Technical Report No. 474

# COGNITIVE APPRENTICESHIP AND INSTRUCTIONAL TECHNOLOGY

Allan Collins

BBN Laboratories
Cambridge, Massachusetts

July 1989

CENTER FOR THE STUDY OF READING

Technical Report No. 474

# COGNITIVE APPRENTICESHIP AND INSTRUCTIONAL TECHNOLOGY

Allan Collins

BBN Laboratories
Cambridge, Massachusetts

July 1989

University of Illinois at Urbana-Champaign
51 Gerty Drive
Champaign, Illinois 61820

The work upon which this publication was based was supported in part by the Office of Educational Research and Improvement under Cooperative Agreement No. OEG 0087-C1001, with the Reading Research and Education Center. The publication does not necessarily reflect the views of the agency supporting the research.

## Abstract

In earlier times, practically everything was taught by apprenticeship: growing crops, running trades, administering governments. Schools are a recent invention that use many fewer teaching resources. But the computer enables us to go back to a resource-intensive mode of education, in a form we call cognitive apprenticeship (Collins, Brown, & Newman, in press). Cognitive apprenticeship employs the modelling, coaching, and fading paradigm of traditional apprenticeship, but with emphasis on cognitive rather than physical skills. My basic thesis in this paper is that technology enables us to realize apprenticeship learning environments that were either not possible or not cost effective before.

This paper addresses the questions: What kind of leverage do we derive from computer technology, and what design criteria can we specify for building computational learning environments? We have developed a tentative set of characteristics (Collins, Brown, & Newman, in press) we think computational learning environments should have, based on analyzing what kinds of tutoring systems we see emerging, what we have learned from studies such as Lampert (1986), Palincsar and Brown (1984), Scardamalia, Bereiter, and Steinbach (1984), and Schoenfeld (1983, 1985), and what resource rich learning environments (such as tennis coaches and graduate school instruction) are like.

This paper discusses six characteristics of cognitive apprenticeship for which technology provides particular leverage. For each abstract characteristic I will address: (a) what the abstraction refers to, (b) the implications technology has for realizing the abstraction in practice, (c) why realizing the abstraction is of benefit to students, and (d) an example of a computer system that embodies the abstraction.

# COGNITIVE APPRENTICESHIP AND INSTRUCTIONAL TECHNOLOGY

## Situated Learning

Situated learning is the notion of learning knowledge and skills in contexts that reflect the way the knowledge will be useful in real life. It is the sine qua non of apprenticeship. But it should be thought of in the most general way. In the context of math skills, they might be taught in contexts ranging from running a bank, or shopping in a grocery store, to inventing new theorems or finding new proofs. That is, situated learning can incorporate situations from everyday life to the most theoretical endeavors (Schoenfeld, in press).

The computer allows us to create environments that mimic situations in the real world that we cannot otherwise realize in a classroom (or home). One approach is through microworlds, but also through computer networks, data bases, graphing packages and text editors (Collins, 1986). There are inherent dangers in microworlds, such as learning to make decisions based on a few variables rather than a rich set of variables (Dreyfus & Dreyfus, 1986), but potentially the benefits far outweigh the risks.

### Benefits of Situated Learning

**Students learn conditions for applying knowledge.** By learning arithmetic, for example, in representative contexts such as grocery shopping or running a bank, the student ties the knowledge learned to specific contexts. Then when they are in novel situations, for example, buying airline tickets or working in an accounting department, they will be able to see how the knowledge learned might apply in these new situations by analogy to the situations they learned about.

**Situations foster invention.** When students use their knowledge to deal with real problems and situations, they are forced to make inventions to apply their knowledge (Lampert, 1986). Thus they are learning how to use their knowledge flexibly to deal with novel situations.

**Students see the implications of the knowledge.** When learning is embedded in context, then its uses are more apparent to students. They can actually see how the knowledge is used in different settings, and what power it gives them to use the knowledge. It is not readily apparent to students how most of what they learn in school might be used.

**Context structures knowledge appropriate to its uses.** When students learn things for school, they often invoke suboptimal schemes for remembering the information. For example, they may infer that all arithmetic word problems with the word "left" (e.g., "Mary had 7 apples. She gave 3 to John. How many did she have left?") are subtraction problems (Schoenfeld, in press). Or they may memorize facts in a rather rote way so that they can be retrieved for a test (e.g., the capital of Oregon is Salem: With the *oar* in Oregon you can go *sail*ing). When knowledge is learned in the contexts of its uses it is more likely to be stored in a form that is usable in novel contexts.

These may be the central issues as to why transfer of knowledge is so difficult: By learning in multiple contexts you learn different ways knowledge can be used, and begin to generalize on these ways. This is opposed to the way we teach knowledge in an abstract way in schools now, which leads to strategies, such as depending on the fact that everything in a particular chapter uses a single method, or storing information just long enough to retrieve it for a test. Instead of trying to teach abstract knowledge and how to apply it in contexts (as word problems are supposed to do), we advocate teaching in multiple contexts and then trying to generalize across those contexts. In this way knowledge becomes both specific and general.

An excellent example of a computer-based situated learning environment based on a microworld is *Geography Search* by Tom Snyder (Snyder & Palmer, 1986) which is one of five programs in the Search Series by McGraw Hill. It teaches history, math, planning, and problem solving. In this simulated microworld, groups of students sail ships from Europe to the New World about the time of Columbus, in order to look for a treasure that is distributed around North and South America. Land and other ships come into view on the screen when the ship nears them. Students have to calculate their route using sextant and compass in the way sailors did of old. They must also keep track of food and supplies, so they don't run out while they are at sea. So students are learning history and math in a context where novel problems continually arise.

Another example of situated learning is the reconvening of the Constitutional Convention among school students in the Boston area using a computer mail system (personal communication, William Fitzhugh). Different schools represent different delegations (e.g., Delaware, Virginia). The students prepare by reading about the concerns of their states in 1787. During the convention they will try to negotiate a draft constitution to correct for the difficulties that were encountered with the Articles of Confederation. A similar kind of convention could be held to cope with the modern day problems that have arisen with the Constitution (e.g., the budget process, the advent of media and its expenses, the disagreements over who controls foreign policy, the difficulties when Presidents are disabled for any reason). Government and its structures become real in dealing with these kinds of questions.

History is typical of the information schools teach in a non-situated way. Schools try to pour in a lot of facts and theories, and make no use of that knowledge other than recall. Computers give us enormous power to create situated learning environments where students are learning about reading, writing, math, science, and social studies in ways that reflect the kinds of activities they will need these for.

## Modelling and Explaining

Modelling is showing how a process unfolds and explaining involves giving reasons why it happens that way. It is the showing and telling that is so characteristic of apprenticeship.

### Two Kinds of Modelling Important for Education

**Modelling of processes in the world.** For example, one might show how electrons move in circuits (Haertel, 1987), or how information coded in DNA is translated into protein molecules.

**Modelling of expert performance.** For example, in teaching reading, the teacher might read in one voice and verbalize her thinking in another voice, like a slow motion movie. She could verbalize what is confusing, what to do when you don't understand, any tentative hypotheses about what is meant and what will come later, any evidence as it comes in about these hypotheses, her summaries and integrations, her guesses as to the author's intentions, and evaluations of the structure and style of the writing: in short, all the thoughts of a skilled reader (Collins & Smith, 1982).

The computer makes it possible to represent processes in ways books never could, and even in ways people cannot. Computers can make the invisible visible: They let you see inside pipes or inside the body, how current changes in circuits based on electron flow, where the center of mass for a group of bodies is, how microscopic processes unfold. At the same time they can make tacit knowledge explicit, by showing the strategies experts use to solve problems that students set for them. To the degree we can develop good process models of expert performance, we can embed these in technology, where they can be observed over and over for different details.

Computers can use multimedia, that is, animation, voice, text and graphics to characterize different aspects of processes. Ways of integrating animation and voice are just beginning to be explored, but it is clear that they have enormous potential for making things clearer. For example, it is possible to highlight each component of a system while it is talked about. One can show both what happens and what does not happen. We can render unto voice what verbal description best transmits (e.g., reasons why, abstract ideas) and render unto animation what visual description best transmits (e.g., processes and relations between components, concrete ideas). And we can achieve *simultaneous presentation* so that what is seen happening on the screen can be explained orally without looking elsewhere. Eventually much of the information in textbooks and libraries will be in this form.

## Benefits of Modelling

**Seeing expert solutions to problems set by the student.** For the most part in school, students never see how experts solve problems--they see only worked out solutions. Worked out solutions do not show the false starts or dead ends that characterize real world problem solving. If students can pick problems that raise issues in their mind, and watch how an expert computer program attacks the problem, then they will have a genuinely new kind of learning experience.

**Integrating what happens and why it happens.** Demonstrations in class or process descriptions in text have inherent limitations. Many demonstrations occur too fast to assimilate what is happening and why it is happening. In books simultaneity can only be approximated with static diagrams and text explanation. The ability to see what is happening, in slow motion if need be, and hear at the same time a verbal explanation of why it is happening facilitates building an integrated understanding of processes.

**Making visible parts of a process not normally seen.** Because much of what we want students to learn is not ordinarily seen, students must infer from end products and a few intermediate states how processes unfold. By revealing these processes in detail, many more students will have a chance to figure out what is happening.

I can illustrate modelling with a computer system called *Summit* (Feurzeig & White, 1984), designed for teaching addition and subtraction. Summit combines visual animation with spoken explanation. The system had two representations for addition and subtraction: the standard algorithm and a bin-model representation, derived from Dienes (1960) blocks. In both representations students can pose problems to the system to see what happens. Figure 1 illustrates how the system represents a simple addition problem (2593+9) in the two representations.

**[Insert Figure 1 about here.]**

The bin model works as follows. Suppose that the ones bin has 3 icons in it and 9 more are added to it. The model first displays all 12 icons, stacked one on top of the other, in the ones bin. Next, the computer says, "Now there are too many ones. We have to take some to the tens." An empty box then appears in between the ones and tens bins. Next 10 icons are removed from the ones bin, one at a time and placed in the box. There is a counter displayed on the box. When there are 10 icons in the box, the box becomes a tens icon, which is added to the tens bin. This in turn leads to an overflow in the tens bin, and the process repeats itself.

Summit models the addition process for the standard algorithm in a similar way, saying: "Then we add the ones. 3 plus 9 is 12, but 12 won't fit. So we write the 2 under the ones column, and take the 1 over to the tens column," writing it just above the tens column. "Then we add the tens. 1 plus 9 is 10, but 10 won't fit. So we write 0 in the tens column, and take the 1 over to the hundreds column," and so on. Each of the actions appears on the screen as the voice explains it.

There are other systems such as Sophie (Brown, Burton, & deKleer, 1982) and Quest (White & Frederiksen, in press) in the electricity domain that not only model how a process unfolds, but how experts troubleshoot a faulty system. In these systems students can see an expert troubleshooting whatever faults they decide to set in the system. Thus, they can pick problems that are at the edge of their own competence in order to see how to extend their troubleshooting strategies.

## Coaching

Computer systems have the ability to patiently observe students as they try to carry out tasks, providing hints or assistance when needed. This kind of personal attention is simply not feasible in most school classrooms.

Not only is the computer patient, it can remember perfectly what the student did before. It can consider multiple hypotheses about the difficulties the student is having (Burton, 1982), and it can observe over a period of time in order to tell what problems the student is really having. Moreover, a computer coach gives students a different perspective from which to understand their own performance.

## Benefits of Coaching

**Coaching provides help directed at real difficulties.** By observing students in a problem solving situation, the coach can see what difficulties a particular student is having. Because teachers in school rarely have a chance to observe student problem solving, most of the help they give is not really directed at the problems students actually have.

**Coaching provides help at critical times.** Coaching provides help to students when they most need it: when they are struggling with a task and are most aware of the critical factors guiding their decisions. Thus they are in the best position to use any help given.

**Coaching provides as much help as is needed to accomplish tasks.** Coaching enables students to do tasks they might not otherwise be able to complete. It gives them a sense that they can really do difficult tasks. As they become more skilled, the coach's role can fade giving students more and more control over execution of the task.

**Coaching provides new eyeglasses for the student.** Coaching can help students see the process from an entirely different perspective. The coach can point out things that do not go as expected and explain why. Furthermore, a coach can introduce new terms such as "forward chaining" and "thrashing" (see below) that are critical to employing heuristic and metacognitive strategies.

The best example of a computer coach is the coach built by Burton and Brown (1982) for the computer game "How the West Was Won." Originally developed as part of the Plato system, the game, which is a variant on Chutes and Ladders, is designed to teach children basic arithmetic operations. It can be played either by one person against the computer or by two people against each other. Figure 2 shows a display of what the game board looks like to the players.

**[Insert Figure 2 about here.]**

The rules of the game are as follows: When it is a person's turn, they must form an arithmetic expression from the three spinners using two different operations (e.g., 2 x 1+2), and after they have formed such an expression they must input the value of the expression (in this case 4). If they miscalculate, they lose their turn; if they are correct, they move forward on the board the number of spaces calculated. The object is to reach the last town (which is 70) before your opponent, but to reach the last town you must form an expression that lands you exactly on the town (i.e., you cannot overshoot it). There are some special rules that make the game more challenging: (a) if you land on a town (every 10 squares) you advance to the next town, (b) if you land at the beginning of one of the short cuts (i.e., 5, 25, 44) you advance to the end of the shortcut (i.e., 13, 36, 54), and (c) if you land on your opponent, you send her back two towns.

These special rules make it advantageous to consider many possible different moves, which would require trying out different arithmetic expressions and calculating their values. But students playing the Plato game do not consider alternative moves: They tend, instead, to lock onto a particular strategy, such as multiplying the largest number times the sum of the other two numbers, which gives them the largest value they can make. Thus, they do not play very well or learn much arithmetic without coaching.

To remedy this situation, Burton and Brown (1982) built a computer coach that observes students as they play the game and gives them hints or advice at critical moments. The computer coach rank orders every possible move a player might make given the three spinner values. The rank order is constructed with respect to how far ahead of your opponent any move leaves you. This gives the coach a way to evaluate the effectiveness of any move.

The coach observes the players' moves by looking at certain "issues," for example, whether the student knows to land on towns or on the opponent when it is effective to do so, whether the student knows to use parentheses or the minus sign or the divide sign when it is effective to do so. If the student systematically fails to make moves that require understanding of any of these issues, then the coach will notice the pattern. The coach will then intervene with a hint about a particular issue when that issue is particularly salient to making a good move; that is, when the student's move is much inferior to the best move possible, taking into account that issue. The intervention by the tutor occurs just after the student's move: The system points out how using parentheses or a short cut, for example, would improve the player's position, and gives the student a chance to retake their turn. Thus the coach tries to expand the student's awareness of how they might play the game more successfully.

There are other examples of computer coaches that help students when they are carrying out tasks. For example, Anderson, Boyle, and Reiser (1985) have developed computer coaches for plane geometry and computer programming that pose problems to students and offer advice when the student takes a step in working the problem that reflects a common misunderstanding or error. Similarly, the PROUST system of Johnson and Soloway (1985) recognizes common errors students make in computer programming, and gives advice as to what the problem is. Computer coaches make it possible for students to spend their learning time actively carrying out tasks and projects, receiving personal help or guidance when they need it.

## Reflection on Performance

Reflection refers to students looking back over what they did and analyzing their performance. There are four ways of permitting students to reflect on their performance (Collins & Brown, 1988): (a) imitation, (b) replay, (c) abstracted replay, and (d) spatial reification. These can be illustrated in the context of tennis. *Imitation* occurs when a tennis coach shows you how you swing your racquet, perhaps contrasting it with the way you should swing it. *Replay* occurs when he videotapes your swing, which you can compare to videotapes of experts. An *abstracted replay* might be constructed by using reflective tape on critical points--the elbows, wrist, end of racquet, etc.--so that the student can see how these points move with respect to each other in a videotaped replay. A *spatial reification* might be a plot of the trajectory of these points moving through space. In general, students should be given multiple views, and be able to compare their performance to expert performance. Technology makes the last three forms of reflection possible, so this is a genuinely new teaching method emerging out of technology.

### Benefits of Reflection

**What the student did becomes an object of study.** The students begin to see their performance on tasks as data to be analyzed. They may never have taken what they did before seriously: Reflection encourages them to think about their processes from the point of view of how they might be different, and what changes would lead to improved performance.

**Students can compare their performance to others'.** Reflection lets students see how different students and more expert performers carry out the same task. This encourages them to form hypotheses about what aspects of a process are critical to successful and unsuccessful performance.

**Abstractions about the process can be used for characterizing strategies.** It is possible to describe various heuristics and metacognitive strategies (Schoenfeld, 1985) in terms of the process the student is reflecting on. For example, in working geometry problems (see below), it is possible to characterize forward chaining as working from the givens and backward chaining as working from the statement to be proved. A good heuristic strategy is to start forward chaining from each of the givens to see what they imply, and then switch to backward chaining to work out the problem solution.

**Spatial reification permits comparison of multiple performances to form abstractions.** If students can see a process laid out in graphic form, then they can compare different people's approaches and try to characterize what aspects of the process are critical to expert vs. novice performance. The spatial representation permits them to see and even measure aspects of the process that are not apparent in a replay.

A good example of a spatial reification that permits reflection on a process is the problem solving trace that Anderson, Boyle, & Reiser's (1985) Geometry Tutor constructs as the student works a problem. Figure 3 shows three views of the screen as a student works a problem. Initially in the top view the student sees a screen with the givens at the bottom and the statement to be proved at the top. In the middle screen the student has worked part way through the problem, working both forward from the givens and backwards from the statement to be proved. The third screen shows the final problem solution, along with various dead ends (e.g., <MDB = <MCA) that the student never used in the proof.

**[Insert Figure 3 about here.]**

As Anderson, Boyle, Farrel, and Reiser (1984) point out, geometry proofs are usually presented in a fundamentally misleading way. Proofs on paper appear to be linear structures that start from a set of givens and proceed step by step (with a justification for each step) to the statement to be proved. But this is not how proofs are constructed by mathematicians or by anybody else. The process of constructing proofs involves an interplay between forward chaining from the givens and backward chaining from the goal statement. Yet, the use of paper and its properties encourage students to write proofs as if they were produced only by forward chaining--starting with the givens at the top of the page and working downward to the goal in a two column linear format (left column for the derived statements, right column for the logical justifications). If students infer that they should *construct* proofs this way, they will fail at any long proof. Properly designed computational learning environments can encourage students to proceed in both directions, moving forward, exploring the givens, and moving backwards, finding bridges to the goals.

The representation in Geometry Tutor is an abstraction of the problem solving process in terms of a "problem space." The system shows the states in the problem space that the student reached and the operators used to reach each of those states. Simply seeing the steps toward a solution reified in this way helps to create a problem space as a mental entity in its own right. This, in turn, makes it possible, for both teachers and students, to characterize problem-solving strategies in terms of abstractions that *refer* to the properties concretely manifested in the reified problem space. For example, in geometry it is a good strategy to forward chain at the beginning of a problem in order to understand the implications of the givens. Similarly, if you are stuck backward chaining, and do not see a way to connect your backward chain to any of the givens, then either go back to forward chaining or go back to the goal state again and try backward chaining along a different path.

Videotape first made systematic reflection on performance possible. The computer extends this to abstracted replays and reifications that highlight critical aspects of performance. In this way, students can analyze their performance from different perspectives and compare themselves to other students and experts.

## Articulation

Articulation refers to methods for forcing students to explain and think about what they are doing, that is, making their tacit knowledge explicit. There are two ways that computers provide leverage for encouraging students to articulate their knowledge. First, computers make it possible for students to actually build their theories or ideas into artifacts that can be tested and revised. One challenge for their fellow students might be to show the limitations or failures of these theories. Second, computers can provide tools and settings where students try to articulate their ideas to other students, as in the Constitutional Convention convened on an electronic network in Boston (see Situated Learning).

### Benefits of Articulation

**Making tacit knowledge explicit.** When knowledge is tacit, it can only be used in contexts that elicit the knowledge automatically, that is, that call up the knowledge because they are very similar to the conditions in which the knowledge was acquired. By forcing students to articulate their knowledge, it generalizes the knowledge from a particular context so that it can be used in other circumstances.

**Making knowledge more available to be recruited in other tasks.** Knowledge that is articulated as part of a set of interconnected ideas becomes more easily available. For example, if students acquire the idea of "thrashing" in problem solving (i.e., the concept of moving through a problem space without getting closer to the goal), then they can learn to recognize thrashing when they see it in different circumstances and develop strategies for dealing with it when it occurs.

**Comparing strategies across contexts.** When strategies are articulated, students can begin to see how the same strategies apply in different contexts. For example, the strategy of decomposing complex problems into simpler problems takes the form of writing subroutines in computer programming, and proving lemmas in geometry.

**Articulation for other students promotes insight into alternative perspectives.** If students try to explain an idea (or problem) to other students, then they begin to see the idea from the other students' perspectives. If they get responses from other students, they can see what difficulties other students have with the idea and how other people view the same issue (as will occur with the Constitutional Convention mentioned above).

We can illustrate articulation by a computer program called Robot Odyssey developed by the Learning Company. In it, students first learn how to wire robots to behave in different ways when they sense different conditions. The robots have bumpers that detect when objects or walls are encountered on different sides, and sensors that detect the direction of particular objects in the vicinity of the robot. The robots have thrusters that can send them in different directions and grabbers to pick up objects. Students can wire these robots in very complicated patterns to move around, explore, and pick up or avoid different objects. Thus, students can construct robots that behave in complicated ways, and turn them loose in the world to see what happens. The challenge of Robot Odyssey is to use the robots to navigate through a complicated maze, as in an Adventure game, where there are puzzles to solve and enemies to avoid in order to come through successfully. This then is the test of how well students have wired their robots, or more generally how deep their understanding is of how to build intelligent artifacts to deal with different situations.

Wiring robots is just one of many such enterprises. In languages like Logo (Goldenberg & Feurzeig, 1987), students can write programs that, for example, take any noun and generate a plural for it, or transform input sentences in active voice to passive voice. Thus students can articulate their grammatical theories in forms that can be tested out.

In an entirely different vein, Sharples (1980) has developed a Fantasy game kit that allows students to create computer Adventure games for other students to play (where they explore caves looking for treasure, avoid monsters, and try to find the way out). This forces students to articulate their images of different caverns in a series of caves, and pose problems for others that are solvable but challenging.

These examples illustrate some of the different ways computers can be used to foster articulation by students. While articulation is often encouraged by teachers in schools, computational articulation requires even more explicitness.

## Exploration

Exploration involves pushing students to try out different hypotheses, methods and strategies to see their effects. This puts students in control of problem solving, but they need to learn how to explore productively. The computer provides powerful tools that allow students to explore hypotheses and solutions (i.e., problem spaces) faster, so they don't become frustrated.

### Benefits of Exploration

**Learning how to set achievable goals.** Many students set goals for themselves that are either too easily achievable or impossible to achieve (Atkinson, 1964). Studies of successful artists (Getzels & Csikszentmihalyi, 1976) have identified problem finding as the most critical skill for success. And yet very little effort in school goes into teaching students how to set reasonable goals and to revise their goals as they proceed deeper into a problem. Instead school emphasizes giving students well-defined tasks, unlike anything in the real world.

**Learning how to form and test hypotheses.** One of the major goals of inquiry teachers (Collins & Stevens, 1982, 1983) is to teach students how to formulate hypotheses, rules, or theories and then how to test out whether they are correct. Making sense out of the world around us requires these skills, and unless students practice forming and testing hypotheses in many different domains with the help of expert guidance, they will not learn to do so effectively.

**Students will make discoveries on their own.** When students are put into an environment where they are making and testing hypotheses, they get a sense of what it is like to be a scientist. They will feel the joy of generating their own ideas and seeing if they are correct. They may even discover genuinely novel ideas. But even when they come up with old ideas, they will at least sense where the ideas came from and why they are important.

We can illustrate the potential for exploration with the science laboratories developed by TERC (Mokros & Tinker, 1987). In one laboratory students can detect how far a moving object is from a sensor, and plot velocity, distance, and acceleration. This enables them to discover Galileo's laws of falling objects. Or they can conduct experiments with balls hitting other balls to see how changes in direction affect velocity. There are many different phenomena that students can investigate, and with the immediate graphic plotting they can test out many different ideas quickly.

Another example of an exploratory computer-based environment is the economics simulation called Smithtown developed by Shute and Bonar (1986). In Smithtown, students can manipulate different variables, such as the price of coffee, to see how they affect other variables, such as the amount sold. They then are encouraged to try to figure out the various laws that relate the different variables. Thus they are discovering basic economic relationships. Similarly, students with a music or painting program can compose works quickly, and with much less effort. Hence, they can explore many different techniques and see how effective they are. Computers provide powerful tools for exploration (Papert, 1980), and we are just beginning to investigate their potential.

## Conclusion

As technology becomes cheaper, it gives us the capability to realize resource intensive education once again. In particular, technology allows us to return to a kind of apprenticeship centered around modelling, coaching, and fading in situations that reflect the kinds of uses that the knowledge gained might be used. Moreover, technology enables us to create environments where new methods of learning--that is, reflection, articulation, and exploration--are possible. This raises the question of what form education should take, given the capabilities and limitations of the technologies that are developing.

## References

Anderson, J. R., Boyle, C. F., & Reiser, B. J. (1985). Intelligent tutoring systems. *Science, 228,* 456-468.

Anderson, J. R., Boyle, C. F., Farrell, R., & Reiser, B. J. (1984). Cognitive principles in the design of computer tutors. In *Proceedings of the Sixth Annual Conference of the Cognitive Science Society* (pp. 2-9). Boulder: University of Colorado.

Atkinson, J. W. (1964). *An introduction to motivation.* Princeton, NJ: Van Nostrand.

Brown, J. S., Burton, R. R., & deKleer, J. (1982). Pedagogical natural language and knowledge engineering techniques in SOPHIE I, II, and III. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (pp. 227-282). New York: Academic Press.

Burton, R. R. (1982). Diagnosing bugs in a simple procedural skill. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (pp. 157-184). New York: Academic Press.

Burton, R. R., & Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In D. Sleeman & J. S. Brown (Eds.), *Intelligent tutoring systems* (pp. 79-98). New York: Academic Press.

Collins, A. (1986). Teaching reading and writing with personal computers. In J. Orasanu (Ed.), *A decade of reading research: Implications for practice* (pp. 171-187). Hillsdale, NJ: Erlbaum.

Collins, A., & Brown, J. S. (1988). The computer as a tool for learning through reflection. In H. Mandl & A. Lesgold (Eds.), *Learning issues for intelligent tutoring systems.* New York: Springer.

Collins, A., & Brown, J. S. (in press). The computer as a tool for learning through reflection. In H. Mandl & Lesgold (Eds.), *Learning issues for intelligent tutoring systems.* New York: Springer.

Collins, A., Brown, J. S., & Newman, S. (in press). Cognitive apprenticeship: Teaching the craft of reading, writing, and mathematics. In L. B. Resnick (Ed.), *Cognition and instruction: Issues and agendas.* Hillsdale, NJ: Erlbaum.

Collins, A., & Smith, E. E. (1982). Teaching the process of reading comprehension. In D. K. Detterman & R. J. Sternberg (Eds.), *How much and how can intelligence be increased?* (pp. 173-185). Norwood, NJ: Ablex.

Collins, A., & Stevens, A. L. (1982). Goals and strategies of inquiry teachers. In R. Glaser (Ed.), *Advances in instructional psychology* (Vol. 2, pp. 65-119). Hillsdale, NJ: Erlbaum.

Collins, A., & Stevens, A. L. (1983). A cognitive theory of interactive teaching. In C. M. Reigeluth (Ed.), *Instructional design theories and models: An overview* (pp. 247-278). Hillsdale, NJ: Erlbaum.

Dreyfus, H. L., & Dreyfus, S. E. (1986). *Mind over machine: The power of human intuition and expertise in the era of the computer.* New York: Free Press.

Feurzeig, W., & White, B. Y. (1984). *An articulate instructional system for teaching arithmetic procedures.* Cambridge, MA: Bolt Beranek and Newman, Inc.

Getzels, J., & Csikszentminhalyi, M. (1976). *The creative vision: A longitudinal study of problem finding in art.* New York: Wiley.

Goldenberg, E. P., & Feurzeig, W. (1987). *Exploring language with Logo.* Cambridge, MA: MIT Press.

Haertel, H. (1987). *A qualitative approach to electricity.* Palo Alto, CA: Xerox Corporation, Institute for Research on Learning.

Johnson, W. L., & Soloway, E. (1985). PROUST: An Automatic Debugger for Pascal Programs. *Byte, 10,* 179-190.

Lampert, M. (1986). Knowing, doing, and teaching multiplication. *Cognition and Instruction, 3,* 305-342.

Mokros, J. R., & Tinker, R. F. (1987). The impact of microcomputer-based labs on children's ability to interpret graphs. *Journal of Research in Science Teaching, 24,* 369-383.

Palincsar, A. S., & Brown, A. L. (1984). Reciprocal teaching of comprehension-fostering and monitoring activities. *Cognition and Instruction, 1,* 117-175.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* New York: Basic Books.

Scardamalia, M., Bereiter, C., & Steinbach, R. (1984). Teachability of reflective processes in written composition. *Cognitive Science, 8,* 173-190.

Schoenfeld, A. H. (1983). Problem solving in the mathematics curriculum: A report, recommendations and an annotated bibliography. *The Mathematical Association of America*, MAA Notes, No. 1.

Schoenfeld, A. H. (1985). *Mathematical problem solving.* New York: Academic Press.

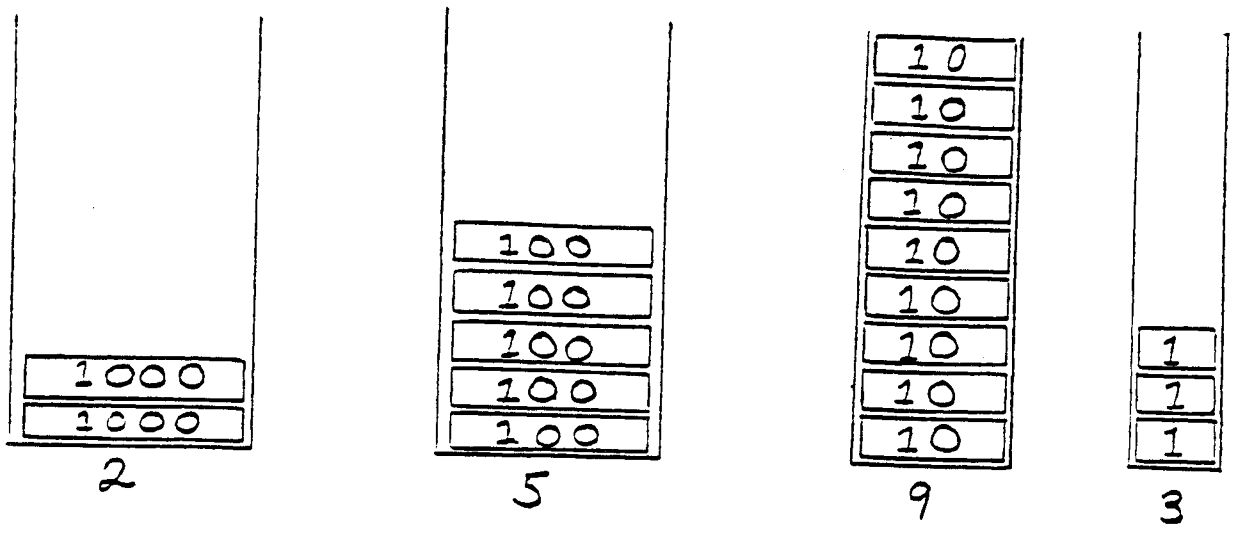
Schoenfeld, A. H. (in press). On mathematics as sense-making: An informal attack on the unfortunate divorce of formal and informal mathematics. In D. N. Perkins, J. Segal, & J. Voss (Eds.), *Informal reasoning and education.* Hillsdale, NJ: Erlbaum.

Sharples, M. (1980). *A computer written language lab* (DAI Working Paper No. 134). Edinburgh: University of Edinburgh, Scotland, Artificial Intelligence Department.

Shute, V., & Bonar, J. (1986). Intelligent tutoring systems for scientific inquiry skills. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society* (pp. 353-370). Hillsdale, NJ: Erlbaum.

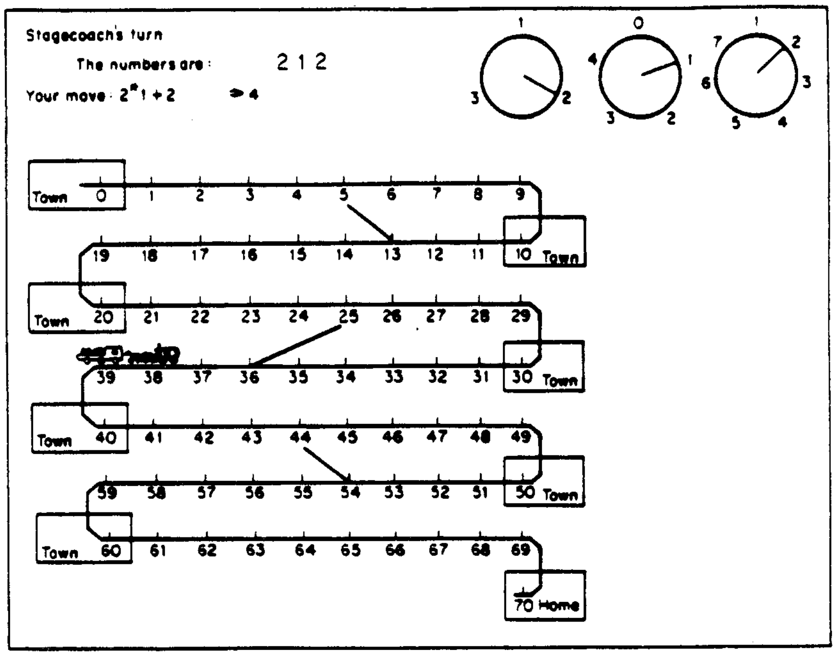
Snyder, T., & Palmer, J. (1986). *In search of the most amazing thing: Children, education, and computers.* Reading, MA: Addison-Wesley.

White, B. Y., & Frederiksen, J. (in press). Progressions of qualitative models as a foundation for intelligent learning environments. *Artificial Intelligence.*



$$\begin{array}{r} 11\phantom{0} \\ 2593 \\ \underline{\phantom{000}9} \\ 2602 \end{array}$$

Figure 1. Bin model representation of 2593 and standard representation of the addition problem 2593 + 9 in Summit (from Feurzeig & White, 1984).



Stagecoach's turn

The numbers are: 2 1 2

Your move: 2*1+2 => 4

Figure 2. Screen used in "How the West was Won" game (from Burton & Brown, 1982).

This page is intentionally blank.