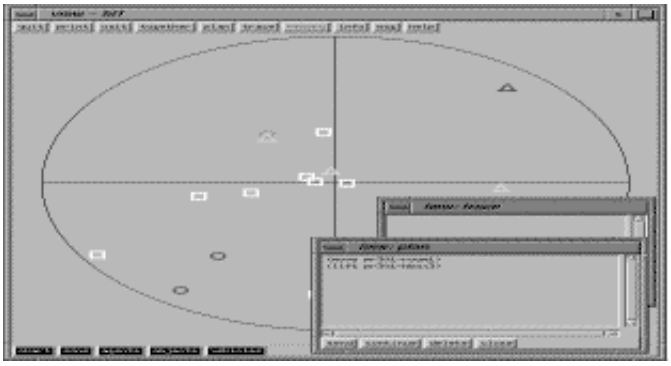


Figure 2

DATA COLLECTION

We are recording all actions that the user makes while running **imw**. This includes key strokes, mouse clicks, domain commands and inter-user communications. The data is examined for recognizable patterns. Note that this data can include domain and interface action chunks that are created by the user during run-time. While solving a problem, users have access to a trace of their actions which they can manipulate to create chunks that simplify their work. As input to the re-design process, these chunks provide valuable information as to how the user tailored the system to his task domain.

As a first step in fitting the usage data, we are currently implementing several techniques: the COBWEB incremental classifier [1] is being used as a basis for

comparison; the MULTICASE [5] is a technique that may prove effective with our data. The MULTICASE is a method for merging multiple episodes of routine activity into a single representation. Our plan is to use the MULTICASE to represent the base of expert activity, which can then be used to recognize subsequent user behaviors, for both sign and tool functions.

PRELIMINARY TESTING

In November 1996 we ran a series of preliminary tests. The resulting run-time data is illustrated in Figure 3. The data revealed that the rates at which users cooperated and communicated with each other did not correlate to the rate of success for a given problem. This result is counter to the intuitive notion that more cooperation/communication should lead to higher success rates. This highlighted the need to re-design the communications facility in the next release of the system.

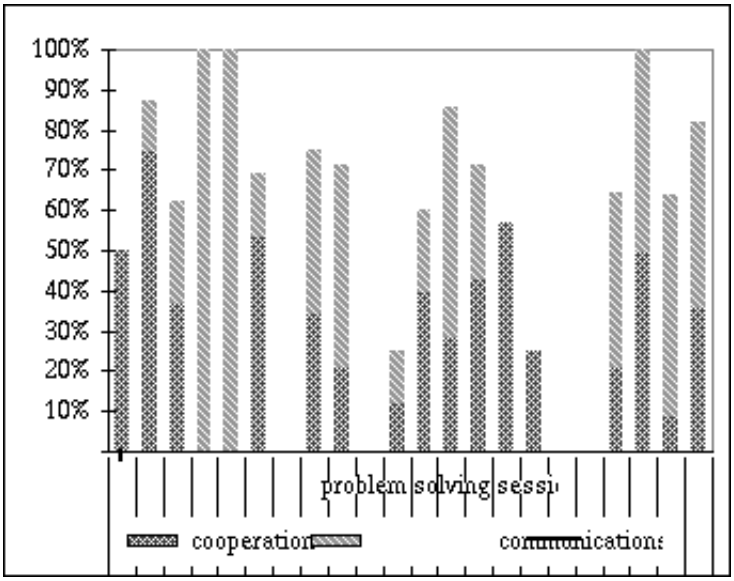


Figure 3

CONCLUDING REMARKS

The system still needs to be further complicated by adding real-time data input that will constrain task completion and by allowing asynchronous activity that will introduce competition between teams of users. Supplying these complicating factors is necessary for simulating an environment where a definite distinction can be made between expert and novice users.

Our long-term goal is to be able to recognize and exploit the methods of experts by capturing and analyzing usage data generated during the run-time of a complicated problem solving system. Our plan is to use this run-time data as a basis for adapting the system so as to improve the performance of novice users. We plan to test this hypothesis experimentally by comparing the efforts of two sets of novice users: those who work with the adapted system and those who work with the original (unadapted) one. Our belief is that the re-designed system, adapted through the participation of expert users, will serve to enhance novice performance.

REFERENCES

1. Fisher, D. H. 1987. Knowledge acquisition via incremental conceptual clustering. *Machine Learning 2*.

2. Oppermann, R., ed. 1994. *Adaptive User Support*. Lawrence Erlbaum Assoc.

3. Schuler, D. and A. Namioka, ed. 1993. *Participatory Design: Principles and Practices*. Lawrence Erlbaum Assoc..

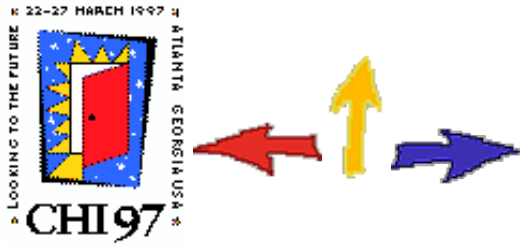
4. Vygotsky, L. S. 1978. *Mind in Society*. Harvard Univ. Press.

5. Zito-Wolf, R., and Alterman, R. 1992. Multicases: A case-based representation for procedural knowledge. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*.

---

* This work was supported by ONR (N00014-96-1-0440).

** It was Vygotsky [4] who noted that any mediating artifact has two functions: sign and tool. In the case of computer systems, its sign function provides the basis of communication with the user, and the tool function, a set of methods for manipulating the data.



CHI 97 Electronic Publications: Late-Breaking/Interactive Posters