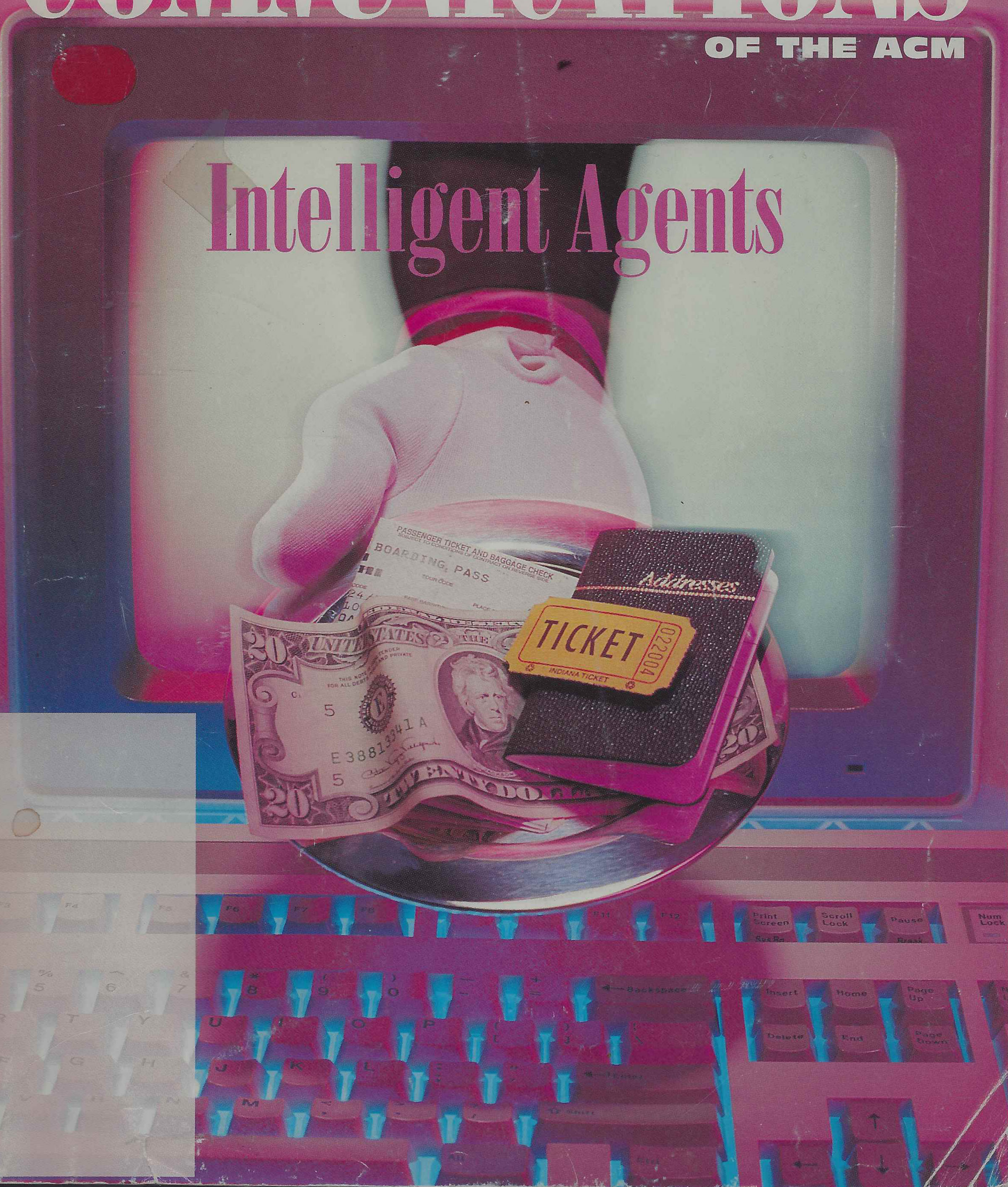
July 1994–Volume 37, Number 7

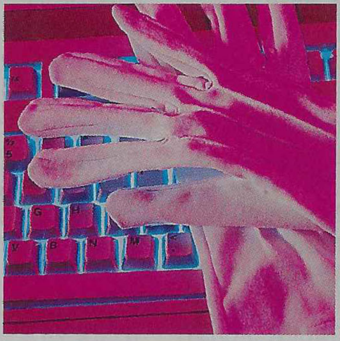
COMMUNICATIONS OF THE ACM

Intelligent Agents



Coach: *A Teaching Agent that Learns*

Ted Selker



COgnitive Adaptive Computer Help (COACH) is a system that records user experience to create personalized user help. Imagine you are learning a new operating system or programming language. COACH watches your actions to build an adaptive user model (AUM) that selects appropriate advice. Just as a football coach will stand on the sidelines and encourage, cajole or reprimand, so COACH is an advisory system that does not interfere with the user's actions but comments opportunistically to help the user along.

COACH might choose to use description, example, syntax, timing, topic, style and level of help according to user-demonstrated experience and proficiency. A description that advertises a command or function is helpful for getting started, but might become ignored if it is presented too often. Example information demonstrating how to perform a procedure is often valuable until the procedure is mastered. Syntax information generalizing the procedure becomes valuable when the procedure is more or less mastered.

An example user is starting to write Lisp programs, typing `(DEFUN`. The help pane serves as a reminder that the function being defined must be named and then given an argument list. The system provides an abbreviated syntax: `DEFUN function-name ( ) function-body)`, omitting the difficult argument types (optional arguments, keyword arguments, etc.). As the user types `TIMES-2 (I) (PLUS,` the system shifts its focus to helping with the `PLUS` function, and an example of a previous use of `PLUS` is displayed. The user realizes that adding numbers was not intended and backspaces and types `TIMES I 2)`. The system changes its focus of help to `TIMES` as it is being typed, and back to `DEFUN` when the user is done with `TIMES`.

Intermediate-style programmers have problems keeping track of the context and appropriateness of program pieces. The adaptive teaching scenario works to keep this type of programmer oriented by providing context-sensitive help and user examples.

The example user's experience with Lisp data structures is more extensive, however. When `(SETF` is typed, the system's AUM knows to show only the very complex argument list syntax for the `SETF` function. If an error is made (e.g., wrong argument type), the system changes its view of the user's expertise slowly, at first giving examples of the use of `SETF` and information on `CONS` cells (a related topic) to support progress. Experts use complex arguments even if they can't remember them, but often have enough experience so as not to need examples to remember usage.

Agents that Teach

Agents are computer programs that simulate a human relationship, by doing something that another person could otherwise do for you. Negroponte draws the lovely image of a butler performing tasks with a clairvoyant understanding and ability to take care of user needs [12]. To the extent that such a system understands how a particular user's needs differ from what the standard interface presents, it builds a private interface between the computer and the user. In this interface, the agent will "understand" the user's needs to perform formerly complex or unknown tasks with computer-created simplifying macros.

We could imagine that it might always be best for the agent to perform tasks for the user. Unfortunately, the reliance on such an assistant is reminiscent of the dependency children experience when relying on others who know the ropes. Such an agent, an *assistant-style agent,* is building a relationship in which its very success creates dependency for the user. As the system builds agent macros to save the user time and frustration, a private language develops between the assistant and the user. If a human coach tries to help a user, the lack of common interface language can become a barrier. If the user has problems, or loses the assistant (during a disk crash, computer repossession or company layoff) the user is back at novice level.

The search for a zipless interface should persist and assistant-style agents will be part of this. We want the interface language to feel "ready to hand" in the sense in which Heidegger [20] describes useful tools that become invisible to the user in a task. We try to design interfaces with well-thought-out metaphors of what the interface is like and carefully designed scenarios of how to perform tasks. When they do not do what we expect, we need help to teach us what the interface does do and to explain the metaphor so that the system will make sense.

An *advisory-style agent* builds a user relationship with the explicit goal of educating the individual. One could see this goal in terms of the fishing fable: give a person a fish and you've fed them once, teach a person to fish and you have fed them for life. The assistant-style agent gives the user a fish, the advisory-style agent teaches the user to fish.

Teaching Systems

Various systems have demonstrated and evaluated reasoning in educational situations [2, 3, 5, 13]. Learning technology used in software educational goals has been scarcer [6, 18]. For some years we have been building and working with interfaces that offer users help and tutoring support when needed, without interrupting user tasks [16].

The Proactive Interactive Adaptive Computer Help (PIACH) scenario for adapting help to a user [18] runs concurrently with the computer program in use. COACH is an adaptive interactive help system that implements the PIACH scenario, changing its model of a user character-by-character as they work. The computer creates a record in an adaptive user model (AUM) of a user's experience and expertise. Machine-learning and reasoning techniques attempt to provide help to match the needs of the particular user. Such help is said to be provided proactively when the computer anticipates user needs to present help before it is requested. Both the user and the computer can initiate help in a mixed-initiative interaction. Several representations work together to create help for the COACH user: the subject frames (definitions of the domain), the adaptive frames (recording of a user relative to a domain), the presentation rule sets (which embody a model of teaching) and the multilevel parser (syntax definition of domain).

The COACH System

Initially researchers attempted to create systems that would analyze user work to understand semantics. Such ambitious approaches were computationally impractical [6, 22]. Even though personal computers now have astonishing performance, any successful use of artificial intelligence (AI) that is to work in interactive environments must choose reasoning and learning goals using a model of computation. Finally, actual systems are being built that use learning in the user interface to change the user experience [10, 16].

COACH represents user activity to reason about how to provide help as the user is typing. Several strategies limit knowledge search and access problems. Relationships in the knowledge representation are recorded in predefined links. By limiting the depth of relationship links, search difficulty caused by complex links is decreased. To limit representation growth and reasoning difficulty most user model characteristics are recorded as scalars. Reasoning is conducted by small rule sets. These strategies allow COACH to be computationally practical.

The COACH interface separates user input from computer output and advisory help to provide visual aids, permitting the user to focus more attention on the problem to be solved and less on the computer mechanics (see Figure 1).

A system could build the adaptive model by asking a user questions [14]. We, instead, explore user models that are built by watching the user's actions [16].

COACH uses an explicit user model (see Figure 2). Frames, facts and rules represent the user and the skill domain the user is learning. The AUM is a set of user model frames [11] for syntactic and conceptual parts of the domain being coached. While the user is working on a task, these frames record aspects of the user's successes and failures. This representation of the user and an associated reasoning system for creating and accessing knowledge frames comprises the AUM. The defined network of relationships between skill domain parts, what the user is doing, and the state of the user model is the basis for selecting user help.

Domain Knowledge is represented in the help system subject frames, adaptive frames and the parser grammar. *Statements, tokens, concepts* and *basis sets* are the *learnable things* in COACH, the smallest quantities of information represented as discrete entities to a user. COACH domain knowledge is composed of frames for each of these learnable things, each with slots describing its utility and the user's facility with it.

To use a computer language effectively, a student needs to understand its syntax and semantics. So, it is reasonable to use a syntax definition as part of the structure of the representation for teaching. This definition is