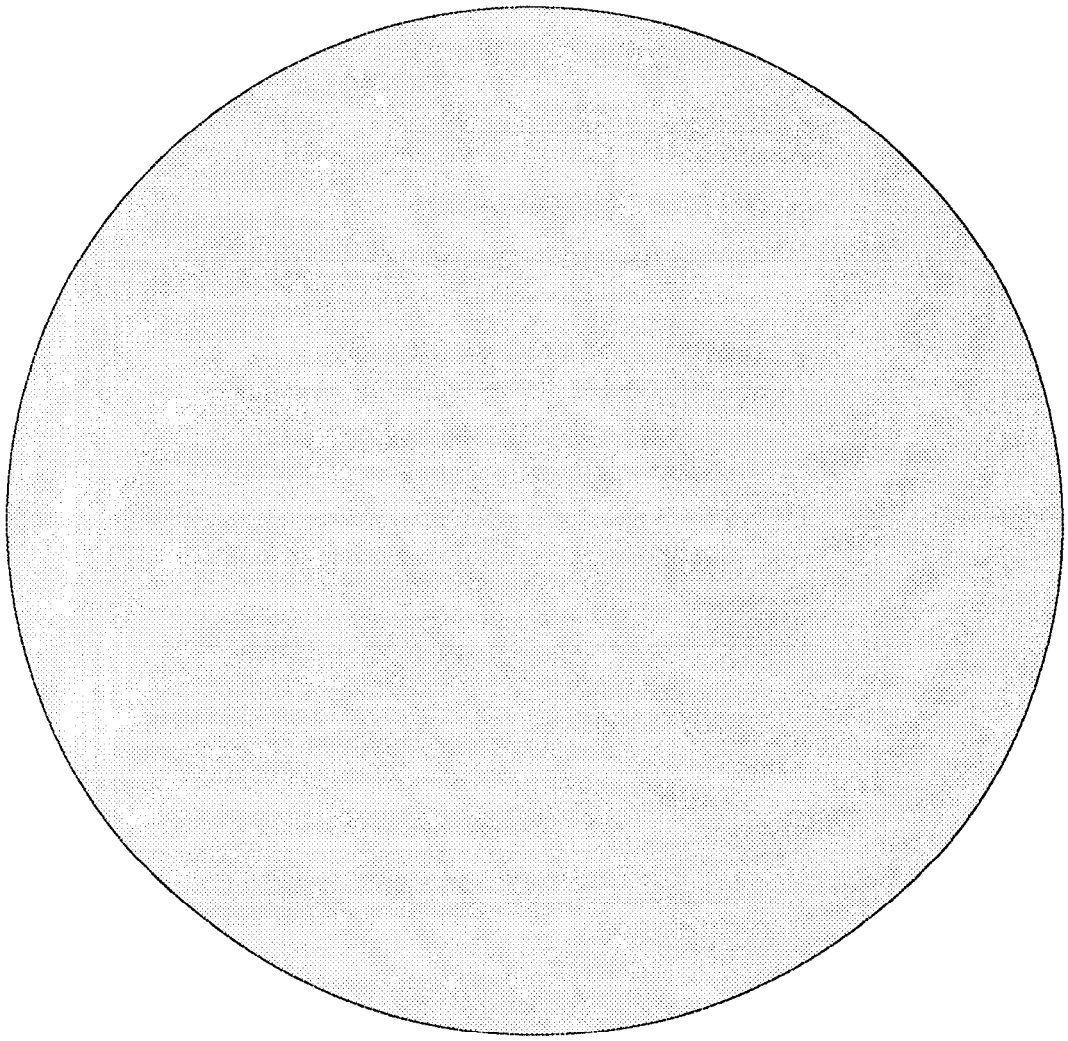
TEACHING ARCHITECTURES

Roger C. Schank

Technical Report #3 • August 1990

The Institute for the Learning Sciences

Northwestern University



Established in 1989 with the support of Andersen Consulting

Northwestern University

The Institute for the Learning Sciences

TEACHING ARCHITECTURES

Technical Report # 3 • August 1990

Roger C. Schank

Established in 1989 with the support of Andersen Consulting

TEACHING ARCHITECTURES

Roger C. Schank

August 1990

The Institute for the Learning Sciences
Northwestern University
Evanston, IL 60201

This work was supported in part by the Department of Defense, specifically by the Advanced Research Projects Agency, the Air Force Office of Scientific Research, and the Office of Naval Research. The Institute for the Learning Sciences was established in 1989 with the support of Andersen Consulting. The Institute receives additional funding from Ameritech (an Institute Partner) and IBM.

Education by computer has almost always entailed a certain explicit teaching architecture, something we can call "the page-turning architecture." This architecture has been implicit in the design of educational software and is responsible for most of the lack of excitement or educational value of the majority of the software on the market today. The page-turning architecture basically involves putting up a screen of information and asking the student to indicate when he wants the next page of information, or to answer a question that will cause him to get another page of information.

The role of the student assumed by this architecture is that of a reader of information or a selector of multiple-choice answers. Clearly, there is a problem here. Such software assumes that learning involves either the passive absorption of information or the recognition of answers to questions. We refer to these models of learning, which I have discussed elsewhere, as the absorption model and the assessment model of instruction. The idea behind these student models is one that is often kept quite implicit, namely, that a student is a kind of sponge. The model assumes that information given to a student is absorbed by the student independent of how interested the student is in the material. This supposed absorption is then tested to assess how much has been absorbed. This, by and large, is our current educational model. Certainly we can do better.

In this paper, I present six teaching architectures for the creation of computer-based learning environments. These architectures assume that learning involves more than reading and answering, or, if reading and answering are involved, that their involvement ought to depend upon preparing the student such that reading or answering is precisely what he wants to do at the moment that the computer causes that to happen. To do this means treating software somewhat differently. For one thing, it entails enabling computers to store and retrieve far more information than has been typical so far. It also entails representing that information, or at least the content of that information, so that it is accessible at the right time for the right reason. In other words, to make machines better instructors, we must give them better information in an accessible form. Further, we must design teaching architectures that make use of such information.

The six architectures presented here are inspired by what we know of how people learn and the information that we believe can be imparted to machines that will enable them to be better instructors. These architectures are, then, artificial intelligence (AI) architectures for the design of teaching systems.

1. Case-Based Teaching Architecture

The first architecture, called the Case-Based Teaching Architecture, depends upon two ideas. The first is that experts are repositories of cases. When a doctor or a lawyer is doing his job, for example, he is likely to rely upon his knowledge of previous cases to help him in current decision making. He might recognize that a situation he is working on is quite like one he has previously encountered. His task, then, is to draw upon the similarities in the case he has been reminded of, in order to help him better understand the new case. He would also want to understand the differences between the cases. This would help him assess just how relevant the case he has thought of really is, and help him think about other cases he has experienced that might be more significant for his work on the current case. This is what we call case-based reasoning. It is clear that much reasoning by experts relies upon a significant case base that can be utilized when necessary.

When experts teach, they often rely on their case base. The second key idea in this architecture is that good teachers are good story tellers. Or, to put this another way, a good teacher will relate a relevant case to a student who would benefit from learning about it. However, the student has to be ready to hear about the case. He has to be in a position, and realize that he is in a position, to use the information that is contained in the story for something he is working on or is interested in. In addition, the case has to be told to him is such a way as to capture and hold his interest. In other words, when he is ready to listen to a good story, he is in a position to learn from that story.

The case-based teaching architecture exploits this basic capacity of a student to learn from stories, and the basic desire of teachers to tell stories that are indicative of their experiences. The premise of this architecture is to place a student in a situation that the student found inherently interesting. Such a situation might involve, for example, having a student attempt to build, design, plan, or create something on the computer. This basically creative task should be one that is appealing for a student, so that there is no problem getting him interested in the task. The task should be complex enough that everything he might need to know is not available to him. The task of the software program is to teach the student what he needs to know, or what he might consider, while doing his task, at precisely the points in the task in which he becomes interested in knowing this information. This information should be presented in the form of stories.

Inherent in the case-based teaching architecture is the idea that learning takes place on a "need to know" basis. Thus, there are two major issues involved in case-based teaching. First is the construction and exploitation of a case base from which the cases can be drawn when they are needed. The second is the creation of a situation that would cause a student to be interested in hearing about a relevant case. The case base must be indexed in such a way as to relate to the situations that are encountered within the situation. The juxtaposition of a case base and a situation that indexes the case base is the essence of case-based teaching.

As an example of how this architecture is actually used, we can discuss two projects currently underway at the Institute for the Learning Sciences. These are the animal design project and the cross-selling project.

The animal design project depends upon the idea that children like to build things, draw things, and, in general, create and fantasize. We asked a number of sixth-grade children to tell us about an animal they might design if they could design any animal they would like. They all had immediate responses. The idea behind the animal design program is that we can take this inherent interest in animals and design and use it as a vehicle for telling good stories about biology. The premise here is that the student is an interested listener if and only if the stories we tell relate to the concerns that the student has at the moment. In other words, if the student is attempting to design something that already exists, that doesn't exist for good reason, that is faulty in its physics, that is unnecessary for an animals survival, and so on, then these are good reasons to talk about these issues.

A great deal of interesting video about animals already exists, but is never told in a way that has urgency for a student. A student who has just decided to build a giraffe that flies, however, might well want to know the use to which the giraffe puts its long neck, or how birds get off the ground. In other words, a mutual design task enables good story telling to occur in a context relevant to the student. Doing this implies, though, that no one set of animal stories are the right set to know. In other words, teaching biology means talking about function and evolution and not about one particular animal and not another. Each student would hear different stories in this model, a point which frustrates traditional assessment people, but which otherwise is typical of real life.

A more business-oriented version of this same phenomenon occurs in the cross-selling project at the Institute. Imagine a company that has many products to sell, yet no

one salesman is an expert at selling all of them. When a selling opportunity arises for a product that a salesman knows how to sell, the opportunity to sell a different, less familiar product, might also have arisen. But, how would the salesman recognize this?

If he had a program available to him that had captured the knowledge of many salesmen about many products, and he were able to describe the sales situation to the computer, then we would have another instance of the use of case-based teaching. Each salesman who is a specialist in a product can be seen as a story teller, teaching how to sell that product and how to notice when a selling opportunity has arisen. A program with fifty experts on video, each embodied by a system of rules that identify when that video should appear, would create the basis of a useful cross-selling program. What would be included on the video? The video would contain stories about selling a given product, advice about how to sell that product, information on the product itself, and the name and phone number of the expert so he could be contacted for help.

This is an unusual kind of teaching. Students only learn when they have the need for the information that is taught. This kind of teaching corresponds to how learning actually takes place. Learning occurs when students really want to know what someone has to teach, because knowing that information will be useful in a way that is readily apparent to the student. This is the essence of the Case-Based Teaching Architecture.

2. Incidental Learning Architecture

Not everything is fun to learn. In fact, some things are terribly boring to learn. And, when some curriculum inventor decides that such things must be mastered, he is likely to try to teach them as a list to be memorized, or a set of items to be tested. But, people do habitually learn a variety of information that is quite dull, without being completely bored by it. They do this without intending to learn the information at all. For example, most Americans know the fifty states that comprise the United States. They know them well enough so that it is a rare American who, when confronted with the word "Utah" or "Tennessee", cannot identify it as a state. On the other hand, most Americans cannot name the fifty states. They may be able to name forty-seven, but, in general, it is very difficult to remember each and every one. People do not have this problem with the alphabet, on the other hand. Most everyone can name every letter. The difference here is not one of length of the list, nor is it attributable to frequency of use. The explanation is

quite simple. Everyone was taught to memorize the alphabet, but almost no one is forced to memorize a list of the states.

It might seem that the natural conclusion from these observations is that memorization works, as indeed it does if your goal is to make sure that the memorizer can recite a list back to you. On the other hand, if your goal is useful information, memorization isn't the issue at all. Rather, we must ask the question, "How did all these states come to be recognizable entities?"

Much of what we know we have simply learned "in passing." That is, we have accumulated knowledge through the act of living and doing things which included situations in which that knowledge came up and was of use. We may know something about Iowa, for example, because we read a book that took place there, or we knew someone from Iowa, or because we like to watch college football. Our knowledge comes from experience with a subject and thus is scattered around memory, stored with those experiences. We may find it difficult to retrieve a set of facts about Iowa when asked to produce them, but we may find that the information comes up in a natural way when we need it.

Much of what we currently learn in school involves presenting information in a manner that is quite different than the natural way. We have lessons in a subject wherein we expect students to learn lists of items, or sets of facts. The method we use is to tell them these facts and hope they remember them. But, we can much more reasonably expect students to remember facts that they gathered for some use, for some real purpose in which they were interested.

The task, then, is to design software that will present students with situations that are inherently interesting. Then, we can use those situations to teach certain information by causing the interesting situations to present themselves only if the desired material is learned. Obviously, this method can be misused. We could require students to recite the alphabet in order to watch TV. This might work to a limited degree, but it is very important to keep in mind that the extent to which the material to be learned and the reward material are naturally intertwined strongly relates to the ability to hold a student's interest.

The Incidental Learning Architecture is based on the creation of tasks whose end result is inherently interesting, and which can be used to impart dull information. There are

likely to be many sub-architectures involved here. In other words, there are many methods of doing this. Each method would serve as a standard software tool into which relevant material could be put to create new learning situations.

The key to exploiting incidental learning is to find things that are inherently fun to do on a computer. This could be any good video game, for example. The next part is trickier. What the student naturally wants to learn in the video game ought to be worth learning. The goal, then, is to change the skills to be learned from hand-eye coordination tasks to content-based tasks, where one needs to know real information in order to accomplish one's goal on the computer. This will work very well if there is a natural correlation between the content-based tasks and what is inherently fun. One should not have to learn verb conjugations in order to be allowed to throw darts in a computer game.

One of the best examples of this I have encountered was during an unusual journalism class I attended at Northwestern University. This class was taught by a man who came from the news division of a major television network. This man was trying to teach students how to put together the evening news. To do this, he created three teams of students and gave them five hours to watch the wire feeds and plan the broadcast. At the end of the process of deciding what would make up the news program, they have to write what the anchors would say, find the accompanying video, block out the time, and then actually do the taping of the news show.

Needless to say, the students were thoroughly entranced by this task. Making the television news captured their interest totally. Students line up to get into this class. But, what does this class teach? For the professor, it teaches what he wants it to teach; that is, how to put together the evening news. The professor advises the students on what they are doing, on what matters, on what mistakes they are about to make, and helps them evaluate the finished product. Clearly, he is teaching an important aspect of journalism.

But, he is teaching something else as well. In order to put together the evening news effectively, you must know something about history, current events, what the public cares about, what the public ought to care about, and what people should know in order to live their lives. In short, the professor is not only teaching journalism, he is teaching quite a bit more.

Now, suppose we took the journalism element out of the picture. Putting together the evening news may be fun, but not that many of us need this skill. On the other hand, this may be a good way to teach current events, history, and social issues. A very dry subject could be made very real, if the goal of putting together the evening news drove the learning of current events. This course, if used as a method of teaching social studies in high school, could be every bit as exciting to high school students as it is to Northwestern journalism students. The important premise behind the Incidental Learning Architecture is that when a student is doing something that is fun, that student can be learning a great deal without noticing it. Learning does not have to be jammed down students' throats.

3. Cascaded Problem Sets

We tend to teach problem solving by giving students rules to apply and problematic situations in which those rules are applicable. Then, we increase the complexity of the problem and hope that the student can adapt. Often such material is presented in a workbook format, with text that is doing the teaching and problems that test what the text was intended to teach.

This format for teaching is ubiquitous in the schools and in training situations in business. It has many inherent problems. One is that students are usually required to do every example, since each one leads to the next. A second is that there is no remediation if a student has failed to grasp an idea in time for the problem that tests it. Often, the material proceeds too slowly for the taste of the student, with textual material that is dull and irrelevant to the problems to be solved. All in all, this is a boring way to teach, and most students object.

Given the opportunity, most students will simply go to the back of these books, and attempt to answer the last problem. This is the most difficult problem, if indeed the problems have built upon each other. If they fail to answer that problem, they will read as much of the book as they need, and in any order, to answer the problem.

In general, school is much too problem-oriented, much too concerned with getting a student to learn the one and only way to do things that enables him to repeat properly

certified right answers. But, there can be no doubt that under certain circumstances there are right answers, as well as skills for determining those answers, that are necessary to obtain. When this is the case, the teaching problem is to present the necessary information to each student as he needs to know it, when he wants it, and in response to what he has been doing so far. This is the point of the Cascaded Problem Sets architecture.

In this architecture, the idea is to build a problem space whereby each problem relates to each other problem with respect to the extra layer of complexity of a certain sort that it entails. In other words, "if you can't solve problem A, you certainly can't solve problem B" means that B is logically above A. Between A and B would be some information that B entails that A does not. Below A would be something simpler than A that perhaps does not entail the knowledge common to A and B. As a student has trouble with one problem, he moves down the cascade of problems by learning about the issues that one would need to know to solve the problem he was having trouble with. The task of the "tutor" in the software is to determine how to guide his motion down the path of problems such that he is being presented with problems that address the issues he needs to deal with in order to move up the cascade.

Teaching, when it occurs, happens in response to a student's request to help him with something he doesn't understand. In this way, learning is student-directed and relevant to a particular student's needs. This architecture would be of use when there are skills to learn that depend upon skills that should have been learned before.

In order to exploit this architecture, it is necessary to build libraries of cascades and to determine the content-based connections that relate one problem to another. To do this, a problem must be decomposed into its constituent parts. Each constituent would itself be a problem, and it too would have constituent parts. This cascade of problems would continue until a basic level of knowledge is reached that, it is assumed, any user of the cascade would have. For example, at the bottom of a cascade of algebra problems would be basic arithmetic. Determining how one problem is related to another is the major issue in the construction of cascaded problem sets. Exploiting this architecture means analyzing a domain of knowledge well enough so that the cascade can be built and the connections can be established.

4. Directed Exploration of Video Data Base Connections

Learning depends upon good information being available at the time one is ready to hear it. To make computers truly useful information providers we must create learning environments so that each user can follow his or her own interests. Artificial Intelligence is concerned with building these knowledgeable machines. We need to build a machine that holds as much knowledge as a television set, except that the control is in the hands of the user. TVs communicate only what their programmers tell them to say. Successful completion of our Video Data Base project would enable TVs to be entirely interactive, awaiting the command of the user before continuing on. The computer would be able to assess what the user wanted, based on the nature of what he had seen so far, and by understanding what the user has requested. The job of the computer would be to find, from vast stores of video information, what it deemed most relevant to the user, thereby linking what the user needs to know and what the computer/TV knows to say. This conversational model is quite similar to that used by expert humans.

Interactive computers have always been seen, ultimately, as conversational engines, programs to which one could type English sentences and get answers. But, three serious problems exist with this imagined technology. First, the natural language problem is difficult enough in limited domains. In unrestricted domains, where real knowledge is necessary for linguistic analysis, the problem is extremely complex. Second, theories of conversation have often revolved around rules having to do with figuring out the intention of an utterance, and other pragmatic issues. Recent research at our lab has indicated that, in conversations, people choose what they will say from a storehouse of things they have already said at some previous time. Most experts wind up saying the same thing over and over again rather than reconstructing their ideas from first principles each time. Third, computers have to have something to say in order for a user to want to talk to them. This is usually not the case.

Usable conversational systems ought to have three modes of access. The first is direct inquiry, with the computer understanding any input in the user's language. Another method of access would be by visual navigation. If information is properly organized, then when viewing one piece of information, related topics would be readily available, so that a user could continue a natural conversation through the dynamic nature of the data base. The third method of access is "button-based." The creation of a standard set of "buttons" (for example, "WHY?" "WHAT NEXT?" "CONFUSING," and others) would allow the

student state to be understood by the computer, with the same buttons interpreted in different ways depending on the context and timing of when it was pressed.

Buttons are an attempt to move beyond the natural language parsing problem by providing a standard set of query types, anticipated during the construction of the data base. In other words, if we know that when a speaker makes an assertion, he might be asked WHY, or told that he is BORING, we would anticipate this in the original video taping and ask relevant questions.

In teaching today, it is currently quite common to give assignments to students. Students are asked to write reports by going to the library and researching a topic. The concept here is a good one. Let students discover what they might find interesting and then have them organize what they find into a coherent report. Of course, students are not always allowed sufficient freedom to insure that the task is always interesting to them. Often they are told what books to read, or, alternatively, they cannot easily find what might be of most interest to them. Also, the topics they pursue are frequently assigned to them, regardless of their interests, as if it were the content and not the act of discovery and reporting that were being taught.

This method of instruction has its flaws, but it can be used quite effectively if it is employed correctly. Even more effective, however, would be the creation of video data bases that were easily explorable. There is a tremendous amount of information available on video. Go to any video rental store and you can see what is commercially available to the general public. But, beyond movies and workout tapes, there exists, in the archives of television networks, for example, a tremendous amount of video footage. Consider the existing video of the important news event of the world in the last thirty years, interviews with important leaders in every field, studies of animals in the wild, instructional material, and so on. Imagine that this material was available to any student who wished to view it. Children who have grown up on television are more receptive to video than to print. And, while print authors can analyze a subject very effectively, nothing compares to seeing it for yourself.

The problem here is one of information organization. It really doesn't help anybody to make thousand of hours of video available without an organization scheme that would allow a user to find what he wanted instantly. Further, this information is of no use if it cannot tell the user, at precisely the moment when he might be interested, what other related

information might be available. After all, standard book libraries are, in essence, giant explorable data bases. Indeed, any textbook is itself a set of information, organized in a left-to-right fashion, or by a key concept index in the back of the book. Both of these schemes fail to work as effectively as they might because, any good piece of information causes questions to come to mind that make the reader want to know more. There are many possible questions that an individual might ask after hearing someone say something or after reading something.

In other words, video data bases would be valuable things if they were indexed in such a way as to present information organized by the content of that information, always pointing the way towards additional information that is similar in content, where similar is defined in a variety of different ways. To do this effectively, the concept of a "show," of an hour long set of video clips, is irrelevant. It is the clips that comprise such shows that must be organized so that material, for example, on "man's inhumanity to man" that appears in thousands of shows or news programs, would be available by searching on various subtopics within that theme.

Any scheme of this sort would cause any one video clip to have multiple indices, since it could relate to many possible points. It is clear that the indexing problem would cause the creation of such video data bases to be a tremendous undertaking. Nevertheless, the rewards of such an undertaking, if properly executed, would be tremendous, not only for motivating students to find out about the world, but as an archive of past history that would be useful for many different purposes.

The idea behind the Directed Exploration of Video Data bases Architecture is that it should be possible to ask these questions and have the answer be the very next piece of video information that is presented by the system. This is possible if the information that relates to that question has been determined and collected, and if it has been indexed in a such way so that it can found at the right point. The key problem, then, is anticipating how information might provoke questions, making sure that those questions have been answered and that the answers are readily available. This requires having indexed the answers such that questions and indexes are one in the same and such that users can readily understand what questions can be asked.

Ultimately, we must create a video data base of the important stories of the decision makers of America. To accomplish this, we must make use of the members our society

who have important things to say, including those who can provide corporate memory, who are experts on a given subject, and who have had important experiences that decision makers ought to be aware of. In a society with the breadth and depth of knowledge of the U.S., it is important to make that knowledge accessible to those who need it.

In order to create this data base, we have begun to interview experts who are good story tellers in a given domain, and videotape them telling stories in a casual setting. The interviewers, who do not appear on camera, are professionals in a variety of subjects, including psychology, artificial intelligence, education, and certain specific subject matter experts. Subjects are asked to tell stories of three basic types: stories that illustrate points in their fields that they wish to teach; stories about their own professional lives; and stories that would help users understand technical issues that are raised in the first two types of stories. These stories illustrate problems, principles, events, facts, or opinions that comprise the core of what great teachers know and want to communicate. In addition, some stories are simply fun to hear and can be used to help motivate learning in domains where it is difficult to find good stories.

The collected stories are then classified on the basis of their content, their points, the lessons to be learned from them, the occasions in which they should be told, and their relationship to other stories in the archive. This classification is implemented in a computer program and made accessible in a variety of ways. The major technical problem in this process is one of content-based indexing. In order to tell a story at the proper time one of two things needs to happen. Either the computer must have assessed the situation well enough to have understood that a given user needs to be told a certain story, or the user must naturally "run into" the proper story in the natural course of what he is doing. Each video clip must be labelled by multiple indices that indicate the content, intent, use, one or more points, and place in context of the given story. Work has begun on doing this with various subject matter experts utilizing the Universal Indexing Frame. Research is continuing on these three technical problems: the content-based indices, button-based indices, and next-neighbor indices.

5. Simulation-Based Learn-by-Doing

There is really only one way to learn something that is a "doable" thing, and that is to do it. If you want to learn to throw a football, drive a car, build a mousetrap, design a building, teach students, or be a management consultant, you must simply learn by doing it. There is, of course, a problem here. Many of these tasks cannot be learned except by doing them. We hand a child a ball and teach him to throw, without much risk. If he throws poorly, he simply tries again. Parents are often the teachers, for example, sitting in the passenger seat nervously while the teenager tries out the driver's seat for the first time.

However, instead of allowing students to learn by doing, we often create courses of instruction that tell students about the task in a theoretical way, without concentrating on the doing of the task. To put this another way, when apprenticeship will not work all that easily, or is risky, we try lecturing.

The teaching architecture implied by all this is to change every possible skill into a learn-by-doing situation. To do this, one must create simulations that effectively mimic the real life situation so well that the student is prepared for real life situations without having to be in them. The air flight simulator has been effectively employed in the training of pilots, in part because there was money available to build such simulators, and because the risk of injury in training by the more normal learn-by-doing method was quite high.

Simulations of all kinds can be built. What is required is to understand the situation to be simulated well enough so that the simulations will be accurate portrayals. This can mean, in the case of simulations of people-to-people interactions, having to create complex models of human institutions and human planning and emotional behavior. These simulations would have to be built such that roles are available for the student to play within the simulation. Thus, to learn to be loan officer in a bank, for example, the student would play the role of loan officer in the simulation, trying out new situations that would cause him to have simulated experiences analogous to those he might encounter in the real world.

To build such learn-by-doing environments properly requires least four basic components: the simulator itself, a teaching program that helps the student through the simulator and can discuss issues with the student, a language understanding program that

can understand what a student might ask the computer teacher, and a story-telling program, activated by the teacher at appropriate times, that would tell stories from the experiences of experts in actual situations.

As an example, consider the teaching of physics in high school. Most of the time, students listen to lectures, or watch teachers perform so-called experiments which always turn out as expected. Students memorize formulas and try to apply them. They learn that science is a bunch of facts to be memorized and that experiments are exercises in report-writing in a tedious format.

To scientists, science is fun. The intent of a high school physics course should be to convey the joy of discovery, the excitement of speculation followed by confirmation. It is difficult to have this happen without allowing students to discover things for themselves. Creating simulations of the physical world that allow for and encourage experimentation is one good way of teaching physics. But, this would not work well without a good physics teacher who can advise on things to try and explain situations that did not work out as expected. We have been attempting to write such a program that will enable high school students to perform physics experiments that involve, for example, high energy particle accelerators. Why shouldn't a student enjoy smashing atoms? It sounds better than a video game. With the help of physicists who do this kind of work, we can build simulators of real physics laboratories and use the stories of real physicists to make suggestions and convey the excitement. They can also do the instruction on an as-needed basis. We can turn the whole idea of what it means to learn and what it means to teach completely around by changing reading about something someone else did into learning to do something that only experts do.

The Simulation-Based Learn-by-Doing Architecture is critical when the subject matter to be learned is one that is really experiential at heart. Much of natural learning is the accumulation of experience. Schools have trouble allowing children to learn from experience both because their individual experiences are so different and because the classroom situation does not allow for much actual experience to occur. Nevertheless, experience is the best teacher.

6. Responsive Questions

We can, of course, be our own teachers. Sometimes we have a good idea, or a problem, or just an issue, that we would like to discuss. When we find someone to talk to, that person may not be the best listener in the world. They might want to tell us what to do, rather than just listen to us. A good listener is also a good provider of questions. People who are creative know how to provide those questions to themselves. They know how to both have an idea and constructively criticize it at the same time.

The Responsive Questions teaching architecture is intended to provide good questions to someone who is thinking about an idea or a problem. This is a very important part of teaching because it teaches, among other things, reliance on one's own ideas. In the classroom, such teaching is virtually impossible to find, since the one-on-thirty nature of classroom instruction seriously inhibits a teacher's ability to follow the reasoning of any one student .

The problem is to create software that has good questions in it and knows when to ask those questions. The questions need not be too specific. The idea is simply to know how to provide a thought-provoking question rather than a thought-stifling one. Such a program would be quite useful to talk to, even if it never "understood" what you were saying.

Students need to be able to try out their ideas without having those ideas judged. This is very difficult to do in a classroom. Other students snicker, argue, and ridicule when students talk. Teachers can only pay attention to one student at a time. In a classroom, the race goes to the most confident and risk-free idea.

Much of what a student needs to hear about an idea that he has had is a question that will make him elaborate upon that idea in some way. Such questions are easy to ask if one has the patience. The more of an expert one is on a given subject, the more one knows what questions to ask. Experts know questions rather than answers. Why not make those questions available to students who are just beginning to encounter these areas of expertise? The Responsive Questions Architecture involves allowing a student to speculate, and then responding to that speculation from a library of questions pertinent to the subject matter and to the particular point the student is considering. In this way, students can teach themselves. Students enjoy thinking.

Conclusion

The work that has been done in computer-based training and educational software is very rarely extremely impressive. There are many intelligent people working in these areas, but their work is inhibited by the software architecture and the concept of what it means to be educated. The one available software architecture is very limited. This architecture concretizes a feature of education that is actually quite awful; namely, that being educated means knowing lots of answers. More and more in our society we are indundated with all the facts that students don't know, their cultural illiteracy, their mathematical illiteracy and just plain illiteracy. However right these assessments may be, the solution to these problems is not making students memorize more answers. Education is not memorization. Knowledge is not a list of answers.

Educational software has not helped change this answer-based view of eduaction because that was not its intent, nor was it possible using current computer software methods. We must change what computers can do and then we must change the teaching that can be done with computers. Curiously, it is the second that will drive the first. Once we understand what education by computer could mean, it will be possible to design new types of software that can facilitate that kind of education.

To make possible the architectures I have talked about here does not require major advances in computer technlogy. Much of what I am talking about can be done today, or, if not today, then in a near tomorrow. There are many AI problems that need to be solved to create these architectures to be sure, but not all the AI problems that there are need to be solved. The most difficult aspect of what needs to be done here is not the AI part but the education part. The public needs to be re-educated to understand that standardized tests that emphasize competency also emphasize memorization of answers; that learning can be and ought to be fun; and that computers, because that can be one-on-one instructors are the wave of the educational future. Making these all happen means looking at both education and computers in a new way.