A BETTER WAY TO HELP

A computer help system that combines the patience of a machine with the watchfulness of a human lets users learn faster and perform better

Ted Selker's COACH system is the first adaptive help system that both works interactively and provides users with immediate productivity benefits.

In the absence of a royal road to learning, the next best thing is a good tutor. That thought has no doubt often sprung to mind in those trying to learn a new programming language or application in which the nearest source of instruction is a fat, dense manual. Even in the few ideal instances in which the manual or accompanying tutorial really does provide help in getting started, most users at some point find themselves laboriously trying to recall details from an earlier lesson or struggling to marshal their newly acquired knowledge to solve a problem not dealt with in the book. What we need in such situations is a tutor, or coach, someone who, by watching what we do, will patiently guide us through the fog of half-assimilated knowledge and keep us from getting too far off course.

But humans are not needed in all cases. In learning a computer language or application, one often commits elementary but frustrating mistakes that the computer—with its distressingly literal approach to communication—simply balks at. Yet, in principle, computers are quite capable of acting in a more flexible manner, responding to an error in command syntax, for example, with a list of possible alternatives. How much simpler, then, to take the next step and turn over the entire tutoring operation to the computer.

"The idea," says Selker, "was to have the computer create a model of your abilities and then be able to change, or adapt, the model by watching what you do."

Harnessing computers as tutors, or coaches, is not a new idea, and various approaches have been tried. At the most basic level, at which the computer functions as a surrogate for an instruction manual or textbook, popping up help screens at the user's instigation, there is little that one can properly call tutoring. Nevertheless, there are challenging and important issues in designing online help menus, and such static, passive "help" systems are useful to a point.

The intelligent tutor

Qualification as a tutor requires a more sophisticated appreciation of the learner's predicament than can be incorporated in a noninteractive program. So-called intelligent tutoring systems, first introduced in the early 1970s, go considerably further toward providing functions that one normally associates with a human. Instead of trying to simply mimic a textbook or manual, these systems attempt to capture features that reflect the interaction between a student and a teacher. One of the first such systems, called SCHOLAR, developed by Jaime Carbonell in 1970, posed questions to the student and allowed the student to ask questions about the subject domain (South American geography).

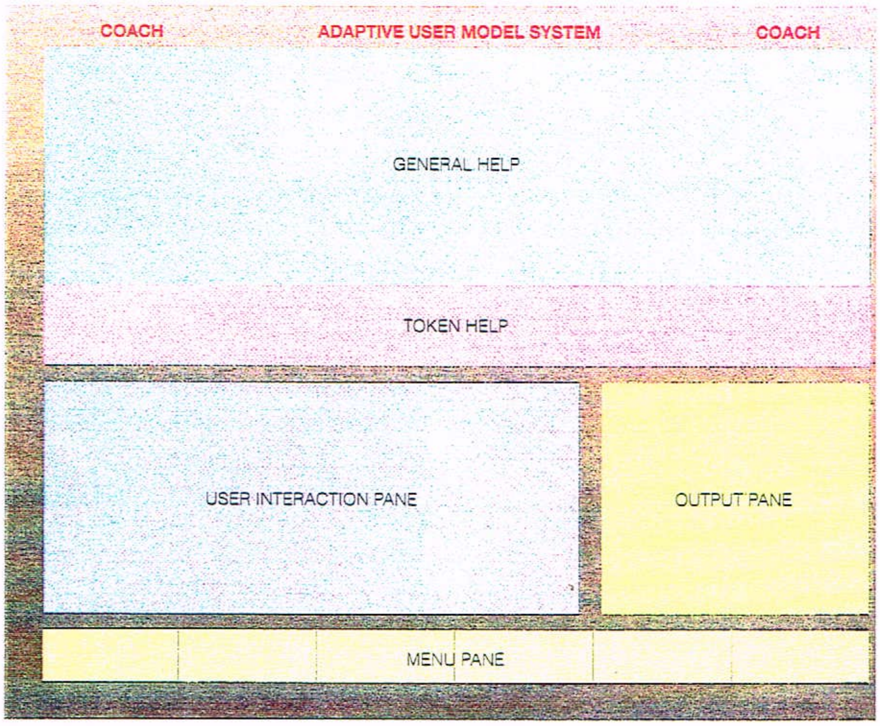
But is that what it really means to enlist the computer as teacher or tutor? "Since people first started working with computers, they have typically imagined it was like talking to a person," says Ted Selker, mathematical sciences, Watson (now in computer science, Almaden). Unfortunately, that's not the case, he notes. The computer doesn't know anything about the person and just does what it's told.

For a computer to be of real assistance, it needs to know enough about the learner to pitch its explanations and advice at the appropriate level, just as a skilled tutor does. That realization led Selker, in 1983, while a visiting graduate student at Stanford University, to begin thinking about a computer help program that would adapt to the individual needs of each user, an approach that he captures in the phrase "Proactive Interactive Adaptive Computer Help."

"The idea," says Selker, "was to have the computer create a model of your abilities and then be able to change, or adapt, the model by watching what you do." Selker implemented the idea in a system he called COACH, an acronym for COgnitive Adaptive Computer Help. While other people had discussed adaptive user models, COACH represented the first implementation that worked in real time, providing help instantly, at the first sign of difficulty.

Most important, Selker demonstrated that people using COACH showed measurable performance improvements compared to a control group using a nonadaptive help system. His behavioral studies, which constituted the first proof that such benefits could be gained from an adaptive system that automatically offers help and tutoring support, have done much to resolve the long debate over the value of such systems.

The window interface separates user input from help and system responses. A menu at the bottom allows a user to request help directly.

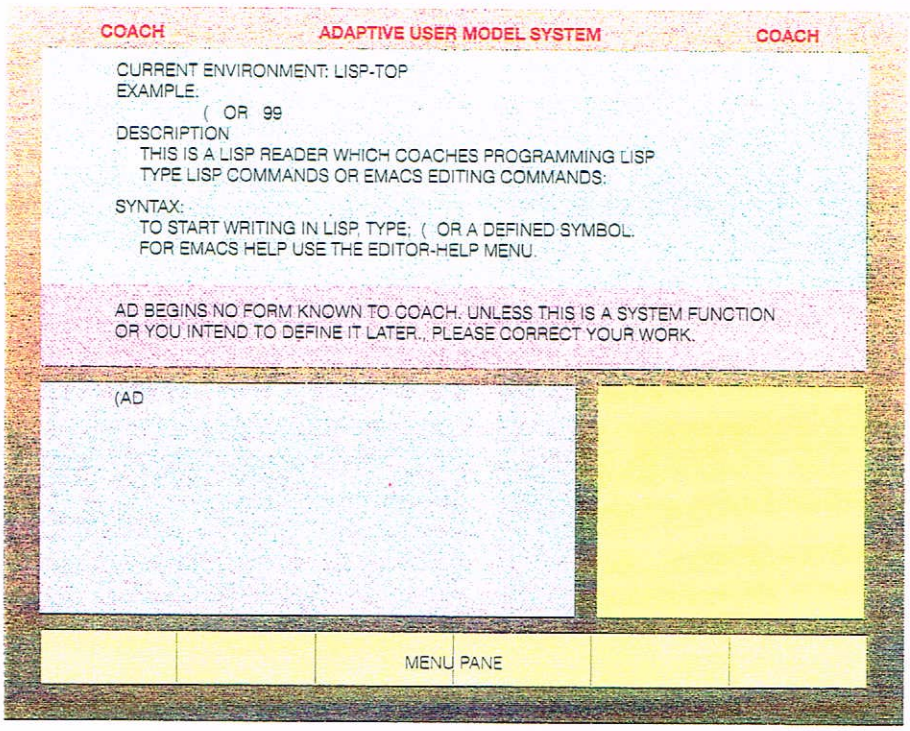


Adapting to the user

Much like a typical human tutor, COACH operates within a specific domain of knowledge, though different domains can be built on top of its underlying adaptive user model (AUM). Initially, Selker set up COACH to teach Common LISP, a programming language that has been widely used by the artificial intelligence (AI) community.

Though the idea of a tutor or coach suggests help for beginners, COACH provides assistance for experts as well as novices. The AUM, by embodying, in effect, a teacher's sense of a student's mastery of a lesson or subject area, classifies users in four levels of proficiency based on the students' responses, questions and actions. Because the user model is adaptive, it changes as

A COACH interface during a simple error situation.



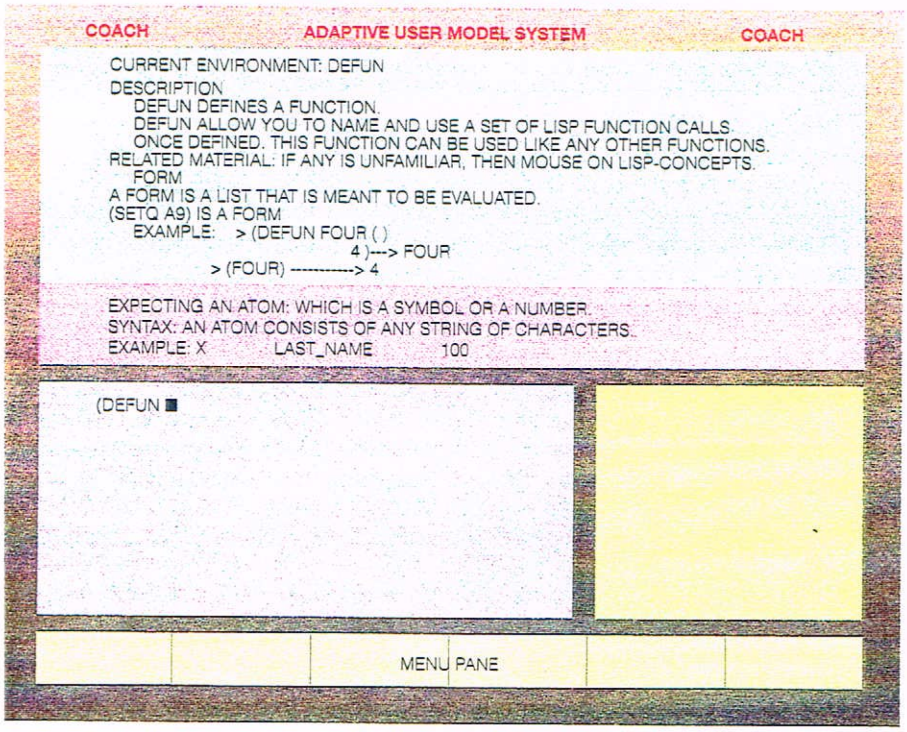
```
CURRENT ENVIRONMENT: LISP-TOP
EXAMPLE.
            ( OR  99
DESCRIPTION
    THIS IS A LISP READER WHICH COACHES PROGRAMMING LISP
    TYPE LISP COMMANDS OR EMACS EDITING COMMANDS:

SYNTAX:
    TO START WRITING IN LISP, TYPE;  (  OR A DEFINED SYMBOL.
    FOR EMACS HELP USE THE EDITOR-HELP MENU.

AD BEGINS NO FORM KNOWN TO COACH. UNLESS THIS IS A SYSTEM FUNCTION
OR YOU INTEND TO DEFINE IT LATER., PLEASE CORRECT YOUR WORK.

(AD
```

A COACH interface supporting learning about a specific form and the idea of a form



```
CURRENT ENVIRONMENT: DEFUN
DESCRIPTION
    DEFUN DEFINES A FUNCTION.
    DEFUN ALLOW YOU TO NAME AND USE A SET OF LISP FUNCTION CALLS.
    ONCE DEFINED. THIS FUNCTION CAN BE USED LIKE ANY OTHER FUNCTIONS.
RELATED MATERIAL: IF ANY IS UNFAMILIAR, THEN MOUSE ON LISP-CONCEPTS.
    FORM
A FORM IS A LIST THAT IS MEANT TO BE EVALUATED.
(SETQ A9) IS A FORM
    EXAMPLE:    > (DEFUN FOUR ( )
                              4 )---> FOUR
             > (FOUR) ----------> 4

EXPECTING AN ATOM: WHICH IS A SYMBOL OR A NUMBER.
SYNTAX: AN ATOM CONSISTS OF ANY STRING OF CHARACTERS.
EXAMPLE: X          LAST_NAME          100

(DEFUN ■
```

the learner's skill level increases (or even decreases), and the type of help it offers is correspondingly modified.

To achieve this personalized instruction, COACH contains both a knowledge of the subject domain—which so far has included LISP and UNIX® but could be another programming language, an area of mathematics or, in principle, any corpus of knowledge—and a shell, which consists of a set of "interacting objects" that combines domain knowledge and knowledge about the user in order to perform various coaching functions.

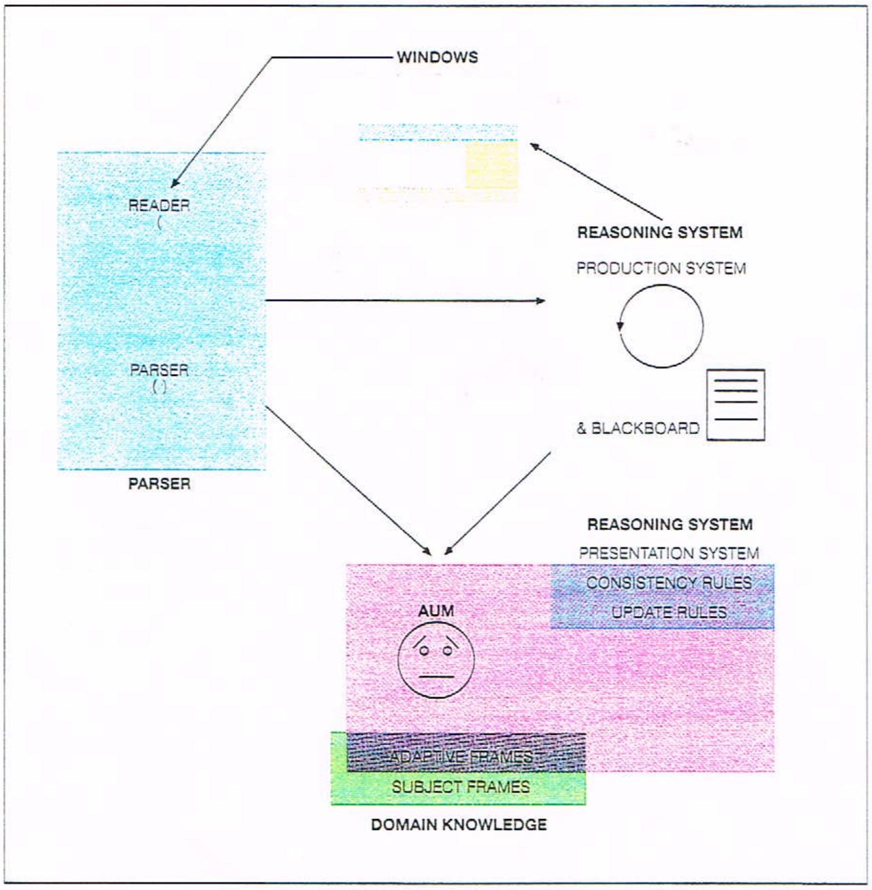
Coach presents all users, no matter what their level, with a computer screen divided into four windows, or panes. The user works in the interaction pane, and the result, for example, of an arithmetic calculation computed with LISP, is shown in an output pane. Above these working areas of the screen are two windows with help, or tutoring, information. The first, called token help, is directly over the interaction pane and provides help—primarily to novices—on specific kinds of entities, or tokens, such as numbers, symbols, or defined variables. All other information provided by COACH is presented in another window, called general help.

Like a good teacher, COACH encourages novices to get started, jogging their memory by illustrating, for instance, the general form of LISP syntax. A typical beginner's error might be typing ADD instead of the correct LISP function PLUS. As soon as the user begins typing, COACH intervenes by posting in the token-help window the remark: Ad begins no form known to COACH. Unless this is a system function or you intend to define it later, please correct your work.

Such gentle reminders and admonitions are expected to prompt the student to recall the correct function or request additional help from COACH's user-initiated help screens. More advanced users require and receive more specialized assistance.

UNIX is a registered trademark of UNIX system Laboratories, Inc., in the U.S. and other countries.

The COACH architecture is composed of interacting parts, or objects, and the arrows show the information flow among them. The window interface manages text editing, output formatting and menus. The reasoning system creates and uses the adaptive user model (AUM) to display domain-knowledge help and to modify domain knowledge. Coaching knowledge controls these reasoning activities. A parser notes a user's work context and dispatches information to the reasoning system.



There are presentation rules for each kind of learnable unit. for example, the rules for statements—syntactically correct combinations of keywords and variables in LISP or, more generally, in any skill domain—include the following:

- Losing Ground—provides the user with the most basic help
- Out of Practice—reminds the user of information that was previously understood
- Encourage Exploration—suggests useful information not presently being used
- Veto Overly Sophisticated Help—protects the user from help beyond an appropriate level
- Veto Extra Help—protects the user from too much help

The Losing-Ground presentation rule is expressed in the conditional form:

IF statement used has a low goodness score and a low learning slope, THEN present a user example and a system example

AI and the intelligent tutor

Behind this sophisticated interface lies an architecture built on concepts drawn from the field of artificial intelligence. The main elements are the AUM, a reasoning system and a parser; together, they monitor the user's actions, update the AUM and then make decisions about the type of help to provide.

In order to evaluate the user's needs, COACH has to be able to compare the user's knowledge to the total knowledge of the subject domain itself. "Everything that can be learned about the domain," explains Selker, "is expressed in terms of learnable units, each of which is recorded in a frame, consisting of slots for specific pieces of knowledge." Among the learnable units are groups of interrelated units, such as those units that must be mastered in order to understand another unit. The set of subject-frames, therefore, both defines the domain and the logical interrelations within it. COACH draws on the content of the subject frames to present information in the help window.

Another set of frames embodies knowledge about the user, which, in toto, constitutes the AUM. Called adaptive frames, they include examples of user errors and subsequent corrections, data on how many times a learnable unit has been used and how long since it was last used, a measure of how fast the user is learning or forgetting (slope), and a measure of the user's progress with respect to a particular learnable unit (goodness).

COACH's tutoring knowledge, or set of teaching skills, is embodied in a reasoning system that both continually revises the AUM and determines what parts of the domain knowledge—that is, which subject frames—are to be presented to the user in a given situation. The decision is based on the AUM, and on what the user is doing at a particular moment. In effect, the system reasons, "If the user satisfies such and such profile (as recorded in the adaptive frames) and if the user makes this kind of error, then present such and such information."

There are separate presentation rule sets for different kinds of learnable units (*see box above*). The Losing-Ground rule, for example, kicks in when a slow learner making little progress exhibits trouble. The idea is to first remind the user of something he once knew by showing a correct instance of the user's own input, followed by an example drawn from the system's help text at the user's level.

The Veto-Extra-Help rule expresses the idea that, if the user is exhibiting a lower level of competence than in the past, the amount of detail in the help message should be limited.

This description of the rules is not literally correct in one respect. The help messages that actually appear in the help window are not necessarily the consequents of the rules. Rather, the consequents are reg ded as proposals, which are first posted on a "blackboard," where they are able to interact with vetoes.[1] The order of the rules in the rule set dictates the order in which they are applied, and vetoes are able, under certain circumstances, to remove proposals from the blackboard, so that the user does not see them.

The key to the entire system is the accuracy of the AUM. If it is not adequately maintained, the presentation rules function like a lecturer speaking to an unknown audience in the dark—there is no way to judge the appropriate level of the presentation or to infer whether it is being understood.

The task of creating and maintaining the AUM is the responsibility of update and consistency rules (*see box above right*). These rules are run at the instigation of another element of the COACH architecture—the parser, whose function is to determine what the user is typing in the input window. Each time the parser signals a change in the "parse state"—potentially, as each character is typed—the rules change the AUM frames. "In this way," Selker explains, "user model frames are revised each time a function is closed, a token is typed or found to be undefined, and so on."

[1]The use of blackboards was first introduced by L. D. Erman and Victor Lessor in 1975 in Hearsay, a speech-activated, chessplaying program.

Some examples of update rules are:

- Note Success—activated by a correct use of a learnable unit; it improves the user's rating on the AUM frame slots for that unit and related material
- Was Good but Getting Worse—activated when a user begins making mistakes with a learnable unit that had previously been rated well; it decreases the relevant rating slowly

Consistency rules complement the update rules. Examples include:

- Note Used—activated each time a learnable unit is used
- Bound Goodness and Best—activated to record user's "personal best"

It works!

To find out how well COACH worked, Selker and his student assistants carried out a number of user studies. They began with the knowledge of the inauspicious fact that, traditionally, help systems have been shown not to increase the speed with which a user completes a given task. While in the long term there are productivity improvements as a result of the user's increased skills, in the early stages the distraction of accessing the help menus can slow a person down. There was every reason to expect, then, that an adaptive system that interjected advice and corrections on its own would be even more distractive.

But that wasn't the case. Instead, says Selker, "our studies showed that those using the full system were more comfortable with the tasks and materials and, moreover, outperformed users who had access to the static help menus but not the adaptive part of the system."

Future enhancements

Many of the AI concepts that underlie COACH's architecture were derived from the technology of expert systems—programs that embody the skill of an expert in a given domain. COACH, in fact, can be regarded as a special kind of expert system—a learning expert system—in which the expertise is not contained in the domain itself, for example, LISP—but in how to characterize, or capture, the knowledge in the domain, build a model of a user relative to it, and present help based on the model and a set of rules.

As in an expert system, in COACH one can separate the structure—the so-called shell—from the domain content. COACH, therefore, can be adapted—relatively simply—to work with other programming languages and, in principle, with any corpus of text-based knowledge. Already, COACH has been extended to teach the UNIX operating system's shell command language.[2]

Further uses are also being contemplated, such as with graphics and multimedia applications. But the full scope of a system like COACH is broader still. "This technology," says Selker, "is useful anytime the software tool—whether a programming language or application—is sufficiently complicated that users are less interested in spending time mastering the intricacies of the tool than in plunging as quickly as possible into actual problem solving. COACH allows the user to focus on the task, learn how to use the tool and become productive far sooner than would otherwise be possible."

Finally, though the current prototype of the COACH system runs on the IBM RT PC running the MACH operating system, as well as on the Symbolics computer family, work is currently under way to make it possible to run COACH on a personal computer platform.

—R.L.D

[2]This work was done by Matt Schoenblum, a 17-year-old high-school student. He carried out this project during a 10-week internship, part of Watson's educational outreach program, which is dedicated to promoting science and mathematics education at local high schools.