Tools to Assist Learning by Doing

Achieving and Assessing Efficient Technology for Learning

Alan Lesgold and Martin Nahemow
Learning Research and Development Center
University of Pittsburgh

August 29, 2005

Abstract

Learning by doing is a central way in which people acquire substantial expertise. It offers a number of advantages over other learning approaches. Most important, it avoids many aspects of the "inert knowledge" problem. Because principles are acquired through experience, the terms used to state those principles have clear referential meaning, at least within the scope of that experience. Learning by doing also affords opportunities for learning how to manage the full complexity of a real domain, at least when the learning tasks are of the same difficulty and complexity as the harder tasks posed in real life. In this chapter, I explore some of the tools and approaches that allow learning by doing to be a powerful scheme for education and training. I also discuss some specific difficulties in developing learning by doing technology.

Table of Contents

Introduction

After a few decades, we now have many important foundations established for a cognitive science of learning. Much remains to be done, of course, and many of those attending this meeting are leaders in the refinement and expansion of this science. Personally, I count myself more a cognitive engineer. That is, I use the accumulated understanding and principles from the cognitive science of instruction to address major design problems for the worlds of schooling and training. Like any engineer in a university faculty, I do research, but of a different form. Engineering research is generally driven by real world problems of complex scale. Experiments are conducted and design principles are tested. Often though, the scale of human enterprises prevents the same kind of reductionist single-factor validation that is possible in the laboratory. Rather, efforts are made to demonstrate the efficacy of particular design approaches and to understand the range of contexts in which those approaches can be effectively, efficiently, and safely used.

Consider, for example, the engineering of bridges. Clearly, this is an enterprise that must be informed by materials science and mechanics. However, there will never be a universal answer to such questions as "which is better, a suspension bridge or double-lenticular truss bridge." Rather, the conditions under which each is safe and effective are established, approximately, and trade-off studies are conducted to determine the range of situations in which one design is favored over the other. For a substantial body of cases, there is no research-based answer to the question, leaving the choice as one to be driven more by aesthetics than by science.

I believe that the same state of affairs exists in education. We have multiple methods of learning that can be engineered, including learning by doing, didactic, peer collaboration and argumentation, drill, etc. Each of these methods has costs and benefits that determine whether it is reasonable to use them in specific circumstances. For example, many simple arithmetic skills, like single-digit addition, probably benefit from a bit of drill, while drill is probably not the way to understand electricity and magnetism. Similarly, while a few work situations can best be addressed through brief didactic, many others are better addressed through learning by doing. The job of the cognitive instructional research engineer is to design new instructional approaches and to establish something of their effectiveness, efficiency, and safety. In this paper, I discuss some of the effectiveness and efficiency concerns that arise in developing and applying learning by doing as a training strategy. Safety is gained largely by using simulated rather than real work environments.

Learning by Doing

Learning by doing is a central way in which people acquire substantial expertise. It offers a number of advantages over other learning approaches. Most important, it avoids many aspects of the "inert knowledge"[1] problem. Because principles are acquired through experience, the terms used to state those principles have clear referential meaning, at least within the scope of that experience. Learning by doing also affords opportunities for learning how to manage the full complexity of a real domain, at least when the learning tasks are of the same difficulty and complexity as the harder tasks posed in real life. In this chapter, I explore some of the tools and approaches that allow learning by doing to be a powerful scheme for education and training. First, though, I need to say a little about why learning by doing schemes remain hard to develop.

Three Levels of Facility

Learning by doing facilitates development of all three levels of knowledge that Rasmussen (Rasmussen, Petjersen, & Goodstein, 1994) proposed. Rasmussen, reflecting on the different ways that technicians solve problems in their work, suggested that three levels of knowledge can be involved in expert performances, as listed in Table 1. First, there are highly overlearned skills, generally direct connections between recognition of a situation and an action sequence keyed to that situation. Second, there is an intermediate level of performance driven by cognitive rules. That is, action is not reflexive and immediate but is guaranteed as a set of productions executes. Finally, at the fringes of expertise, performance is driven by conceptual knowledge from which an expert can often infer a course of action using "weak" methods.

---

[1] A term used by Alfred North Whitehead (1929). He used the term mainly in the context of not having sufficient connections between ideas or sufficient depth to ideas, but I use it more to denote specifically the situation in which the referential meaning of a term has not been acquired fully. I believe that I am addressing the same concern as Whitehead, though he might well have argued that a community school, in which teachers and students share a body of common experiences, can reasonably include considerable didactic grounded in those experiences. Conceivably, it is the pluralism of experience and background that makes learning by doing more important than it might have been in the past.

Table 1. Rasmussen's three types of knowledge.

Knowledge Type	Description	Trigger
Skills	Direct connections between perceptual recognitions and specific cognitive or behavioral actions	Perceptual Features
Rule-Based Knowledge	Specific Productions for Expert Routine	Skills failure
Conceptual Knowledge	Conceptual understanding of the domain plus mappings of this knowledge to models of domain problem situations	Rules failure plus connections between problem manifestations and relevant understanding

It is important to note that there is not really an extended body of laboratory research confirming that the three levels identified by Rasmussen involve different cognitive or brain processes, though there are hints here and there of multiple learning and performance mechanisms. The claim I make is that attention to these three different aspects of competence is essential for any kind of training or education that aims at high levels of performance and adaptive flexibility. It is quite possible to construct a cognitive theory that has only a single formalism for representing knowledge, including both conceptual and procedural knowledge (cf. Hunt & Lansman, 1986). But, such a unified model would still need some tools to help instructional designers determine what knowledge needs to be captured for a given training task, and that knowledge is usefully considered to have multiple levels or forms.

This multilevel character for technical expertise presents several major problems for those who build learning-by-doing systems. Cognitive task analysis methods tend to involve a mixture of demonstration by an expert of problem solving activity and commentary by that expert on such performances. For any given problem that is solved, a given expert will use some combination of knowledge at Rasmussen's three levels. That combination may not – indeed probably will not – be the same as we would see in a trainee. For example, an expert might know a number of explicit rules that permit a rapid solution of a problem and may even recognize exactly what do to (equivalently, have a compiled rule that is directly triggered by recognition and does a major piece of the complete task). It is more likely that a trainee will not know exactly what to do and will therefore need support at the conceptual level and possibly even guidance in making inferences from conceptual understanding of the problem situation. In other cases, an expert may provide commentary that is conceptual even though neither experts nor novices actually use the conceptual knowledge in their performances.

One alternative is to focus completely on the middle, rule-based level of knowledge. This, in essence, is how the model-tracing approach of Anderson (Anderson & Corbett, 1993) works. The problem with this approach is that often the real goals of training are not only to teach explicit algorithms for specific problems but also to expand conceptual understanding and to provide practice in the conceptually-driven level of problem solving, i.e., to teach for transfer. This, in turn, requires that the cognitive task analysis that will drive a training system must focus heavily on the conceptual level.

One peril of such an analytic approach is that there is no limit to the potentially relevant conceptual knowledge an expert will cite when asked to state what one needs to know to understand how to work in a domain. The history of both schooling and training is replete with examples of excessive theory being taught in the absence of relevant tasks requiring the theory and also drill on procedures not anchored to theory, as Whitehead (1929) observed. This happens because experts generally do not know explicitly how their conceptual understanding is linked to their practical (rule-based) knowledge (Chi, Glaser, & Farr, 1988). In order to make some progress in overcoming this problem, we need to have a sense of what the learning goals for a training course should be, a sense of how to extract the needed knowledge to support those goals from subject-matter experts, and a sense of how to use that extracted knowledge to build a strong training system.

Knowing a Fixed Algorithm versus Knowing a Generative Algorithm

Having rejected the idea of teaching fixed rule sets of fixed algorithms,[2] we must consider what we do want to teach. One way of thinking about this is to imagine a generative algorithm that involves a combination of specific rule-based activity with reasoning from conceptual

> "…links between concepts that are applied together in the diagnosis, explanation, or treatment of a patient are strengthened, leading to clusters of concepts that are closely linked; the encapsulating concepts that summarize the knowledge in these clusters are, in fact, no more and no less than concepts that have acquired a central status in such a cluster, since they are directly and strongly linked to sets of clinical features, other encapsulating concepts, and causes of disease. …routine cases activate a path of encapsulating concepts between signs and symptoms and diagnosis. In more difficult cases, such a path cannot be set up instantaneously, hence spreading activation activates the underlying more detailed concepts leading to a reasoning process in which biomedical concepts play a role" (Boshuizen & van de Wiel, 1999).

[2] While this rejection extends to the basic model-tracing approach of Anderson and his colleagues (Anderson & Corbett, 1993), it is important to note that the relevant transfer studies to support such a rejection have not been performed. It is quite possible that acquisition of certain rule systems results in a more general ability to transfer to tasks those systems cannot quite handle by themselves. We do have at least "proof of concept" data that the approach we describe in this chapter will yield such transfer (Gott & Lesgold, in press).

knowledge and problem-specific data. As we see it, such a generative algorithm must include schemes for representing a problem, conceptual knowledge sufficient to drive the inference of problem solution steps from the specific details of the problem, and collections of rules for carrying out parts of the problem solution for which fixed algorithms have already been acquired. Our purpose then can be recast as learning how to teach generative algorithms for large classes of problems to trainees, including the knowledge needed to drive such training as well as the methods to be used.

Problems of Multiple Representation Levels

There are a few problems in specifying the multiple knowledge levels needed for expert performance. First, there is no guarantee that the terms used and the forms of encoding for a given object will be the same in each level. Often, the engineering knowledge behind a particular system is a mixture of many different forms. For example, in work we are doing training people who repair ion beam implant devices, we see signs of competence that can be related to various snippets of knowledge from quantum physics, magnetics, optics, and physical chemistry, and it is not clear that the technicians really think about the systems in exactly the terms of any of those domains of knowledge, though their knowledge bears family resemblance to aspects of each.

Further, most technicians have gaps in any one form of knowledge that could support expert performance. They do well precisely because their knowledge, while fragmentary from any one viewpoint, is redundant with respect to the demands made upon it. This makes knowledge engineering extremely difficult. Even if every technician in a particular job had the same fragmentary knowledge, the same gaps, and the same redundancies, it is never as easy to teach partly incoherent knowledge as it is to use it once it happens to be acquired. Somehow, a means must be found for building sufficient but flexible competence in technical jobs without forcing the prior acquisition of all the relevant areas of conceptual understanding as a prerequisite to having a particular job.[3]

An additional problem is the lack of certainty that the constructs of one body of knowledge will translate easily into those of another. Consider medicine, for example. Diseases arise and are defined by their symptoms. Over time, the medical world learns associations between particular symptoms and particular diseases. While all this is happening clinically, medical scientists are also trying to understand the mechanisms of a

---

[3] We fully value deep understanding and the teaching of core concepts in ways that achieve completeness and coherence of understanding. However, it may not be practical to insist on this conceptual completeness as a minimal requirement for a job.

particular disease. Sometimes, it can occur that a diagnostic rule or a treatment rule can develop completely independent of any substantial underlying conceptual understanding. An example of this might be seen in the development of aspirin. At first, certain natural substances were associated with pain relief. Then, the chemistry of salicylates was understood well enough to begin refining acetylsalicylic acid. Then, molecular manipulations were performed in hopes of producing compounds that might be more effective, still without complete knowledge of how aspirin-type drugs worked. Only very late in the process has the mechanism of aspirin become understood. In turn, this has provided a more coherent understanding of why some chemicals work better than others. However, that knowledge is generally in a language that many health care workers refer to minimally if at all, even though they dispense a lot of aspirin.

A Hypothesis

One means for dealing with some of these problems has been suggested by Boshuizen and van de Wiel (1999). They argue, based upon earlier work by Boshuizen and Schmidt (1992), that experience with a specific problem (in their work, a medical case) provides an opportunity for linking the various bits of knowledge that are related to the solution of that problem. This would include both specific rules and various fragments of conceptual knowledge that might come up during problem solution, including comments from tutors, ideas considered and abandoned, etc. Further, to some extent the problem itself provides a partial translation among the different languages within which the different kinds of knowledge are encoded.

> "A critical body of knowledge needed for effective performance of complex tasks – and hence for coaching of that performance – is the collection of connections between rule-based and conceptually-driven representations of the task domain. …repeated applications of biomedical knowledge in clinical reasoning at the earlier stages of development toward medical expertise result[s] in the subsumption of lower level, detailed [biomedical] propositions under higher level, sometimes clinical propositions (Boshuizen & Schmidt, 1992)."

The Encapsulation (Schematization) Process

A general plan for encapsulating or schematizing knowledge from cases seems to have evolved in the cognitive apprenticeship literature (cf. Brown, Collins, & Duguid, 1990; Collins, Brown, & Newman, 1989; Boshuizen & Schmidt, 1992). The plan has three basic parts:

- Develop an articulate conceptual framework
- Reflect on situated rule-driven performances in real task situations

- Elaborate connections between situated (and rule driven) knowledge and the global framework

There are natural roles for tutoring in this kind of scheme. First, the tutor can present at least a rough skeleton of a conceptual framework that the trainee can use to organize new knowledge and experience. Then, it can support a process of reflection on the experiences of solving a particular problem or otherwise dealing with a particular situation. Finally, the tutor can help the trainee discover the most powerful connections between successful performance in the situation and the conceptual knowledge base(s) that can facilitate that performance. There are a number of different ways that such tutors can work.

Designing Learning Systems to Support Encapsulation

The overall goal of a tutor, within this viewpoint, is to find ways to connect conceptual knowledge to "clinical" or rule-driven knowledge. There are several ways this might be done, varying mostly in the grain size of the effort to connect rule-based and conceptual knowledge. Generally, systems focusing primarily on teaching a body of conceptual knowledge using problems as a means of shaping concepts, work with a relatively coarse-grained connection between specific problem situations and specific concepts and relational networks of concepts. So, for example, VanLehn's ANDES system (Gertner, Conati, & VanLehn, 1998), attempts to determine what knowledge is present and what absent in a student's understanding of mechanics, based upon the student's problem solving performance. What is unique to the ANDES approach, though, is multiple levels of focus. One addition to ANDES deals exclusively with self-explanation and coaches the student to carefully work through each example problem solved by the expert system, explaining to himself why each expert step was taken (Conati & VanLehn, 1999). Another variant provides a small minilesson covering, in a coherent way, a set of concepts that seem relevant to the kinds of mistakes the student has been making (Albacete, 1999).

An alternative approach, which we discuss in more detail below, is to provide very explicit links between particular features of problem situations and particular bits of conceptual knowledge. The advantage of this approach is that it focuses very directly on the specific problem of linking conceptual and other knowledge forms around specific problems. The possible disadvantage is that there may be little connection built up between the various conceptual fragments triggered by different problem situations. Compared to the macro level approach of VanLehn and his colleagues, this micro approach promotes inter-knowledge base connections explicitly and assumes that intra-knowledge base linkages will be handled by the trainee or will have been acquired previously. The VanLehn approach makes less

complete explicit connections between problem situational specifics and conceptual knowledge, leaving some of that work to the student, but its variants seem to foster learning of more intra-knowledge base connections by addressing them more explicitly. However, neither approach was designed to test this distinction, so we must assume that further research will be needed to identify how explicit these different kinds of connections must be made in the course of coaching complex problem solving.

Forms of Knowledge Needed to Support Learning by Doing

Learning by doing systems require several forms of data in order to be fully effective. In this section, I describe these data varieties. To be more concrete, I will speak of the needs for a specific class of learning by doing systems, namely systems that teach diagnosis of complex equipment failures. Later, I will discuss how these requirements generalize to other areas of learning. The following knowledge bases are needed for building such systems: (1) a simulation of the technicians' work environment; (2) specifications for a set of problems that reasonably cover the range of difficult-to-diagnose failures that we would like technicians to be able to handle; (3) a set of rules that capture sufficient expertise to permit the training system to solve every problem used for training; and (4) a set of conceptual principles that allow coaching to connect specific troubleshooting rules to core principles that lie behind the thinking experts bring to difficult problems.

It is important to bear in mind the three forms of expertise identified by Rasmussen when considering what knowledge is needed. Rasmussen suggested that technicians often operate at a level of practiced routine, in which they simply recognize what to do based upon automated, overlearned perceptual knowledge. When that fails, Rasmussen suggested, they elevate their cognitive processing to a richer representation of the task situation, to which they apply domain-specific operating rules. When that also fails, they elevate further and apply weak methods (general problem solving strategies) to their conceptual representation of the problem.

When we look at the goals of the training systems we have built, we see that we are preparing technicians for the non-routine parts of their work, the really hard stuff. Certainly, we aren't focusing on the practiced routine – that seems to be handled well by cheaper approaches. We aren't even really focusing on the more common of the rule-based expertise, since again that seems to get acquired via experience and "war stories." Rather, we are focusing on a mixture of the conceptually based "weak methods" reasoning and the most sophisticated of the rule-based processing, the parts most dependent on a powerful representation of the work domain in systemic terms. This means that connecting specific troubleshooting rules with deeper

conceptual content is a critical part of building the knowledge base for a learning by doing system. Further, since expertise does leverage the rule-based and skills levels of knowledge when appropriate, we need to be sure that the most important aspects of those levels are captured as well. Below, we consider these various knowledge requirements.

No matter which knowledge forms turn out to be most critical to achieving particular learning or training goals, it is likely that a number of different forms will be needed to support one aspect or another of an effective coached learning environment. First, the work environment itself must be represented, so that it can be simulated. In addition, expert rule-based problem-solving knowledge must be represented, since it is helpful for the system to be able to demonstrate how a problem can be solved or to take over in the midst of problem solution if the student is completely stymied. As we have already discussed, the conceptual knowledge that supports expert procedures must be defined and its connections to rule-based knowledge explicitly represented. Finally, a rationale must be developed for selecting learning tasks (problems for the student to solve).

The Work Environment

The relevant knowledge base about the environment for which students are being trained includes all of the attributes and methods needed to produce the level of simulation relevant to training. At one time, we had hoped that knowledge related to the work environment could be imported automatically from design databases used in building the work place (e.g., the computer-assisted design databases used in developing new machinery). While part of the needed knowledge can indeed come from that source, additional knowledge must come from experts who use or maintain the work environment. Specifically, we need to be concerned with three aspects of knowledge structure here, the *contents of the individual objects* that comprise the device simulation, the *classification hierarchies* that interrelates those concepts, and the information paths that comprise the work environment.

Function. The content of individual objects in the work environment can indeed be extracted from documentation of the environment's design, e.g., vendor manuals. In addition, though, it is important to know how experts organize the work environment into functional subsystems. This expert knowledge should be used to organize how the work environment is represented. The interface for access to the simulation should have a hierarchical display and menu scheme that recapitulates the expert decomposition of the work environment. This allows every interaction with the interface to reinforce the in the trainee a decomposition of the system that experts would apply. The purposes of components and subsystems need to be recorded as part of the knowledge base as well.

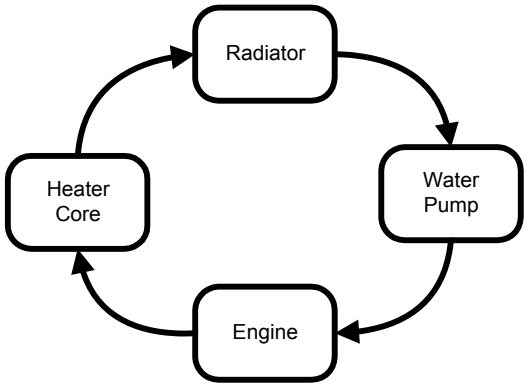


Figure 1. Schematic of cooling system.

Classification and flows. In addition, the various components of the work environment need to be organized categorically, into an inheritance hierarchy. This is important in developing the object definitions that turn the knowledge base into a learning environment. Another kind of knowledge needs to come from experts, too. This is knowledge about the flows of material, energy, and information between components. Consider, for example, the path that antifreeze takes in the engine of your car (see Figure 1). The engine receives coolant from the radiator and transmits it to the heater core. If the water pump fails, then the radiator will receive no coolant. So, we can define the water pump in part as passing on input water at a particular flow rate when it is running and otherwise simply blocking the line. Further, if we could verify that the radiator has a good water flow through it, we could infer that the water pump is at least partly operative. This sort of information, on what can be inferred about other system components from any given observation of some particular component, is needed to drive simulations of the work environment. Each component must know which other components to "tell" when its inputs or outputs change. And the input-output relationships within a component also must be represented. Much of this connection information can be inferred from the design documents for a machine, but some may need to be extracted from a technical expert.

The data that signals a problem. In order to present learning-by-doing simulations that are contextualized, it is necessary to acquire information about how problems manifest themselves. For example, if we were building a water pump tutor, we would need to know that water pump problems can manifest themselves via the signal for an overheated engine, through certain kinds of noises associated with a breakdown of the bearings or the rotating parts, and perhaps occasionally through large puddles of water under the car. Experts use this kind of symptom or context information to shape a representation of the problem and a plan for its solution.

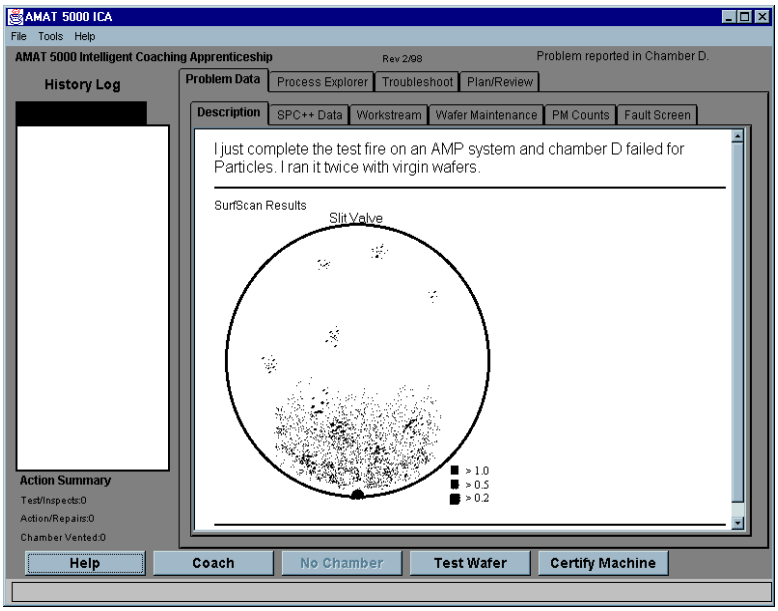


Figure 2. Initial staging of a problem.

Often, various kinds of instrument and console displays are part of the work environment. In such cases, part of the staging of a problem will include aspects of these displays. Figure 2 shows an example of this from a training system built by a joint team of Intel and University of Pittsburgh colleagues. In this case, a special kind of display illustrating the surface of a wafer (from a computer chip processing machine) is included in the specification of the problem, and other tabs in the display frame (e.g., "SPC++ Data") provide access to additional displays from various consoles that might be relevant to the problem. Overall, a number of forms of data that fix the context for a problem may be needed, including some that need not be dynamically represented during the process of problem solving. For example, the log of past actions on a machine can be represented statically – the past will not change while a trainee is working on a problem. On the other hand, a statistical process control (SPC) chart may well need to be incremented as the technician cycles the system through activities that would ordinarily produce chart entries.

Expert Diagnostic Knowledge

Rules. In any domain of expertise, much of routine activity is, as noted above, rule-driven. Consequently, it is important to identify the rules that characterize the process the expert uses to represent and solve a problem. These can be inferred from protocols of experts solving target problems out loud or, even better, by having experts go through a trace of their actual diagnostic activity while solving a problem, giving a justification for each action taken and an indication of what was learned from taking that action. As Gott has suggested (Gott, 1987; Gott, Bennett, & Gillet, 1986; Hall, Gott, & Pokorny, 1995; Means & Gott, 1988), much of this information can be

gathered using a "stimulated recall" approach,[4] which is discussed below. As is discussed below (see Page 23), various tools are available that allow an analyst to identify statements in thinking-aloud protocols from which rules should be inferred and to generate rules efficiently from those statements.

Expectations about actions. In addition to explicit cognitive rules, experts often have expectations about the results of an action. This permits a relatively automated level of self-monitoring in which the violation of an expectation becomes a signal that the expert's model of the current task situation needs repairs. It is quite reasonable to represent many expert rules in the form

```
IF <condition> THEN do <action> and EXPECT <outcome>
```

Other rules presumably are learned that have the satisfaction or nonsatisfaction of an expectation as their conditions. A specific kind of violation of expectations can itself signal an appropriate next step in the problem solution process. In some areas of expertise, the relatively automated ability of experts to check their decision making by generating expectations and monitoring whether they are met is especially important. For example, the tendency of physicians to converge on a diagnostic schema quickly is counterbalanced by their ability to do quick checks on whether the triggered schema matches the current situation (cf. Lesgold, 1984).

A further role for expectations is to stimulate the conceptual knowledge-driven level of cognitive activity. Rasmussen (Rasmussen, Petjersen, & Goodstein, 1994) suggested that failure of a given lower level of knowledge to drive problem solution triggers higher levels of knowledge. While some such failures will arise from other kinds of impasses – "I don't know what to do here" – very often it will be the violation of an expectation that triggers conceptually driven processing. If an intelligent learning-by-doing system for training is to address the multiple levels of expertise, then it needs to include information about expectations for actions, especially actions driven by perceptual-motor or rule-based knowledge.

Expert Conceptual Knowledge

Expert conceptual knowledge is most useful for intelligent learning-by-doing systems when it is interconnected with the manifestations that can

---

[4] I have not been able to find the source of the term *stimulated recall*, though it seems to have become a widely used method (e.g., Nunan, 1992; Wixon & Ramey, 1996). Basically, it refers to a scheme in which a think-aloud protocol is recorded, usually on video but not always, and then the person who did the thinking aloud is stepped back through the protocol and asked focused questions about each step of the problem solution process. Gott and her colleagues used a variation in which the protocol was taken down in notes by a knowledge analyst who then cycled through each recorded step asking several specific questions about each.

occur in training problems. This allows the conceptual knowledge to be made available at times when it can be connected to situations in which it might be needed, thus avoiding Whitehead's "inert knowledge" problem. However, it has always proven difficult to organize conceptual understanding in this way. In school, this happens in somewhat degenerate form when a class is presented a collection of problems in a specific form directly tied to the day's lesson, after which that form may never appear in class again. For example, physics classes offer inclined plane problems, and algebra classes have their unit on mixture problems. This level of attachment of conceptual knowledge is too coarse-grained. In real life, we do not encounter pure mixture problems or pure inclined plane problems. We need to connect conceptual knowledge to something more like a symptom or manifestation of a problem. This may produce multiple candidate fragments of potentially relevant conceptual knowledge, but at least it will bring the right candidates to mind at the right time.

Martin Nahemow has developed this idea in a particular instructional device he called a *process explorer*. Basically, the idea is to organize conceptual knowledge according to kinds of functions served by various aspects of the work environment. For example, in machines that put layers onto silicon wafers as part of the process of making computer chips, plasma processes are involved. This means that some of the basic functions in such machines include the delivery of various gasses into the chamber in which a plasma reaction will take place, provision of energy to the chamber in the form of heat and electrical power with certain properties (voltage and frequency), and robotic processes that move the wafer to the right place, etc.

But this is not enough. These causal aspects must be mapped onto the outcomes that can occur in the work environment (in the present example, that environment is a wafer layering machine). This mapping can be formalized as a matrix that maps the connections between problem specification data and a prioritized set of functional models. This requires that we distinguish several different kinds of mappings, some in the realm of work environment design and others dealing with problem solving given a design already in place. In system design, one is concerned with a process matrix $\mathbf{P_{ij}}$ that maps the subsystem functional models represented by a vector $\mathbf{F_j}$ into the nominal output parameters represented by a vector $\mathbf{N_j}$.

$$\mathbf{N_i = P_{ij} \bullet \{F_j\}}$$

Problem solving is concerned with the inverse of this design in many cases, starting with a set of deviations from the nominal output parameters. This deviation can be is represented by a vector $\mathbf{\Delta_i}$ and the critical mappings for training problem solving are represented by a matrix $\mathbf{M_{ij}}$ such that

$\mathbf{F_j = M_{ij} \bullet \{\Delta_i\}}$

Thinking more broadly, $\mathbf{P_{ij}}$ could be thought of as the Engineer's worldview and $\mathbf{M_{ij}}$ as the Technician's worldview. One of the most challenging tasks for knowledge engineering results from the fact that $\mathbf{M_{ij}}$ and $\mathbf{P_{ij}}$ are rather orthogonal. Generally, there is no way to map one into the other directly or to observe them both at the same time. For coaching purposes in a learning-by-doing system, we must capture the essence of the troubleshooting experts' worldview and knowledge. This must be consistent with the engineering model but may not be related to it easily.

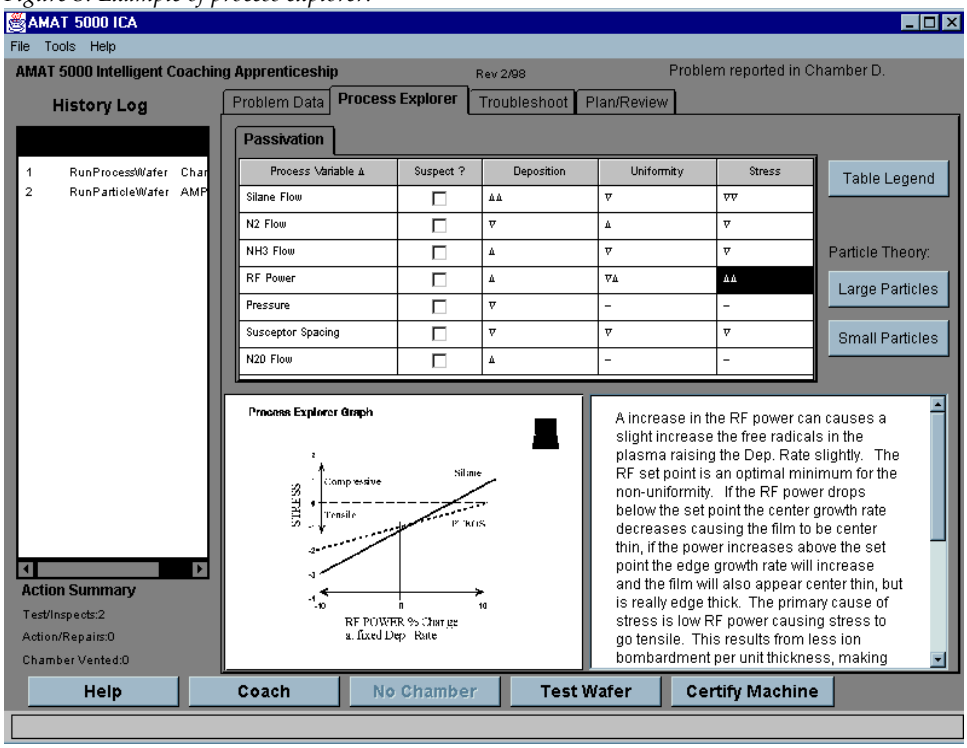
We have developed a practical form – which we call the process explorer – that is suitable for making conceptual knowledge available to trainees for examination during problem solving. It consists of a matrix corresponding to $\mathbf{M_{ij}}$ for which each cell is a hyperlink to an explanation of the relationship between a problem state defined by Δ_i and a functional model $\mathbf{F_j}$. The icon for each hyperlink is drawn to represent the type of relationship between the problem manifest by Δ_i and the functional subsystem represented by $\mathbf{F_j}$. For example, if a manifestation like grit on the wafer were more likely when the heat system was not delivering sufficient heat, then the cell in $\mathbf{M_{ij}}$ representing the connection between heat and grit might contain a ↓ to show that when heat drops, grit increases. Such icons themselves might give the trainee enough of a hint to help him/her limit the set of subsystems that could cause the set of fault indicators, and the hyperlinks can connect to additional graphics and text that further enhance the understanding of relationships and allow for finer tuning of the development of an action plan.

In addition to being part of the mapping between conceptual knowledge and problem manifestations, the breakdown of conceptual knowledge according to functional subsystems of the work environment also supports a scheme for reifying expert functional understanding in the interface to the simulated work environment. We have used schematic diagrams as components of menu systems that trainees use to gain access to a given component of the simulated work environment. Each functional subsystem in the simulation is represented by a schematic diagram that shows all the system components that can be acted upon within that subsystem.

One important task is to decide on the submatrix of the complete mapping $\mathbf{M_{ij}}$ that should be on the screen for a given problem. Ordinarily, one specific cell of the matrix will contain the symptom-concept mapping that is directly relevant to the solution of the problem being posed. However, in order to support search of the problem space and in order not to give away the solution to the problem, it is important to display more of the mapping than that single cell. On the other hand, the matrix is quite large and likely to be

confusing if presented in its entirety. So, it is necessary to decide on an appropriate submatrix for each problem that contains cells that might be the basis for sensible query but does not contain irrelevant rows or columns. This can be done by working from the matrix cell that represents the problem in two directions. First, the matrix should include, as rows, all conceptual categories with strong associations to the particular experience the problem manifests (e.g., if the problem is excess stress, then show all conceptual categories associated with a deviation in stress). Second, the experiential categories associated with the most likely conceptual categories should be represented as columns of the process explorer matrix. An example of an actual submatrix for a problem in one of our systems is shown in Figure 3.

Figure 3. Example of process explorer.



The Problem Set

The final form of knowledge that must be gathered to support learning by doing is a class of troubleshooting problems that span the range of situation types that an expert can expect to face. Presumably, the first concern in developing a problem set is to "cover" the conceptual hierarchy. That is, each kind of system breakdown should be represented in one or more problems. If enough resources exist, one scheme would be to look, for each conceptual category, at the set of surface manifestations bearing a strong relationship to it and write a problem for each such pairing. I.e., if there are a set of conceptual categories of problem areas $\mathbf{F_i}$ and a set of experiential categories of failures $\mathbf{\Delta_j}$, then the *set of problems* is given by the set of *ij* pairs for which

$\mathbf{p}(\Delta_i|\mathbf{F}_j) \geq \theta$, where θ is the lowest level of event probability for which we want to provide training. That set will likely be too large. While no pruning process is guaranteed to work, those selection schemes that give some exposure to each of the functional subsystems seem most likely to work.

One reasonable scheme is to define the problem set after a preliminary round of task analysis aimed at defining $\mathbf{P}_{ij}$ and $\mathbf{M}_{ij}$. Having these defined permits a rational approach to specifying a sufficient set of problems, namely to attempt to cover as many of the cells of $\mathbf{M}_{ij}$ as possible, or, as suggested above, at least as many of the rows and columns as possible. Once the problem set is defined, it may be appropriate to revisit the specification of the simulation, the process explorer content, and the expert rule set to be sure that all and only the knowledge needed for training using the defined problem set is specified. Put another way, any iteration in the process of task analysis and system development will pivot on the problem set. The other knowledge is needed to roughly specify a sensible problem set, and the final problem set is the appropriate base for testing the adequacy of the rest of the knowledge base for the training system.

Tools for Building Learning Systems

The Knowledge Gathering Process

The basic knowledge engineering method is to record an expert solving a problem while thinking aloud, infer a set of rules from the recorded verbal protocol, apply those rules to solutions of related problems, get expert critique of the rule-based solutions, and then edit the rules to take account of the critique. The last three steps (solve using rules, get critique, edit rules) are iterated as necessary until the rules seem to work adequately. This approach is sufficient when the goal is simply to establish the rule system that an intelligent tutoring system is to train. However, it is insufficient when the broader goals of learning by doing are present.

Several approaches have been developed to facilitate gathering the richer data needed for systems that support deeper learning. One, mentioned above, was the PARI method (Gott, 1987; Gott, Bennett, & Gillet, 1986; Hall, Gott, & Pokorny, 1995; Means & Gott, 1988). In this approach, several additional steps are added to the stimulated recall approach discussed above. Experts solve real-world problems that are presented verbally, with the tester providing a verbal response to each proposed action. For example, if the domain were flashlight repair and the problem was to figure out why a flashlight isn't working, the expert might suggest testing the voltage drop across each of the flashlight's batteries. The tester would then tell the expert what voltage levels to assume, the expert would then propose a next action, the tester would indicate the result, and so on until the expert stated a proposed repair. The

tester would then tell the expert whether the repair was successful. After this process was completed, the tester would walk the expert through each of the actions he took, asking, in turn, four questions about each:

- Precursor: What did you know at this point, prior to taking this next action?
- Action: What actions were potentially reasonable, and why did you pick the one you picked?
- Result: What were the possible results of your action?
- Interpretation: What did you learn given the results that actually were reported to you?

From the combination of the actual problem solving steps and the subsequent responses to the probe questions, it is possible to develop a list of the subsystems and components that need to be understood in order for problems of the type used to be solved expertly. From the response to the various queries, it is possible to develop the specifications of the expert conceptual and diagnostic knowledge that are embodied in the solution offered by the expert. So, the PARI approach is extremely useful for getting from problem solution protocols to a knowledge specification. However, more work is needed in order to get all the knowledge described in the previous section.

One additional requirement is the clear specification of the knowledge that might be available and relevant to initial specification of the problem. It is generally necessary to ask experts about this specifically. There is also need for further probing to establish all the knowledge relevant to the device simulation and to experts' structuring of their own representations of the device. This process can begin with a list of system components mentioned during the solutions to various problems that are generated as part of the PARI process just described. Then, it can be useful to get from experts a sense of how these various components should be classified. This can be done through various card-sorting processes or through the use of tools for specifying inheritance hierarchies (the card sort and laddering tools from Shadbolt's PC PACK suite, described below, are useful for this purpose).

The precursor information can be used to develop rules that will become part of the expert diagnostic knowledge component of the knowledge base. Suppose that prior to taking action α_i, the expert noted precursor states π_1, π_2, π_3, π_4, and π_5. This might prompt a task analyst to propose a candidate rule

IF π_1, π_2, π_3, π_4, and π_5, THEN α_i

This rule could be edited further by discussing it directly with the expert, and it would certainly need to be tested as part of the overall rule base.

Similarly, the expected-results information can be used to generate additional candidate rules, especially if the expert is probed for the implications of the possible outcomes to any action he takes. For example, when diagnosing a flashlight, if I measure the voltage levels of the batteries, there are two basic outcomes. Either the voltage level is close to nominal for a functioning battery (1.5v for lead-zinc batteries), or it is too low. If the batteries are weak, then the solution is to replace them. On the other hand, if they are adequate, then an expert might move on to examine the bulb. This simple choice can itself be expressed as a couple of rules:

```
IF battery-voltage-level « 1.5v, THEN replace battery

IF battery-voltage-level ≈ 1.5v, THEN assume-battery-good AND assess-lightbulb
```

Rule editors (discussed below) can be helpful in formulating rules from the problem solving protocols and the probe questions. Any such formulation, however, is an induction from a single instance and must be validated. This validation, moreover, cannot be completed until the entire set of problems for a training system has been defined, since the ultimate test of any rule is that it contributes to (or at least does not interfere with) an efficient solution for each problem in the domain. For this reason, expert diagnostic knowledge must be tested repeatedly and not accepted until it is proven to work successfully with any problem that falls within the scope of the domain for which the training system being built.

It is sometimes possible to infer certain kinds of rules from declarative knowledge. Consider, for example, Table2, which one might build based upon the knowledge needed to simulate working and broken flashlights.

Table 2. Device information from which rules might be inferred.

Item	State	Manifestation
Batteries	Adequate	Voltage ≈1.5v
	Inadequate	Voltage « 1.5v
Bulb	Working	Filament appears intact
	Not Working	Filament appears broken
Switch	Working	Continuous path (low resistance) through switch when on; high resistance when off
	Not Working	Resistance constant whether on or off
Caps	Working	Screwed fully into position
	Not Working	Not screwed down completely or lopsided

This table could be used to infer a set of rules that would probably be an adequate set for this very simple domain. What is not present in the information of the table, however, is any information about the optimal sequence for carrying out the various possible actions. Still, it can be useful to have inference engines to build rule sets from this kind of data, since often it is easier for experts to provide data in tabular form.

Deciding on Conceptual Knowledge and Connecting It to Problem Contexts

Determining the necessary conceptual knowledge to include in a coached learning environment and the connections it should have to particular problem situations involves several steps. The relevant knowledge must be identified, and it must be linked to problem contexts for which it is relevant. A crude way to do this is simply to note any conceptual content that is provided by subject matter experts during the precursor and interpretation phases of the PARI method discussed above. From the specifics of expert replies to the PARI probes, it is generally possible to determine whether the expert relied on conceptual knowledge to take a step or simply added such knowledge as an elaboration of the account of problem solving activity.

For example, if the action taken and the PARI responses suggest a specific diagnosis rule and the subject-matter expert confirms that this is a good rule, then there was probably not much reliance on conceptual knowledge to take the action. On the other hand, if no rule seems to both fit the expert actions and gain acceptance by the subject matter expert, then it is likely that the expert had to appeal to conceptual knowledge to arrive at the action taken. In this case, the relevant conceptual knowledge is likely to be in the precursor and interpretation replies, though follow-up confirmation is desirable if time and resources permit.

As the body of conceptual tidbits (the rows of the matrix that maps conceptual knowledge to problem situation features) and the contexts to which they are tied (the cells of the mapping matrix) accumulate, it becomes necessary to place some further organization on both. The organization of a set of conceptual knowledge fragments will be determined by the disciplinary character of those pieces. For example, if there are a number of pieces of conceptual knowledge that all deal with the gas laws of temperature and pressure, then it may be sensible to collapse these into a single account of those laws that connects a number of more specific instantiations of that basic account. In one system for which we designed training, problems tended to involve abnormal aspects of gas delivery, of energy supplied to certain reactions, and of robotic transport, so those could have become the organizing categories for conceptual knowledge for that domain. The instantiations become part of the content for the cells in the process explorer

matrix that connect the gas laws with particular problem contexts (an example is given by the lower right section of Figure 3).

The contexts that are represented by the columns of the mapping matrix can themselves be organized in a number of ways, depending on which situational features are important in the domain's expertise and how fault states are defined. For example, in one training system for repair of a certain computer chip-making machine, we had broad context features like grit on the wafers, improper wafer thickness, and surface stress outside of quality limits. We might have chosen other categories if they seemed more salient to technicians, such as valve problems, robotic problems, electrical problems, etc.

The choices of categorization schemes for both rows and columns of the mapping matrix are important. It is rare for complete linkage and conceptual information to come out of a single round of task analysis with PARI probing. Rather, it is generally necessary iteratively to

- add to the mapping matrix based upon responses to post-solution probing (e.g., PARI),
- organize the rows and columns categorically, check for missing rows and columns (kinds of knowledge or situations that experts feel were left out),
- develop more problems to cover the matrix (as discussed above), and then
- gather additional data from experts who solve those problems and then respond to probe questions.

Some Available Tools

Tools to support cognitive task analysis are extremely important. The high cost and inadequate results of many instructional and training technology efforts have been due partly to a lack of schemes for assuring the completeness and depth of the analyses performed and also to a gap between the discursive style of many analyses and the knowledge structures needed to support rich learning-by-doing approaches. Even in our own work, we initially found that we needed to add additional knowledge acquisition sessions at the last minute, because content was incomplete or insufficiently understood. In the industrial environments in which we work, a single extra knowledge acquisition trip may generate $5,000-20,000 in added costs for travel, replacement on the factory floor of experts being interviewed, transcription, etc. Given that we aim for a technology that can produce a learning-by-doing system for under $100,000 (we're not quite there yet), we

cannot afford unpredicted costs in this range. So, tools and approaches that assure completeness are essential to our work, especially those that cut the interval between initial interviews and final data structure specification.

Because tools for cognitive task analysis have co-evolved with analytic approaches, one way to better understand some of the methods for getting knowledge from experts is to look at some of the tools that have been built. We focus in this section on tools built by Nigel Shadbolt and his colleagues and now sold as a product called PC PACK.[5] This is a tightly integrated set that has a number of useful components. While we have not found it perfect for our needs, it has been useful, and we have been stimulated by it to develop some of our own, more tailored tools.

One reason for picking this particular tool kit as an example is that it is object based. That is, any component that is explicitly or implicitly identified through the use of any of the tools is represented in an object database that is active whenever any of the tools are being used. Further, since a given kind of object can be identified many different ways, it is important that the tools be integrated. So, for example, a concept that is identified through marking a word or a phrase with a protocol mark-up tool will also be available to tools like the card sorting tool and the tool for developing inheritance hierarchies.

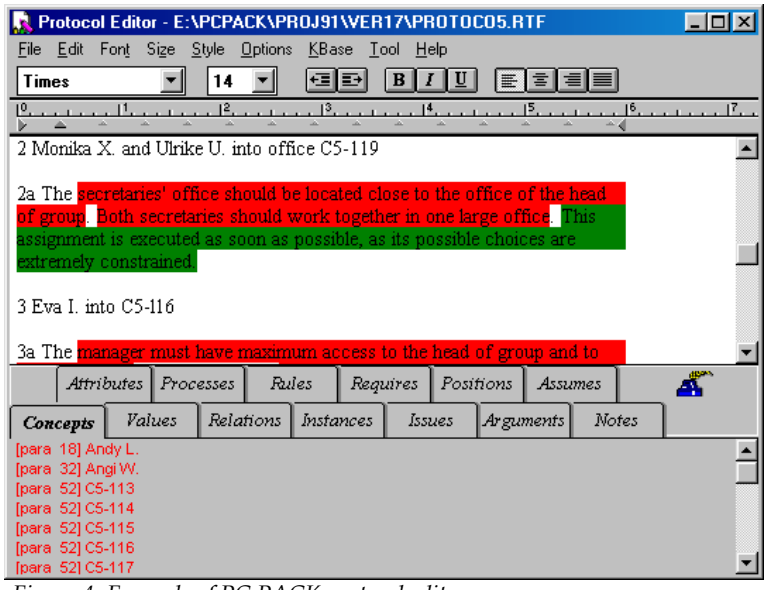


Figure 4. Example of PC PACK protocol editor.

Protocol Editor

As we consider each kind of knowledge needed for a learning-by-doing system, we can see that each would benefit from some specific tools. In addition, certain more general tools are useful broadly. One such general tool

[5] See http://www.epistemics.co.uk/ for information on PC PACK, which is a trade name of Epistemics, Ltd. The authors have no connection with Epistemics, Ltd. We simply chose to use it as an example because the particular tools it contains match well with our overall approach. Our own work has used self-built tools (largely developed by Dan Peters at LRDC and Scott Smith at Intel) because of special needs and the timing of the start of our work.

is a protocol editor. This is a tool that takes the transcript of an interview with an expert and allows it to be marked up to indicate the categories with which each statement should be associated. The most important categories for learning by doing include concepts, attributes, values, relations, and rules. Figure 4 provides an example of a piece of protocol document that is marked with one color in two places to indicate text implying a rule and another color in one place to indicate text implying a process. The ability to systematically catalog all prose segments that imply the need for various knowledge structures is extremely important in making the protocol analysis process efficient and complete.

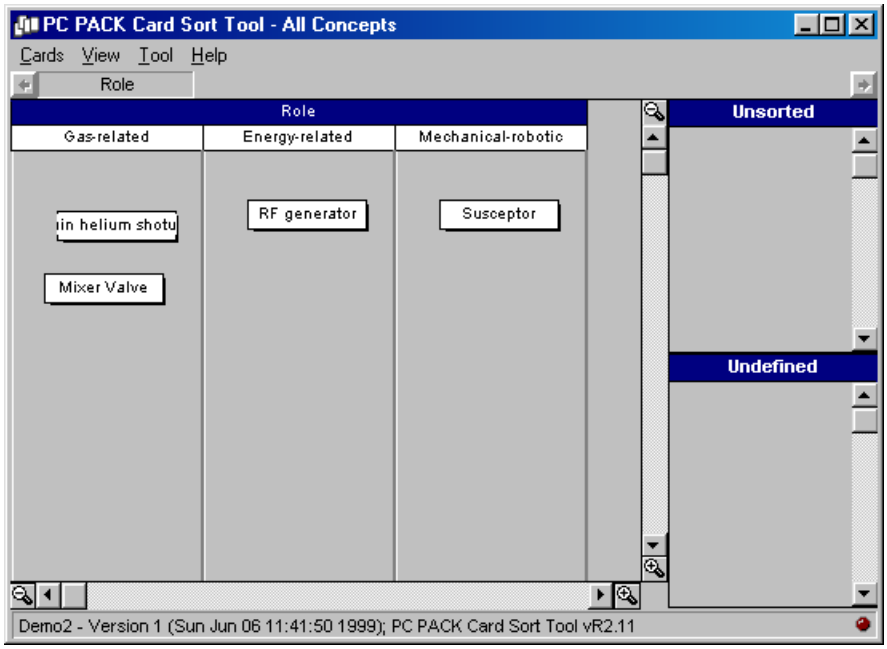


Figure 5. Card Sort Tool.

Card Sort Tool

Another tool in the Epistemics PC PACK is the Card Sort Tool. From the earliest cognitive task analyses designed to support the building of learning-by-doing systems (cf. Gitomer, 1988), one way to get attributes and values assigned to objects is via card sorting tasks. The expert is given cards on which the names of objects are placed and asked to sort them into groups that share properties. Once this scheme has gone far enough for those properties to be identified and labeled, then additional concepts can be sorted into those categories.

PC PACK supports this via a virtual card-sorting scheme. Attributes and values are assembled into hierarchical lists, and then concepts can be dragged on the screen into the appropriate "pile." For example, in one recent project, it turned out that experts roughly classified components of a complex system

as either energy-related, gas-related, or mechanical-robotic (the system used energy and gasses to create plasma fields into which other objects had to be transported very precisely. Figure 5 shows an example of a few "cards" naming components of a mythical system sorted into the three "piles" indicated above. These "cards" would likely have been generated automatically after a knowledge analyst marked phrases in an interview transcript or the documentation for the system involved.

Laddering Tool

The laddering tool supports the accumulation of attribute and category information through a different approach, namely hierarchy specification. Consider, for example, the concepts sorted in Figure 6. If one were to begin developing an inheritance hierarchy for these concepts (i.e., a hierarchy that showed the level of abstraction at which different properties are defined for various classes of objects), it might look something like that shown in Figure 6. The laddering tool in PC PACK permits cognitive analysts to build such hierarchies directly. New concepts can be added to a list, and individual concept boxes dragged to appropriate places in the hierarchy. As with everything else in PC PACK, information represented with the laddering tool is automatically included in relevant object definitions that are stored in the object database. Figure 7 shows one example of a simple object definition that was created by

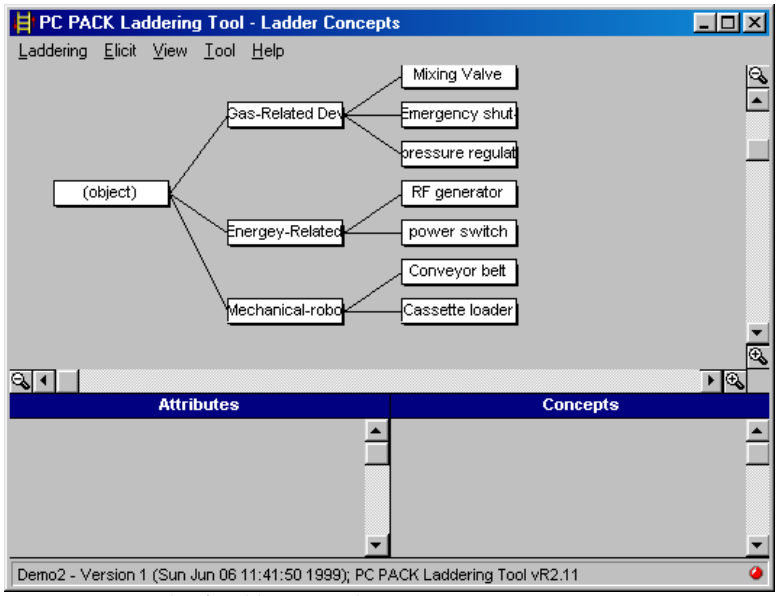


Figure 6. Example of Laddering Tool.

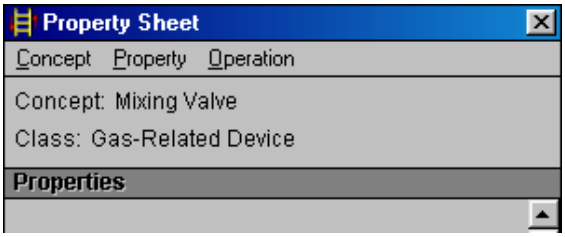


Figure 7. Example of one object defined by laddering tool.

the placement of the "mixing valve" box in the laddering display. From the laddering display, the system has noted that "mixing valve" is an instance of "gas-related device." If any properties were defined for all gas-related devices, they would automatically be "inherited" by the mixing valve.

This ability to convert the manipulations of a knowledge-gathering interface into formalized objects is a central part of the tool requirement for efficient cognitive task analysis. Software designers can elaborate computational objects so that they include not only attributes and values but also "methods." For example, one use we made of an object architecture was to intelligently generate displays of schematic information. Each object to be displayed knew how to draw itself, and an object at a given level knew how to allocate its overall space in a drawing to the objects of which it was composed. Finally, there was an attribute for each system object that indicated whether, in the current context of a problem solution, an expert would be thinking about that object. Objects being thought about were given more space and made more salient by the self-drawing method. This meant that the final diagram drawn on the screen when a part of the system was asked by the program to draw itself was dynamically altered to draw attention to the part(s) an expert might be focusing on in the instant context.[6]

Rule Inference and Rule Editing Tools

The other major formalized knowledge form is the rule. Actually, we have pretty good schemes for writing down rules, since most cognitive analysts have had some exposure to the formalism of production systems. However, it can be much more efficient to have automatic cross-referencing of the objects named in rules to the contents of the declarative knowledge base. This is especially important in light of the need for connections between conceptual and procedural knowledge discussed above on Page 5 and Page 15. This kind of cross-referencing can permit at least an approximation to the final process explorer matrix, and such an approximation can focus and contain the total time that must be invested in developing this matrix of connections among levels of knowledge.

PC PACK contains several mechanisms for developing rules. First, they can be defined directly, one at a time. Second, they can be inferred from "dependency tables" such as the one shown in Table 2. For example, the table implies rules such as *"IF the battery voltage is much less than 1.5V, THEN replace the battery."*

---

[6] Edward Hughes developed this approach while at LRDC.

Assessment in the Context of Learning-by-Doing Systems

As noted above, an engineering research approach would not address the generic question of whether learning-by-doing works. Rather, it would focus on establishing some of the boundary conditions under which it is effective and would also attempt to develop measures that might support trade-off studies comparing learning-by-doing to other approaches as solutions to specific training or education problems. We have begun to do this kind of work in some of our joint activities with industry. To do this, we have had to choose – and to some extent develop – appropriate measures for assessing the yield of learning-by-doing systems.

Across our various projects that produced training systems, we have relied upon three different kinds of evaluation. First, and most important, we have assessed the ability of trainees to solve hard problems from the target domain and even from transfer domains that bear some similarity to the target domain. Since we needed to develop training problems of this type as well, it was easy to stretch the process and develop additional problems for pre and post testing. Second, we have developed various schemes for assessing technicians' ability to explain actions that they or others take in trying to solve a difficult problem. Finally, we have attempted to train neural networks to recognize more versus less expert patterns of problem solving activity by technicians.

Scoring Performances Using Policy Capturing Methodology

The most straightforward test of competence in a domain is to place a person in that domain and give them a very hard problem[7] to solve. Then, it is necessary to evaluate the person's problem solving performance, to determine its value. In many cases, the goals of training represent a mixture of assimilation of expert processes and learning to optimize the value of one's performance within a local economy of practice. To get the needed information on the value of performances and to achieve the appropriate mixture of criteria, Gott and colleagues (Gott, Pokorny, et al., 1997) used a policy capturing approach (Hobson & Gibson, 1983) to establish the features of valued performance in the evaluation of Sherlock.

The approach consisted of having technicians solve difficult real-world troubleshooting problems that were presented verbally, with follow-up structured interviews. The interviews used the same probe questions

[7] Actually, this is already a value decision. Our training goals have always been to produce people who could stretch their knowledge to handle extremely difficult and often novel problems in their work. One could readily imagine a variety of training demands that would focus on routine work instead. For such training, different assessment measures would likely be more appropriate.

employed in Gott's task analysis methodology, the PARI method (Gott, 1987; Gott, Bennett, & Gillet, 1986; Hall, Gott, & Pokorny, 1995; Means & Gott, 1988). These questions probe for understanding of the current problem solution context, understanding of available options, and having accurate expectations of the likely results of contemplated next actions and the meaning of such results.

The anonymous solution traces were then given to job experts, who were asked to order the performance traces according to how well they represented desirable performance within the particular work context these technicians occupied.[8] Details of the approach are described in the next section.

Such a scoring scheme can also be used for criticism of trainees' performances in the simulated work environment, provided that all of the features can be operationally defined in terms of patterns of activity a trainee might exhibit while solving a troubleshooting problem. Further, if the presence of each feature is inferable from some aspects of the trainee's performance, then each element of a critique can be "explained" by reconstructing the inference path that led to it being listed for a given problem-solving performance.

To summarize, both the evaluation of trainee performance and the criticism of that performance can be based upon features of performances that are derived via policy capturing, provided that for each such feature, there is a way to infer the presence or absence of that feature reliably from the activity of the trainee.

Policy Capturing Approach

The basic policy-capturing scheme is very straightforward. Records of various people's performances on a set of problems are given to a panel of experts to score. Their first step is to rank the performances from best to worst. Then, they are prompted to account for the ordering by pointing out features that are present in some performances but not in others. These features constitute a set of variables that are candidate features for scoring. Via regression or other less formal schemes, points are assigned to the various features, so that the rankings can be recovered from the features present or absent in the various performances. These point values then constitute a scoring rubric for scoring future performances by others. It is essential in this

[8] Some worked in the Central Command, where they had adequate parts inventories and a strong mandate to repair equipment as fast as possible. Others worked at reserve bases or other sites where minimizing the use of new parts was the primary requirement, even if this meant longer diagnostic sessions to confirm absolutely the need for any part before it was ordered.

process to work with the expert panel to assure that all the features are defined objectively, so that non-experts can do future scoring if necessary.

Our Sherlock experience (Lesgold, Lajoie, Bunzo, & Eggan, 1992; Lesgold, Eggan, Katz, & Rao, 1992) taught us that there are two basic ways to do the policy capturing and competence assessments. One is to administer the problems completely verbally. That is, an assessment technician states the problem to the person being tested, and the person then tells what his first action will be. The results that would occur in the real world if that action were taken, given the predesignated system fault, are then stated verbally by an expert who works with the assessment technician. After the problem is solved, structured probe questions are used to get additional information about the testee's goals and understanding at different points in the solution process.

The other way to administer problems is to present them via the simulation capabilities of the learning-by-doing system itself. It would be possible to prompt for additional information in a manner similar to the probe questioning, using either text windows or audio recording to capture the responses. This has the advantage that it does not require a domain expert to be present during all the testing. It has the disadvantage that trainees who have been using the system may be more able to work easily on criterion problems and thus may have an advantage over a control group even if the training taught them nothing specific. This disadvantage seems minimal with a group of technicians who are used to using computers, and it can be further ameliorated by allowing testees to take as long as they need to do the problems and using audio recording to capture the responses to probe questions.

Mapping Competence to Value

It is important when doing policy-capturing work to be sure to distinguish an evaluation of performance focused on understanding of the system and its possible failures from an evaluation based upon cost-benefit optimization. The policy capturing scheme just described is very important for training development and refinement, since it can reveal specific areas of competence that are addressed well or poorly by the evolving system. However, in order to determine the economic value of a training approach, it is helpful to have a second kind of evaluation that focuses directly on the cost and value of trainee actions in simulated troubleshooting tasks. This second approach also starts with performance on problems presented either verbally by a training technician or else via the learning-by-doing simulation.

In this case, though, each action the trainee takes is assigned a cost that is determined by the amount of the trainee's time the action would take in real

life, the cost of any materials that would be used, and the cost in terms of the duration that other employees and/or machinery would be unable to function. It is possible in our learning-by-doing systems to place such cost assignments on a scale that is anchored by the cost that would result from the fault if it were immediately addressed by the training system's expert model and by the expected cost for a fault of that type in a work environment with a recently acquired and consequently minimally trained work force (something that would have to be estimated by an expert).

This kind of cost-based scoring can be done substantially by non-domain experts, but experts will occasionally need to be consulted to clarify time estimates for different actions and to delineate the full range of cost sources. Also, cost scoring can, to a large extent, be linked to the policy-capturing scheme, so that approximate cost values are assigned to each feature used in scoring.

Questions Focused on Explanation and Understanding

Another form of assessment focuses on understanding. In this approach, the assessment "items" begin with accounts of some aspect of troubleshooting and then ask the trainee to explain an action that might be taken or to explain some aspect of the situation that has been described. This approach ahs the advantage that it can be administered and scored by non-experts, once the items are developed, at least if "objective" forms like multiple choice are used. Further, this approach has been shown to produce results that bear a reasonable relationship to the results of performance tests like those described above (Gott, 1987; Gott & Lesgold, in press). The primary danger in relying exclusively on such items is that over time, training may evolve to focus only on conceptual understanding and not on the decision making that goes into complex problem solving. Explaining someone else's decisions is quite different from making them oneself (those who describe championship chess tournaments are not always able to win one!).

Scoring with Neural Networks

We have conducted one last kind of analysis of the protocol data from our direct performance testing. This is a test of whether the difference in overall pattern of performance between pretest and posttest subjects on a given problem was sufficient to be discernible in every individual case. The way we tested this was to ask if a simple (three hidden units in a single layer) neural network could identify pretest versus posttest protocols. We built neural networks in which each possible action (test, adjustment, or part replacement) that could be performed was assigned a separate input unit. Two output units were established, representing the choice between pretest and posttest. The individual protocols were then used as training cases to train the network to recognize whether a given case was a pretest

performance or a posttest performance. This was done separately for each of the four problems. The networks were standard three-layer networks, and backpropagation was the network training method.

Table 3 below reports the results of a neural network analysis for an evaluation of one of the training systems we built jointly with Intel Corporation. We used four troubleshooting problems from a library of problems – other problems in the library were selected for use in the training system itself. These problems were problems 3, 7, 8, and 12 of this library and hence are referenced below by those numbers. In this analysis, 0 was the arbitrary value assigned to the classification value for pretest performances and 1 was the value for posttest performances. Hence, the neural network would have classified perfectly if it had learned to generate a 0 for every pretest case and a 1 for every posttest case. In fact, the network converged on a strong solution for Problems 3, 7, and 8, and it produced almost as good a result for Problem 12. Again, it was easier to distinguish pre and post performances on Problems 7 and 8, just as they also gave stronger differences on the action counts, but all four problems showed clear differences between the patterns of pretest and posttest performance. While it is not easy to provide a concise statement about the exact pattern differences between pretest and posttest, the data are consistent with the claim that technicians were learning from the ICA system to avoid time-wasting strategies in their diagnostic work that were either unnecessary or not very informative.

Overall, the multiple evaluation methods described above are helpful in determining the effects of learning-by-doing environments, especially if the goal is to produce expertise sufficient for addressing rare and difficult problems that are especially expensive sources of difficulty for complex work systems.

Table 3. Neural Network Classifications.

Prob	Mean Output Score for Pretest	Mean Output Score for Posttest
3	.026	.704
7	.102	.863
8	.016	.907
12	.063	.845
Aver	**.052**	**.830**

Conclusions

We have described the knowledge requirements for learning-by-doing systems, especially those focused on complex problem solving. In doing so, we have argued that an important form of knowledge that is seldom addressed sufficiently is the connections between rule-based knowledge as it is applied in various problem situations and conceptual knowledge that

provides the basis for stretching rules to fit new situations. We have presented a justification for focusing on such knowledge, derived from the work of Rasmussen and more recent work on knowledge encapsulation by Schmidt, Boshuizen, and van de Wiel.

We have suggested specific methods for gathering the full range of data that learning-by-doing systems require and have discussed some of the relationships among different parts of that data. Out of this has come an initial proposal for selection of problem situations to be presented by a learning-by-doing system, namely to select problems to cover the space defined by the mappings between conceptual knowledge fragments and problem situations. While we have not yet performed specific tests of this selection approach – since it is hard to define an appropriate comparison group – it does at least have a reasonable rationale, and systems built on this scheme have proven effective.

Existing tools are a start toward what is needed, but none have the needed mixture of tight anchoring in an engineering theory of learning-by-doing system design plus sufficient usability and generality to be readily deployed for new training system development by instructional design experts or subject matter experts who may lack software backgrounds. The biggest gap is in systems for directly and completely conducting the iterative process that is needed to develop the mapping matrix relating conceptual to situational and rule-based knowledge. We hope eventually to design such tools if they don't appear from other sources.

Finally, we have reviewed several evaluation schemes that seem appropriate to assessment of learning-by-doing systems. The schemes have been used in out own work, and they have considerable face validity. What is still needed though is a richer evaluation model. After all, the history of instructional innovation has been that every intervention works in some "hothouse" environment and that almost all fail in some of the environments in which they eventually are applied. This suggests strongly that the ultimate goal of evaluation must be to establish the range of training requirements that can be addressed by a given approach, not simply to show that it worked somewhere.

References

Albacete, P. (1999). *An intelligent tutoring system for teaching fundamental physics concepts.* Ph.D. dissertation. Pittsburgh: University of Pittsburgh.

Anderson, J.R. and Corbett, A.T. (1993). Tutoring of cognitive skill. In J.R. Anderson, *Rules of the Mind* (pp. 235-255). Hillsdale, NJ: Erlbaum.

Brown, J. S., Collins, A., & Duguid, P. (1991). Situated cognition and the culture of learning. [Chapter] In M. Yazdani & R. W. Lawler, (Eds.), *Artificial intelligence and education,* Vol. 2. (pp. 245-268). Norwood, NJ: Ablex Publishing Corp.

Boshuizen, H. P. A., & Schmidt, H. G. (1992). On the role of biomedical knowledge in clinical reasoning by experts, intermediates and novices. *Cognitive Science, 16,* 153-184.

Boshuizen, H. P. A., & van de Wiel, M. W. J. (1999). Using multiple representations in medicine: How students struggle with them. In M. W. van Someren, P. Reimann, H. P. A. Boshuizen, & T. de Jong (Eds.), *Learning With Multiple Representations.* Amsterdam: Pergamon.

Chi, M., Glaser, R., & Farr, M. (Eds.), (1988). *The nature of expertise*. Hillsdale, NJ: Erlbaum.

Collins, A., Brown, J. S., & Newman, S. E. (1989). Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In L. B. Resnick, (Ed), *Knowing, learning, and instruction: Essays in honor of Robert Glaser*. (pp. 453-494). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Conati C., and VanLehn K. (1999). Teaching meta-cognitive skills: implementation and evaluation of a tutoring system to guide self-explanation while learning from examples. *Proceedings of AIED '99, 9th World Conference of Artificial Intelligence and Education,* Le Mans, France.

Gitomer, D. H. (1988). Individual differences in technical troubleshooting. *Human Performance, 1,* 111-131.

Gott, S. P. (1987). Assessing technical expertise in today's work environments. *Proceedings of the 1987 ETS Invitational Conference* (pp. 89-101). Princeton, NJ: Educational Testing Service.

Gott, S. P., Bennett, W., & Gillet, A. (1986). Models of technical competence for intelligent tutoring systems. *Journal of Computer-Based Instruction. 13*, 43-46.

Gott, S. P., & Lesgold, A. M. (in press). Competence in the Workplace: How Cognitive Performance Models and Situated Instruction Can Accelerate Skill Acquisition. In R. Glaser (Ed.), *Advances in instructional psychology.* Hillsdale, NJ: Erlbaum.

Gott, S.P., Pokorny, R.A., Kane, R.S., Alley, W.E., & Dibble, E. (1997). *Understanding the acquisition and flexibility of technical expertise: The*

development and evaluation of an intelligent tutoring system: Sherlock 2. AL/HR Technical Report 1997-0014. Brooks AFB TX: Armstrong Laboratory, Human Resources Directorate.

Hall, E. P., Gott, S. P., & Pokorny, R. A. (1995*). A procedural guide to cognitive task analysis: The PARI methodology.* Technical Report 1995-0108. Brooks AFB TX: Armstrong Laboratory, Human Resources Directorate.

Hobson, C. J. & Gibson, F.W. (1983). Policy capturing as an approach to understanding and improving performance appraisal: A review of the literature. *Academy of Management Review, 8,* 640-649.

Hunt, E., & Lansman, M. (1986). Unified model of attention and problem solving. Psychological Review. 93, 446-461.

Lesgold, A.M. (1984). Acquiring expertise. In J. R. Anderson & S. M. Kosslyn (Eds.), *Tutorials in learning and memory: Essays in honor of Gordon Bower.* San Francisco, W. H. Freeman.

Lesgold, A., Eggan, G., Katz, S., & Rao, G. (1992). Possibilities for assessment using computer-based apprenticeship environments. W. Regian & V. Shute (Eds.), *Cognitive approaches to automated instruction* (pp. 49-80). Hillsdale, NJ: Lawrence Erlbaum Associates.

Lesgold, A. M., Lajoie, S. P., Bunzo, M., & Eggan, G. (1992). SHERLOCK: A coached practice environment for an electronics troubleshooting job. In J. Larkin & R. Chabay (Eds.), *Computer assisted instruction and intelligent tutoring systems: Shared issues and complementary approaches* (pp. 201-238). Hillsdale, NJ: Lawrence Erlbaum Associates.

Means, B., & Gott, S. P. (1988). Cognitive task analysis as a basis for tutor development: Articulating abstract knowledge representations. In J. Psotka, D. Massey, & S. Mutter) (Eds.). *Intelligent tutoring systems: Lessons learned.* Hillsdale, NJ: Erlbaum.

Nunan, D. (1992). *Research methods in language learning*. Cambridge: Cambridge University Press.

Rasmussen, J., Petjersen, A. and Goodstein, L. (1994). *Cognitive systems engineering*. New York: Wiley.

Gertner, A. S., Conati, C., & VanLehn, K. (1998). Procedural help in Andes: Generating hints using a Bayesian network student model. *Proceedings of the Fifteenth National Conference on Artificial Intelligence AAAI-98.* Cambridge, MA: The MIT Press.

Whitehead, A. N. (1929). *The Aims of Education and Other Essays*. New York: The Free Press. [Available as of August 1999 from http://dept.english.upenn.edu/~rlucid/A_Whitehead.html]

Wixon, D. & Ramey, J. (1996). Field oriented design techniques: Case studies and organizing dimensions. SIGCHI Bulletin 28 (3), 21-26.