Learning to Troubleshoot: A New Theory-Based Design Architecture

Article *in* Educational Psychology Review · March 2006

DOI: 10.1007/s10648-006-9001-8

Woei Hung
University of North Dakota
93 PUBLICATIONS **2,062** CITATIONS

Educational Psychology Review, Vol. 18, No. 1, March 2006 (© 2006)
DOI: 10.1007/s10648-006-9001-8

Learning to Troubleshoot: A New Theory-Based Design Architecture

David H. Jonassen[1,3] and Woei Hung[2]

Published online: 28 June 2006

Troubleshooting is a common form of problem solving. Technicians (e.g., automotive mechanics, electricians) and professionals (physician, therapists, ombudspersons) diagnose faulty systems and take direct, corrective action to eliminate any faults in order to return the systems to their normal states. Traditional approaches to troubleshooting instruction have emphasized either theoretical or domain knowledge about the system or specific troubleshooting procedures. These methods have failed to develop transferable troubleshooting skills in learners. In this article, we propose an architecture for designing learning environments for troubleshooting. The architecture integrates experiential, domain, and device knowledge in a learning system that enables learners to generate and test hypotheses for every action they take, relate every action to a conceptual model of the system, and query experienced troubleshooters about what they would do. The architecture includes three essential components: A multi-layered conceptual model of the system that includes topographic, function, strategic, and procedural representations; a simulator that requires the learner to generate hypotheses, reconcile the hypotheses to the system mode, test the hypotheses, and interpret the results from the test; and a case library that uses a case-based reasoning engine to access relevant stories of troubleshooting experiences as advice for the learner. This novel architecture can be used to develop learning environments for different kinds of troubleshooting.

KEY WORDS: Cognitive processes; Problem solving; Troubleshooting.

[1]University of Missouri Columbia, 221C Townsend Hall,Missouri 65211.
[2]University of Arizona—South, 1140 N Colombo Ave, Sierra Vista,Arizona 85635.
[3]Correspondence should be addressed to David H. Jonassen , 221C Townsend Hall, University of Missouri Columbia ,Missouri 65211; e-mail: Jonassen@missouri.edu.

WHAT IS TROUBLESHOOTING?

Troubleshooting is among the most common types of problem solving. Whether troubleshooting a faulty modem, a multiplexed refrigeration system in a modern supermarket, or communication problems in an advertising agency, troubleshooting attempts to isolate fault states in a system and repair or replace the faulty components in order to reinstate the system to normal functioning. Troubleshooting is normally associated with the repair of physical, mechanical, or electronic systems. However, organizational ombudsmen, such as employee relations managers, customer relation specialists, consumer advocates, public relations specialists, and human resource directors are also troubleshooters. These people are responsible for handling complaints that represent fault states that must be repaired in customer relations systems. Individuals in their everyday lives engage in personal troubleshooting associated with self-change, especially when related to addictive behaviors (Prochaska *et al.*, 1992). Medical and psychological diagnoses also involve troubleshooting.

On the continuum of problems from well-structured (algorithms, story problems) to ill-structured (systems analysis, design), troubleshooting problems are in the middle (Jonassen, 2000). Troubleshooting problems:

- appear ill-defined because the troubleshooter must determine what information is needed for problem diagnosis (which data about the electrical and fuel systems are needed in troubleshooting a car that will not start)
- require the construction of a robust conceptual model of the system being troubleshot (how do electrical, fuel, and mechanical systems interact)
- usually possess a single fault state, although multiple faults may occur simultaneously (e.g., faulty battery, clogged injector)
- have known solutions with easily interpreted success criteria (part replacement leads to system restart)
- rely most efficiently on experience-based rules for diagnosing most of the cases, making it more difficult for novices to learn (mechanics rely first on experiences for diagnosis)
- require learners to make judgments about the nature of the problem, and
- vary significantly in terms of system complexity and dynamicity (age, manufacturer, engine size, reliance on computer controls in the automobile).

Troubleshooting is predominately a cognitive task that includes the search for likely causes of faults through a potentially enormous problem space of possible causes (Schaafstal *et al.*, 2000). In addition to fault detection or fault diagnosis, troubleshooting usually involves the repair or replacement of the faulty device. The emphasis in troubleshooting, though, is on fault diagnosis, which involves a search for the components of the system that are producing substandard outputs (cause of discrepancy). Troubleshooters then search for actions that will efficiently eliminate the discrepancy (Axton *et al.*, 1997).

WHAT KNOWLEDGE AND SKILLS ARE REQUIRED TO TROUBLESHOOT?

Troubleshooting is usually taught as a linear series of decisions that direct the fault isolation. Flowcharts and decision tables are frequently used to lead the novice troubleshooter through a series of actions that will isolate the fault. This approach often works with simple troubleshooting problems, but it is inadequate for training competent or proficient troubleshooters. This section describes the skills that troubleshooters need to develop in order to move from novice, through advanced beginner, toward competent performers (Dreyfus and Dreyfus, 1986). Expertise results from years of reflective practice and is beyond the scope of this article.

In the transition from novice to competent performer, learners construct increasingly rich conceptual (mental) models of the systems they troubleshoot. Those models contain multiple representations of the system. As troubleshooters obtain more experience, they rely less on their conceptual models and more on the events schemas they construct from their experiences. Boshuizen and Schmidt (1992) showed how with experience in medicine, domain knowledge becomes encapsulated in clinical experiences. Schmidt and Boshuizen (1993) showed that acquiring expertise in medicine begins with rich causal networks of biological and pathophysiological processes. Extensive exposure to patient problems embeds their knowledge into higher-level narrative structures referred to as "illness scripts." Illness scripts for automobile mechanics correspond to specific equipment or subsystem malfunctions on particular brands and vintages of cars. Mechanics often describe tendencies for specific parts to fail in cars with different ages or manufacturers. That knowledge is represented as patterns of symptoms that are normally associated with specific fault states in specific cars. Experienced troubleshooters recognize the pattern of symptoms associated with different fault states, which enables the troubleshooter to rapidly activate solution scripts (Besnard and Bastien-Toniazzo, 1999; Gaba, 1991). Those event schemas (e.g., illness scripts) that are used to trigger solutions consist of well-integrated domain knowledge, contextual information, and episodic memories. What makes these event schemas so resistant to decay is the rich contextual information that surrounds the various events.

What kinds of knowledge do novices need to construct during the transition from novice to competent performer? Learning to troubleshoot begins with the construction of a conceptual model for the system that includes domain knowledge, system or device knowledge, visual-spatial knowledge of the system or device, procedural knowledge of how to perform tests and information-gathering activities, and strategic knowledge that guides search activities. We describe each of these knowledge states next. As the troubleshooter gains experience, these knowledge types become embedded within troubleshooters' memories of their experiences. They come to rely more on their historical knowledge of problems they have troubleshot than their conceptual models. Rather than working through a faulty system conceptually, experienced troubleshooters match new problems with their own event schemas resulting from their experiences and apply the solutions from those experiences to solve the current problem (Aamodt and Plaza, 1994). Learning to troubleshoot involves a gradual shift from conceptual knowledge of systems and context-independent knowledge of strategies to personal, context-dependent memories of similar problems.

Knowledge states

Rasmussen (1984a), for example, argued that troubleshooters must understand the device or system they troubleshoot at different levels of abstraction:

- purpose of the system (represented as production flow models, system objectives)
- abstract functional model of the system (represented as causal structure or information flow topology)
- generalized functions of the system (standard functions and processes and control loops)
- physical functions of the system (electrical, mechanical, chemical processes of the components), and
- physical forms in the system (physical appearance and anatomy, material, and form).

The auto mechanic must understand the engine being troubleshot in terms of the location of all of the components; the flow of fuel, air, water, and electricity through those components; and the functions of those flow states and the reasons for changes in them. Without understanding the system being troubleshot on those levels, troubleshooters are unable to generate adequate fault hypotheses. The multiple representations of problems that expert troubleshooters possess allow them to generate more fault diagnosis and solution strategies (Ericsson and Smith, 1991). The following kinds of system knowledge are most generally accepted as essential for troubleshooting.

Domain knowledge

Domain knowledge refers to the general theories and principles upon which the system or device was designed. For example, Ohm's law is a foundation principle used to describe the flow of electricity from the battery, through the starter, and to the spark plugs. Johnson *et al.* (1995) argued that theoretical knowledge may not be as important as educators believe in training competent technical system troubleshooters. Their study found that there was no difference between high and low troubleshooting performers' theoretical knowledge of the system. Students' theoretical knowledge did not predict their competence in troubleshooting a technical system fault (Johnson *et al.*, 1993). Morris and Rouse (1985) concluded that providing instruction about theoretical principles is not an effective way to train troubleshooters. Domain knowledge is a necessary condition for beginning troubleshooters, but it is not sufficient for learning to become a competent troubleshooter. Domain knowledge is important when troubleshooters transfer their skills to different systems (MacPherson, 1998), and domain knowledge is necessary for constructing deeper understanding of the system (Johnson *et al.*, 1995), as reflected in system or device knowledge (described next).

System/Device knowledge

The primary differences between expert and novice troubleshooters are the amount and organization of device knowledge (Johnson, 1988a). Conceptual knowledge of how a systems works is fundamental to the understanding of any technical system (Chi *et al.*, 1981; Johnson *et al.*, 1995; Larkin *et al.*, 1980). System or device knowledge is an understanding of "(1) the structure of the system, (2) the function of the components within the system, and (3) the behavior of those components as they interact with other components in the system" (deKleer, 1985; Johnson and Satchwell, 1993, p. 80), and the flow control within the system (Zeitz and Spoehr, 1989). Again, auto mechanics understand how the components (air, fuel, and electricity) of an automotive system interact with and affect each other. Skilled troubleshooters are better able to troubleshoot outside their specialty because they know how the components of any system work, what their functions are, and how they are related to the system as a whole (Lesgold and Lajoie, 1991).

System knowledge includes topographic and functional knowledge. Topographic models of the system are spatial representations of the components of a system (Rasmussen, 1984b). Topographic knowledge of automobile systems would include representations of the location of each component within the engine or around the automobile. Fuel filters, for instance, can occupy a wide variety of locations within the engine compartment, depending on the manufacturer and model. Rasmussen showed that experts search for faulty components by means of topographic representations of the system being troubleshot (a diagram of the system). In another study, Johnson (1988a) showed that experts reduced problem space size using a topographic search of the system in an efficient sequence. Topographic searches enable skilled troubleshooters to select hypotheses that bring them closer to the fault. Novices meanwhile generate hypotheses randomly within and outside the problem space. Topographic knowledge predicted troubleshooting performance (Rowe and Cooke, 1995; Rowe *et al.*, 1996).

Topographic knowledge is normally conveyed as diagrams depicting the structure of a system. Manufacturing troubleshooters, for example, must hold a mental image of the components of the system and their outputs in order to identify system malfunctions (Axton *et al.*, 1997). More successful topographic models include not only an image or diagram of the physical characteristics of the system but also different information paths or routes through the system. So the search for faults often involves testing the system along these different information paths. Jonassen and Henning (1999) used a method described by Tversky *et al.* (1994) where troubleshooters generate written protocols depicting a visual tour of the system being troubleshot along various routes. More successful troubleshooters provided more accurate topographic descriptions of the systems.

Functional knowledge, as opposed to topographic, is the comprehension of each individual component's function in a given system and the causal relationships between the components and their structure (Sembugmorthy and Chandrasekeran, 1986). For example, functional knowledge of automobile systems includes understanding how spark timing and valve timing both affect combustion. Skilled troubleshooters organize their topographic models based on functional descriptions of the device (Gitomer, 1988). Thagard (2000) analyzed the process of diagnosing disease states and concluded that physicians' explanations of diseases use causal networks to depict the combination of inferences needed to reach a diagnosis. Jonassen *et al.* (1996) constructed a causal network of diagnoses used by physicians diagnosing a hematology disorder (see Fig. 1). When debugging electronics systems, David (1983) found that skilled troubleshooters organize their models around the causal interactions in the electrical system rather than the linear organization of the wiring. David recommends representing the functional organization of the system (how modules interact showing paths of interaction) so novices learn to trace paths of causality, not the physical wire itself. When troubleshooting electronics problems, novices focused on power distribution and the physical layout of radar, whereas experienced troubleshooters used their understanding of the flow of information (Tenney and Kurland, 1988).

Although both topographic and functional representations of relationships provide the troubleshooter with paths to trace while generating hypotheses, novice troubleshooters are more likely to use topographic search strategies, whereas experienced troubleshooters more commonly use functional representations when troubleshooting (Hoc and Carlier, 2000; Rasmussen, 1984a). Troubleshooting strategies based on functional knowledge of the operation of the device lead the troubleshooter to the problem more efficiently.

Performance/Procedural knowledge

Performing troubleshooting tasks, such as measuring voltage or fuel pressure, conducting tests, and making observations of the operation of different parts, involves procedures that must be known and practiced. Knowledge of these activities allows troubleshooters to carry out the operations for performing routine maintenance procedures or testing the components during the troubleshooting process (Hegarty, 1991). Procedural knowledge is specific to the system and the tools used to troubleshoot it. Therefore, its application is limited to that particular content or system (Schaafstal and Schraagen, 1993). Traditionally, mechanics were required to know how to use voltmeters and pressure gauges to test automotive components. Today, they attach engine sensors to a computer that automatically tests the engine's functions.

Strategic knowledge

According to Johnson *et al.* (1995), strategic knowledge plays an essential role in troubleshooting by reducing the problem space, isolating the

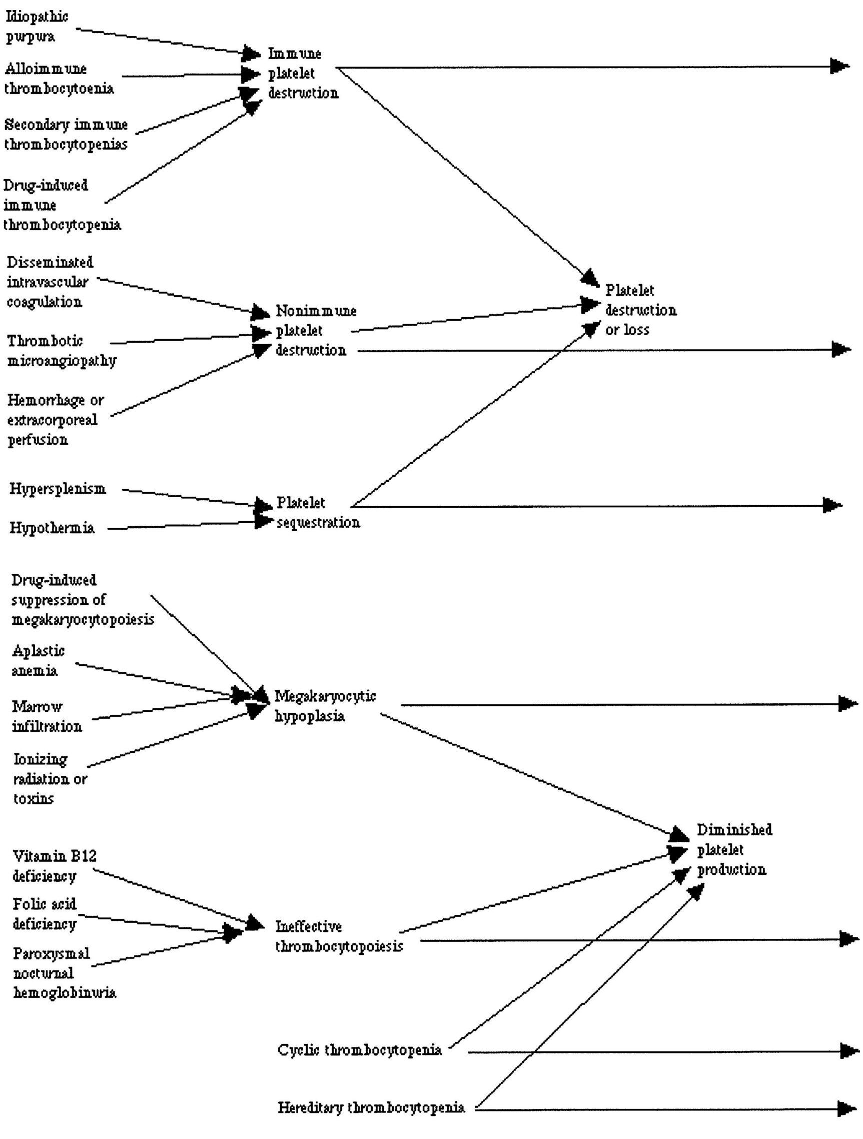
Idiopathic purpura
Alloimmune thrombocytoenia
Secondary immune thrombocytopenias
Drug-induced immune thrombocytopenia
Immune platelet destruction
Disseminated intravascular coagulation
Thrombotic microangiopathy
Hemorrhage or extracorporeal perfusion
Nonimmune platelet destruction
Platelet destruction or loss
Hypersplenism
Hypothermia
Platelet sequestration
Drug-induced suppression of megakaryocytopoiesis
Aplastic anemia
Marrow infiltration
Ionizing radiation or toxins
Megakaryocytic hypoplasia
Diminished platelet production
Vitamin B12 deficiency
Folic acid deficiency
Paroxysmal nocturnal hemoglobinuria
Ineffective thrombocytopoiesis
Cyclic thrombocytopenia
Hereditary thrombocytopenia

Fig. 1 Causal model of medical diagnosis

potential faults, and testing and evaluating hypotheses and solutions. Knowing what part of the electrical systems to test first when diagnosing a car that will not start is important strategic knowledge. Strategic knowledge helps the troubleshooters confirm the hypotheses and solutions they have generated or seek new alternatives when the existing hypotheses or solutions are confirmed false or unfeasible. Schaafstal and Schraagen (1993) classified strategies used in the troubleshooting process as global strategies or local strategies. Global strategies are independent of a specific domain content or system and can be applied across different domains. Local strategies are the ones that are only applicable to a specific content domain or system. Global strategies help the troubleshooter reduce the problem space, whereas local strategies help the troubleshooter conduct the reduction process.

The most prominent global troubleshooting strategy is the serial elimination strategy (start with component nearest the troubleshooter and trace backwards). Because of its inefficiency, this strategy is seldom, if ever, recommended. Johnson (1991) and Brown *et al.* (1975) identified five commonly used global strategies in the troubleshooting process.

1. Trial and error: Randomly attack any section of the system where the possible fault might have occurred. This strategy is most common in the performance of novice troubleshooters.
2. Exhaustive: List all the possible faults and test them one by one until the actual fault is identified. This strategy, similar to serial elimination, is practical only in simple systems.
3. Topographic: Isolate the fault through identifying a series of functioning and malfunctioning checks following the traces through the system. The topographic strategy is usually implemented in two ways, forward or backward. The forward topographic strategy starts the troubleshooting procedure at a point where the device is known to be functioning normally and then works toward the fault by following the system. The backward topographic strategy follows the same procedure but starts at the point of malfunction and then works backward to the input point (Johnson *et al.*, 1995; Newell and Simon, 1972).
4. Split-half: Split the problem space in half and check the functioning condition to determine in which half the fault is located. This method reduces the problem space by confirming the faulty section. The procedure is repeated until the potential faulty area is reduced to a single component. This strategy is efficient when the faulty system is complex and the initial problem space appears to contain several potential faults with no strong indication of where the actual fault lies (David, 1983).
5. Functional/discrepancy detection: Isolate the fault by looking for the mismatches between what is expected in a normal system operation and the actual behaviors exhibited (Brown *et al.*, 1975). By detecting the mismatches, the troubleshooter can identify the

components where the difference is located and, in turn, isolate the actual fault. Performing this strategy requires a thorough integration of system knowledge (especially the interrelationship between functional knowledge and behavioral knowledge).

Little research has compared the effectiveness of domain-general versus domain-specific troubleshooting strategies. Konradt (1995) showed that domain-general strategies, such as split-half and uncertainty rejection, play only minor roles in real life troubleshooting. Experienced troubleshooters rely more on case-based strategies (addressed next), especially in routine failures.

Experiential knowledge

Research studies have confirmed that experience is the most common determinant of expertise, and that the recall of historical information is the most frequent strategy for failure diagnosis (Konradt, 1995). Bereiter and Miller (1989) found that troubleshooters base their diagnosis on their beliefs about the cause once a discrepant symptom is found. Those beliefs are based on historical information (i.e., experience. They also found that the most common reason for taking a particular action during troubleshooting is to test for the most common problem based on experience. Automobile mechanics, for example, often shorten their diagnostic process by applying their historical knowledge of specific fault tendencies in certain models or vintages of cars.

Because of the importance of experiential knowledge, it is essential that learners be required to practice problem-solving tasks. Kyllonen and Shute (1989) recommend troubleshooting a simulated task or "walking through" a performance test. With practice, troubleshooters construct event schemas and rely more on historical information based on experience.

Capacities

In addition to different knowledge states, there are individual differences in experience, cognitive abilities, aptitudes, and cognitive styles related to troubleshooting performance (Morris and Rouse, 1985). Research has focused on only three of those differences: Working memory capacity, causal reasoning, and analytical reasoning (field independence).

Working memory

Working memory is a short-term memory store that enables humans to access and temporarily store information needed to complete a task. Working memory was a predictor of troubleshooting performance (Axton *et al.*, 1997). Troubleshooting performance degrades when working memory is exceeded, which imposes greater cognitive load on the learner (Cooper and Sweller, 1987; Sweller and Cooper, 1985). Cognitive load is intrinsic to the processing demands of the task (Mayer and Moreno, 2003; Paas *et al.*, 2003). The primary cause of cognitive overload is system complexity (Perez, 1991). As systems become more complex, troubleshooting problems place more demands on working memory, and therefore became more difficult to troubleshoot (Allen *et al.*, 1996). More time is required to solve the problems because learners take more actions and repeat more tests. Later, we describe the use of worked examples as an antidote to some aspects of cognitive load.

Causal reasoning

Causal reasoning describes the cognitive abilities required to understand the co-occurrence of cause–effect relationships (Kelley, 1973) and the mechanisms responsible for linking the cause to the effect (Hung and Jonassen, 2006). Causal reasoning enables learners to make predictions, explain relationships, and infer causes. It is an essential skill in solving any kind of problem involving multiple, interacting components, such as identifying causes of discrepancies in system states in order to troubleshoot (Axton *et al.*, 1997). Perkins and Grotzer (2000) found that students engaged in any kind of meaningful learning must move beyond their simplified causal reasoning habits.

Analytical reasoning

Analytical reasoning is another important cognitive capacity for troubleshooting. Analytical reasoning is most often described as field independence, which describes the extent to which the surrounding perceptual field influences a person's perception of items within it. Non-analytical people (field dependents) find it difficult to locate the information they are seeking because the surrounding field masks what they are looking for. Analytical reasoners (field independents) are more adept at disambiguating information from its surrounding field, and therefore are better problem solvers because they are better able to isolate task-relevant information (Heller, 1982; Ronning *et al.*, 1984). In a study of Irish apprentice electricians, Moran (1986) found that among several individual difference variables, field independence was most highly correlated with fault diagnosis and its strongest predictor. This is because analytics (field independents) are more efficient hypothesis testers than field dependents while learning and solving problems (Davis and Haueisen, 1976).

HISTORICAL APPROACHES TO LEARNING TO TROUBLESHOOT

Because troubleshooting is so commonly performed, many instructional approaches have been recommended and explored.

Procedural demonstrations

The default instruction for troubleshooting is to demonstrate a sequence of troubleshooting actions. Students receiving procedural training (step-by-step) performed more accurately and conducted more correct checks than students who received instruction on the system structure (Swezey *et al.*, 1988). However, students receiving instruction about the system structure transferred their learning better than the learners receiving procedural instruction. Demonstrating a sequence of actions can improve performance on the modeled task, but those gains do not transfer to other tasks (Morris and Rouse, 1985). Students following a Fault Isolation Manual that demonstrated required continuity checks on cables, meter reading, switch setting, and device replacement encountered information overload and were unable to explain why they performed the steps (Kurland *et al.*, 1992). Students learn from procedural demonstrations by reproducing operations. If those specific operations fail to reveal the fault, learners who are taught procedurally do not know what to do. They lack the domain principles, system knowledge, and strategic knowledge required to transfer their troubleshooting.

Conceptual (content) instruction

Content approaches to teaching troubleshooting emphasize theoretical and conceptual understanding of the system, removed from any troubleshooting activity. Unfortunately, conceptual understanding of the system alone does not support fault finding (Morris and Rouse, 1985). Students receiving only content instruction perform slower, make more errors, and are less successful in troubleshooting (Morris and Rouse, 1985). Schaafstal *et al.* (2000) found that instructors who teach conceptual content could not troubleshoot or transfer their skills from one radar system to another. Their trainees understood details of system but were unsystematic in their troubleshooting approach.

When used in combination with practice, content instruction should use a breadth-first organization of instruction that starts with an overview and covers the functions of subsystems before describing subsystem components (Zeitz and Spoehr, 1989). The organization of content affects learners' knowledge representation and the degree to which information can be applied in practice.

Related research indicates that the ways that people have learned system-related concepts depends on the job that people perform. Flesher (1993) compared the understandings of design engineers and maintenance technicians and found that designers' understanding emphasizes theoretical concepts when compared with maintenance technicians who actually troubleshoot the systems. In fact, Johnson (1989) found that designers required longer to troubleshoot problems than novices because they were sidetracked by what they perceived as design flaws. Flesher concluded that theory-based approaches to instruction for troubleshooting are not the most effective. Learners lack device knowledge and most of the procedural, strategic, and experiential knowledge required to troubleshoot.

Rule-based approaches

Another prominent approach to teaching troubleshooting requires learners to follow a set of rules for troubleshooting, such as decision trees, flowcharts, or rule-based expert systems that model a series of decisions that troubleshooters use in order to detect faults. These decision aids are often presented as job aids or just-in-time instruction. Rouse *et al.* (1980) developed an expert system rule base for selecting tests when diagnosing three different tasks and compared it with human performance. When they used their rule-base as training, negative transfer resulted. Although novices prefer following rules (Konradt, 1995), learners are not conceptually engaged when they apply rules; they develop inadequate mental models of the system that are required for far transfer.

Other research shows that in troubleshooting practice, rule sequences are abandoned by troubleshooters. When taught how to use search algorithms in real-word diagnostic settings, humans resorted to ad hoc hypotheses (Hoc and Carlier, 2000). Also, it is difficult to reduce an expert technician's actions and knowledge to a set of rules. Experts can easily decide what to do, but they are much less able to provide explicit rules about why they performed as they did (Means and Gott, 1988; Morris and Rouse, 1985). Learners who learn to troubleshoot by following rule-based decision aids lack the domain, device, procedural, and alternative forms of experiential knowledge required to become effective troubleshooters.

Simulations

Troubleshooting instruction often provides practice on simulations of the system being learned. Johnson and Rouse (2001) found that practice on computer simulations resulted in learning that was comparable to traditional lecture and demonstration methods. Much earlier, Johnson and Norton (1992) concluded that simulators alone are insufficient for learning to troubleshoot.

The most prominent issue related to simulator training is the fidelity of the simulation. Johnson and Norton (1992) showed that low-fidelity simulator training should be combined with real equipment or a high-fidelity simulation in order to support learning. Novices need practice on simulators with reasonable fidelity in order to transfer their troubleshooting skills to real equipment. Students trained on simulators with high physical and functional fidelity were able to reach correct solutions more quickly than students using lower fidelity simulators, and they repeated fewer tests (Allen *et al.*, 2001). Functional fidelity is an important determinant of performance.

The most important issue related to fidelity is how accurately the simulator reflects the dynamic interactions within the system. Static simulations of systems are inadequate. In their study of electronics troubleshooting, Park and Gittelman (1992) found that an animated simulator resulted in shorter learning times and fewer trials than a static simulator. Performance on simulators predicts transfer performance on equipment to the degree that the same skills are required (Morris and Rouse, 1985). Therefore, it is essential that transfer of training be evaluated using actual equipment.

Intelligent tutoring systems

Numerous military-funded projects have developed intelligent tutoring systems (ITSs) to teach troubleshooting. These complex systems usually apply an artificial intelligence formalism (e.g., expert systems, neural nets) to represent how an expert thinks (expert model), how a learner performs (student model), and how the instruction should be adapted to the learner's progress (tutorial model). The student model is used to recommend instructional adaptations to individual performance and predict actions of the student based on analysis of a particular problem state (Gitomer *et al.*, 1995). Gitomer *et al.* (1995) built the Hydrive ITS, in which the student model has three components:

1. Action evaluator: Assesses actions in simulation,
2. Strategy interpreter: Assesses strategic understanding, looking for examples of space-splitting, serial elimination, and remove and replace strategies, and a
3. Student profile.

Other examples of ITSs developed to support troubleshooting include Qualitative Understanding of Electric System Troubleshooting (QUEST; Feurzig and Ritter, 1988); Framework for Aiding Understanding of Logical Troubleshooting (FAULT); and MACH-III on radar troubleshooting (Kurland *et al.*, 1992). Mach-III provided animated, physical, and functional diagrams that provide multiple views at different levels; a troubleshooting tree that organizes procedures in functional hierarchy; a troubleshooting advisor that guides mechanics; and an explanation system that provides background information.

ITSs have had different effects on learning to troubleshoot. Johnson *et al.* (1993) developed a technical troubleshooting tutor that supported two troubleshooting activities: Problem space construction and fault diagnosis. They found that students working on the tutor had a 78% improvement in troubleshooting performance with only 19% more practice. Another well-known tutor was SHERLOCK, a computer-coached practice environment for teaching avionics troubleshooting. Its instructional model was based on dynamic assessment of the learner while troubleshooting problems (Lajoie and Lesgold, 1992a). After 20–30 hr of troubleshooting problems, the overall proficiency scores of learners were no better than for trainees receiving on-the-job training (Pokorny *et al.*, 1996). In another study, trainees who used SHERLOCK for 20 hr over 2 weeks performed as well on troubleshooting tasks as experienced technicians (Gott *et al.*, 1993).

ITSs can be effective for training troubleshooters, however, we have some concerns. Most ITSs base their solution paths on an expert model that provides feedback when the learner performs a discrepant action. However, expert models in ITSs do not account for fundamental differences in the ways that novices and experts represent the devices being troubleshot or the diverse strategies that may be used to approach problems. ITSs are also very expensive to build and are system-specific, so they are not applicable to other systems.

Summary

Although numerous instructional approaches for preparing troubleshooters have been developed and researched, none of these instructional approaches have integrated the different knowledge states (especially experiential or historical knowledge) and capacities necessary for learning to troubleshoot. The purpose of this article is to describe a model for designing environments for learning how to troubleshoot that integrates the different knowledge states required to become a proficient troubleshooter. Those environments are based on a cognitive model of troubleshooting, which is described next.

COGNITIVE MODEL OF TROUBLESHOOTING

In order to develop a cognitive model of troubleshooting processes, we begin by reviewing existing conceptions of the troubleshooting process. The simplest conception of troubleshooting is finding the faulty component in a device and repairing or replacing it (Perez, 1991). Troubleshooting requires generating and evaluating hypotheses (Johnson, 1989) and taking corrective action (Schaafstal *et al.*, 2000). According to Schaafstal and Schraagen (2000), troubleshooting consists of four subtasks: Formulate problem description, generate causes, test, and evaluate. Troubleshooting as an iterative process of generating and testing that consists of four subprocesses: Problem space construction, problem space reduction, hypotheses generation/testing (fault isolation/diagnosis process), and solutions generation/verification (Johnson *et al.*, 1993). While troubleshooting, performers:

- use many observations in a sequence of simple decisions;
- use general search procedures that are not dependent on actual system or fault;
- search to find faulty components; and
- search thorough systems to identify appropriate subsystem, state, or component (Rasmussen, 1984a).

According to Axton *et al.* (1997), troubleshooting includes three phases:

(1) Inspection (assessment of the effectiveness of a system by evaluating changes in the characteristics of the system's outputs or components;
(2) Troubleshooting, a search for the components of the system producing substandard outputs cause; and
(3) A search for actions that will fix the discrepancy (cause–behavioral sequence relations or repair).

None of these conceptions, however, addresses the role of previous experience, which is the most frequent strategy for failure diagnosis (Konradt, 1995). Experienced troubleshooters are most efficient because they call on event schemas that are based on the problems they have solved before. So, in order to learn how to troubleshoot, we propose that students must learn how to accomplish the following tasks.

Construct problem space

Constructing problem space is the first step in solving problems (Newell and Simon, 1972). "Problem solving must begin with the conversion of the problem statement into an internal representation" (Reimann and Chi, 1989, p. 165). The problem space of any troubleshooting problem is the mental model of the task environment that the troubleshooter constructs. That model should represent the goal state of the system, the normal states of the system and system components, various fault states, the system structure (including the components of the system and the relationships among the components), the flow control, and a number of potential solution paths (including the most viable one and the possible alternatives). A major difference between proficient and inexperienced repairmen is their ability to conceptualize the problem space (Gitomer, 1988). The best auto mechanics possess rich representations of subsystems for each model and vintage of car they diagnose, and they frequently cite specific fault tendencies for each.

Because they lack system knowledge, novice troubleshooters usually rely on external problem representations. External problem space representations may include flowcharts, schematic diagrams, or functional flow diagrams (Johnson and Satchwell, 1993). Automotive systems are represented as wiring diagrams, exploded views of mechanical systems, and flowcharts of diagnostic procedures. External problem space representations help novice troubleshooters construct internal representations of the system. Later, we describe a multi-layered external problem representation for helping learners that includes topographic description of the system components, functional descriptions of the system flow, normal behaviors of the system components, symptoms or behaviors the system exhibit when operating correctly and faultily, and representations of strategic decisions required during troubleshooting.

Constructing a mental problem space helps troubleshooters to more efficiently isolate the subsystem, component, or device in which the fault is located (Frederiksen and White, 1993). Highly proficient troubleshooters mentally represent the operations of the system in its normal and faulty states (Axton *et al.*, 1997). Because troubleshooters (including both experts and novices) tend not to question their initial problem space once it is established (Johnson *et al.*, 1993), it is essential that learners verify their conceptual understanding whenever troubleshooting actions are taken. Because of rapidly changing systems in automobiles and system differences between different manufacturers, auto mechanics must generate the correct representation of the automobile being diagnosed. Mechanics specialize their work on specific models or manufacturers because they need to construct fewer problem spaces of those complex systems.

Identify fault symptoms

Based on the normal and fault states for system components represented in the problem space, troubleshooters must learn to seek out and recognize faulty components by seeking discrepancies between normal states and existing states of system components. Troubleshooters use strategic knowledge about which procedures to perform in order to identify discrepancies. Recognizing symptoms of faulty components is also aided by experience. The likelihood of symptoms becoming apparent is a function of historical knowledge.

Diagnose fault(s)

After constructing a problem space, the troubleshooter begins the diagnosis process by examining the faulty system and comparing the system states to similar problems that she or he has solved. If a previous problem is recalled, the problem space is reduced immediately to include a description of the old problem.

Experienced troubleshooters categorize problems based on prior experiences. After asking only two questions, the mechanic of one of the authors recently diagnosed a faulty air-flow meter, because those meters are historically the source of problems with the type of automobile being diagnosed. Once, we interviewed an airline maintenance worker attending to a delayed flight, who generated a correct hypothesis about an electrical problem on a DC-9 based on a single symptom, because he "had been working on them for 25 years." The first thing that any experienced troubleshooter does when encountering symptoms is to recall experiences with similar symptoms.

If a previous problem is not remembered and therefore cannot be reused, then the troubleshooter must generate hypotheses by analyzing the initial information collected in order to identify discrepancies between existing states and normal states and by interpreting those discrepancies based on their conceptual model of the system components. Johnson *et al.* (1995) reported that the difference between high- and low-proficient troubleshooters is their ability to correctly interpret the symptoms they have identified. Experts form their initial hypotheses based on the preliminary information acquired during the construction of problem space and the subsequent interpretation (MacPherson, 1998). Newell and Simon (1972) contended that this interdependence is crucial for distinguishing task-relevant and task-irrelevant components within the system. Through this process, initial reduction of the problem space can be achieved by identifying and excluding task-irrelevant components. The next phase is to generate and test potential hypotheses.

Throughout the process of "hypothesis generation and testing" cycles (Johnson *et al.*, 1995, p. 10), the troubleshooters attempt to further narrow the problem space and isolate the potential faults. Johnson (1989) explained that these potential hypotheses are generated to provide possible explanations for the causes of the system fault. Johnson *et al.* (1995, p. 10) classified hypotheses into four levels:

(1) System: The hypotheses conjecture the fault at the system level but do not reduce the problem space beyond the entire equipment or complete system.
(2) Subsystem: The hypotheses conjecture the fault at the subsystem level and reduce the problem space to a discrete subsystem within the complete system.
(3) Device: The hypotheses conjecture the fault at the device level and reduce the problem space to a limited number of components within a subsystem.
(4) Component: The most specific type of hypotheses that conjecture the fault at the component level and result in the identification of a single component as the potential fault cause.

When all potential hypotheses are generated, these hypotheses have to be tested and evaluated (Elstein *et al.*, 1978). After troubleshooters make the initial evaluation, Schaafstal and Schraagen (1993) suggest that troubleshooters prioritize the hypotheses for testing and evaluation based on the likelihood of the cause of the fault and the interdependence level between the component and the symptoms. The process of isolating the fault is a search through the entire system from subsystems, devices, to components in a hierarchical manner in order to identify the cause of the fault.

The process of testing hypotheses is not always linear and straightforward. Rather, it is iterative and recursive. At each level, two possible scenarios may occur. If the high-level hypothesis is correct, then the troubleshooter must be able to continue generating more specific hypotheses about narrower sections of the system until the specific faulty component is found. For example, if a mechanic diagnosed a problem in the fuel system, she/he must generate and test hypotheses about which section of the fuel section is faulty. On the other hand, if the initial, high-level hypothesis is confirmed as incorrect, then the troubleshooter must detect that he or she is heading in the wrong direction and amend the hypothesis and reasoning. Therefore, the ability to evaluate and adjust one's own hypotheses and testing procedures throughout the diagnostic process is critical to becoming an effective troubleshooter. As MacPherson (1998) discovered, when experts found their hypothesis was incorrect, they quickly discarded the false hypothesis and replaced it with an alternative based on the testing results. A key for troubleshooters in using tests results to evaluate their own hypothesis testing process and modifying it if necessary (Means and Gott, 1988).

Generate and verify solutions

The process of solution generation and verification is similar to hypotheses generation and evaluation, although it has not been researched nearly as extensively. The troubleshooter needs to generate one or more solutions for repairing the system based on the results of tests. The simplest solution is to replace a part or module. In many troubleshooting circumstances, that is the preferred solution because it requires the least time. Many contemporary systems are designed so that modules can be easily replaced, because the modules cost less than the troubleshooter's time.

If more than one solution option is generated, then the troubleshooter must select and validate the preferred solution. As with diagnosis, skilled troubleshooters rely first on their experiences. They know that certain solutions are quicker, easier, cheaper, or more reliable. For inexperienced troubleshooters, the solution generation/validation is also an iterative process.

The troubleshooter must select the most plausible solution from the set of solutions generated (Johnson *et al.*, 1993) and determine which best meets all the constraints (e.g., effectiveness, efficiency, system-specifics, or economic consideration). Inexperienced troubleshooters often implement and then test the effectiveness of different solutions. Based on the test results, the inexperienced troubleshooter accepts or rejects the selected solution. This is not the most efficient method of troubleshooting. Experience should eliminate the need for iterative testing.

During the solution evaluation process, the troubleshooter may find that additional information is needed for confirming or disconfirming the selected solution (Frederiksen, 1984). Information may even cause the troubleshooter to reject or modify the original hypothesis or even to revise the initial problem space. Thus, the troubleshooting process is recursive throughout the four phases with adjustment or modification as needed (Johnson, 1989). The solution generation and evaluation process is an essential characteristic in effective troubleshooting (Johnson *et al.*, 1993).

Remember experience

The final step is implicit. Troubleshooters add each troubleshooting experience to their personal case library of experiences. The more difficult or vexing the problem solved, the more likely the problem is remembered (Jonassen and Hernandez-Serrano, 2002).

AN ARCHITECTURE FOR TROUBLESHOOTING INSTRUCTION AND PERFORMANCE SUPPORT

Based on the conception of troubleshooting that we have articulated, we propose the following architecture for designing troubleshooting learning environments (TLEs) to support learning how to troubleshoot (see Fig. 2). This architecture describes the necessary components of computer-based learning and performance support systems for learning to troubleshoot. The architecture describes three main system components (a multi-layered system model, a simulator, and a case library) and two instructional components (worked examples and practice).

Our TLE model assumes that the most effective way to learn to troubleshoot is by solving troubleshooting problems. Learning-to-troubleshoot problems present learners with the symptoms and states of novel problems and require learners to solve them using a simulator. However, successful troubleshooting cannot be learned without adequate system knowledge,

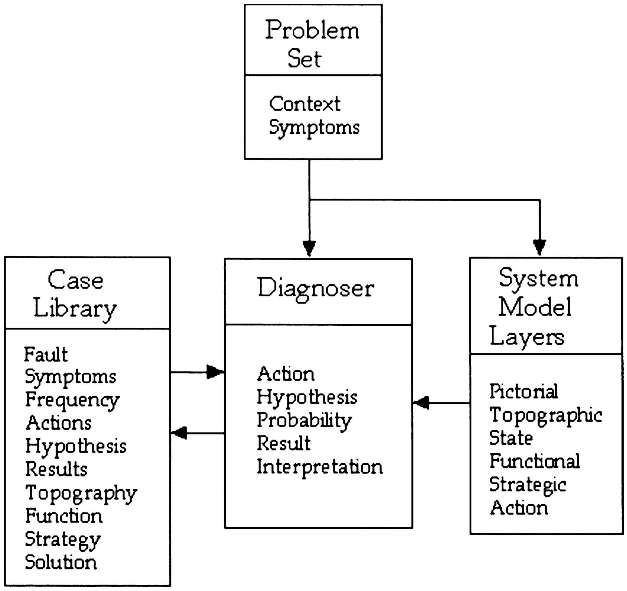

Fig. 2 Architecture for troubleshooting learning environment

so a multi-layered conceptual model of the system is tied to the simulator so that any topographic, functional, procedural, or strategic information about any system component is immediately available while using the simulator. The system model supports conceptual development of device knowledge and support the construction of a mental problem space. Experiential knowledge of troubleshooting is provided by a case library of previously solved problems. We describe each of these components more fully.

System components of TLE

System model

Because novices, advanced beginners, and even competent performers rely on conceptual knowledge of the domain in order to generate hypotheses, it is important that they integrate the different kinds of knowledge of the system being troubleshot into a coherent mental representation. In a series of studies, Kieras and Bovair (1984) showed that a device model illustrating the specific configuration of the components and controls in a device enables learners to infer procedures and learn to operate a device more rapidly. The system model allows learners to view how the system functions (including normal functioning and malfunctioning states) so they can make reasoned diagnoses (which components to test/evaluate based on which hypotheses/solutions). Learners mentally construct problem spaces by selecting and mapping specific relations from a problem domain onto the problem (McGuinness, 1986). In order to do that, multiple kinds of knowledge must be represented in different ways. Rasmussen (1984a) recommended a hierarchy of information types that are needed to diagnose a system, including:

- Functional purposes (production flow models, system objectives)
- Abstract functions (causal structure, information flow topology)
- Generalized function (standard functions and processes, control loops)
- Physical functions (electrical, mechanical, chemical processes of components)
- Physical form (physical appearance and anatomy, material, and form.)

Johnson and Satchwell (1993) showed that providing functional flow diagrams during instruction improved overall system understanding and conceptual understanding of causal behavior. Those diagrams should be simple, showing only the essential components of the system (Johnson and Satchwell, 1993). Therefore, we recommend a system model that integrates multiple, simpler representations of the system that overlay each other. While inspecting any system component on one level, learners can zoom in or out to other layers.

- Pictorial layer contains pictures of the device or system as it exists. Associating representations of the system with the actual system is important (Allen *et al.*, 2001; Johnson and Norton, 1992). Depending on the complexity of the system, pictures of different parts of the system may be necessary. Zooming in from the pictorial layer reveals the topographic layer.
- Topographic layer illustrates the components of the system, their locations, and their interconnections. Topographic representations are important because experts search for faulty components by means of topographic representations of the system (Johnson, 1988a; Rasmussen, 1984a). Zooming in from the topographic layer reveals the state layer.
- State layer provides several overlays to the topographic layer. One overlay conveys normal states or values for each component. These values enable the troubleshooter to compare actual with normal

values in order to determine whether any component is malfunctioning (Patrick, 1993). The symptom overlay conveys symptoms associated with each component malfunction. The probability overlay conveys probabilities of malfunctions or fault states. Being able to match existing symptoms and probabilities with a set of stored symptoms and probable fault states represents a common approach to fault finding (Patrick, 1993). However, Patrick showed that over-reliance on symptoms may result in "tunnel vision" obscuring alternative hypotheses, so the strategic layer provides alternate strategies for diagnosing faults. If the troubleshooter is unaware of the alternative actions, she/he can zoom in on the strategic layer.

- Functional layer illustrates and describes the information, energy, or product flows through the system and how the components affect each other. Understanding system functions is more effective than strategic advice (Patrick and Haines, 1988), however, the combination should be more effective. The learner can zoom from the functional to the strategic layer to identify optional actions and tests.
- Strategic layer consists of rule-based representations of alternative decisions regarding the states described on the state layer. This layer consists of diagnostic heuristics that support fault finding (Patrick and Haines, 1988). Research is needed to determine which method would provide better strategic support during diagnosis. Finally, zooming in from the strategic layer reveals the action layer.
- Action layer includes descriptions of procedures for conducting various tests or operations. The primary purpose of this layer of information is to serve as a job aid or just-in-time instruction for students performing various tests or other actions.

An important rationale for such multi-layered representations in the conceptual model is that they decrease learners' cognitive load, especially while diagnosing problems. The multi-layered conceptual model may provide effective or germane cognitive load (Paas *et al.*, 2003). However, providing an external representation of system components and states scaffolds working memory by off-loading the need to model multiple problem components simultaneously. Being able to move through different layers of a complex conceptual model reduces working memory demands, which can then be applied to diagnosis.

Simulator

The heart of the TLE is the simulator (see Fig. 3). This is where the learner gains experience troubleshooting. The simulator is based on the PARI system of analysis (Hall *et al.*, 1995). After processing a brief story about the behavior and symptoms of the device being troubleshot just before it ceased to work properly, the learner (like an experienced troubleshooter) first selects an action using the pull-down menu at the left of the screen, such as ordering a test, checking a connection, or trying a repair strategy. The novice may be coached about what action to take first based on the symptoms or may select any action. The learner may access the system model at any time in order to see the system and its components in their normal states, how they function, strategic rules for when and how to observe or test the components, how to perform those actions, and the multimodal results from such actions. Jonassen and Henning (1999) showed that refrigeration technicians often rely on different modalities when conversing with machines and tools. Each action taken by the troubleshooter illuminates the corresponding system component in the system model.

For each action the learner takes, the learner is required to select a fault hypothesis that she/he is testing using the pull-down menu to the right of the action menu in the simulator. This is an implicit form of argumentation requiring the learner to justify the action taken. If the hypothesis is

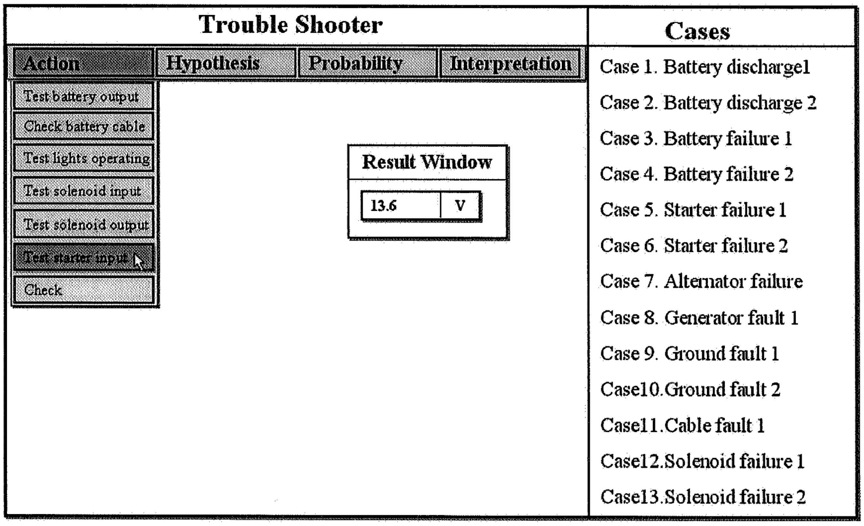

Fig. 3 The simulator

inconsistent with the action, then feedback is immediately provided questioning the rationale for taking such an action. The troubleshooter must also predict the probability that the hypothesis she/he has chosen is actually the fault.

Troubleshooters at all skill levels have difficulty using probabilistic information (Morris and Rouse, 1985). Providing practice in predicting probabilities also acts as a metacognitive prompt depicting the troubleshooter's certainty in hypothesis selection. If the hypothesis or probability is inconsistent with normal states, feedback is provided in a pop-up window and the troubleshooter is required to select another probability. If the hypothesis and probability selected are within normal boundaries, then the troubleshooter sees the results of that action in the results window. Those results may be voltage values, pressure readings, temperature, color of an item, or any other relevant description. The troubleshooter must observe the values in the results window and then select an interpretation of those results using the pull-down menu. An interpretation that is inconsistent with results will also prompt feedback that requires the troubleshooter to select another interpretation.

The simulator is structured to be constructive and performance based. Learners must construct a prediction (a kind of theory) about why the system is not functioning properly) based on system characteristics and then test that theory. Rather than learning about troubleshooting, the learner is engaged in a cognitive apprenticeship (Brown *et al.*, 1989) with a normal troubleshooter. In most troubleshooting tutoring systems, providing feedback usually refers to giving an informative explanation about the correctness of the learners' actions or responses (Frederiksen and White, 1988). In this troubleshooting architecture, the troubleshooter gains competence in interpreting the feedback from the system itself. In real work settings, troubleshooters cannot rely on the feedback from coaches or tutoring systems to see if they are pursuing an appropriate diagnosis. Rather, as Means and Gott (1988) suggested, the troubleshooters need to make decisions for how to proceed to the next step in the troubleshooting process based on the behavioral reactions (feedback) that the system exhibits after the test procedures are completed. In order to troubleshoot independently and competently, troubleshooters must make such judgments on their own.

Second, the simulator enables dynamic assessment of learner performance (Lajoie and Lesgold, 1992b). The actions that a learner takes and the reasons for those actions, both in terms of the hypothesis and interpretation selected, can provide a model of the learner's understanding of the system. The simulator provides clear measures for assessing and evaluating a learner's competence. The number of steps and accuracy of hypotheses and interpretations provides quantitative information about a learner's performance and understanding.

Third, the learner is gaining troubleshooting experience while learning. The results of practice are added to the learner's case library of fault situations, so that the learner can learn from personal experience. Case libraries are described next.

Case library

If the diagnoser is the heart of the TLE, then the case library is the head (memory) of the TLE. Expert's knowledge is primarily derived from cases and concrete episodes (Konradt, 1995), that is, experts use case-based strategies where symptoms observed in previous situations are collected and compared with those in similar and current situations.

The case library or fault database contains stories of as many troubleshooting experiences as possible. Each case represents an indexed story of a context-specific troubleshooting experience. Among technicians, the primary medium of discourse is stories (Jonassen and Henning, 1999). The case library consists of stories about how experienced troubleshooters have solved similar problems that are indexed and made available to learners. Case libraries, based on principles of case-based reasoning, represent one of the most powerful forms of instructional support for ill-structured problems such as troubleshooting (Jonassen and Hernandez-Serrano, 2002). The case library represents the experiential knowledge of potentially hundreds of experienced troubleshooters. In addition to providing potential case problems for solving, the case library can also yield an abundance of conceptual and strategic knowledge that may be included in instruction. When eliciting stories, practitioners naturally embellish their stories with contextual information, heuristics, practical wisdom, and personal identities (Henning, 1996; Schön, 1993). Rather than relying only on a conceptual or theoretical description of a system, when a learner is uncertain about what action to take or what hypothesis to make, the learner may access the case library to gain experience vicariously. The TLE can also be programmed to automatically access a relevant story when a learner commits an error, orders an inappropriate test, or takes some other action that indicates a lack of understanding. Hernandez-Serrano and Jonassen (2003) showed that access to a case library when learning how to solve problems improved complex problem-solving performance.

Building case libraries. In order to analyze stories using CBR, it is necessary first to elicit and capture relevant stories about previously solved problems from practitioners. The goal of capturing stories is to collect a set of stories that are relevant to domain problems and the kinds of information that was relevant to their solution. Relevance to troubleshooting means that the story can provide lessons to the troubleshooter in order to help solve a current problem. In order to collect stories from practitioners, we recommend the following activities.

1. Identify skilled practitioners in the domain. Skilled practitioners are those who have some years of experience in solving problems similar to the ones that you are analyzing.
2. Show the practitioners the problem(s) for which you are seeking support. That is, present one problem at a time. Present the problem to the practitioners. The problem representation should include all of the important components of the problem situation, including contextual information.
3. Ask the practitioners if they can recall any similar problems that they have solved previously. They usually can, so allow them to tell a story about the problem without interruption. Audiotape or (better yet) videotape their recounting of the story. Following their telling of the story, analyze their story with the practitioner.

   An easier method to acquire stories is to have practitioners who troubleshoot document each case they are currently solving. These case descriptions can be dictated or described using a simple survey form. Technicians and professionals alike are normally required to complete paperwork. If you are using that documentation to collect stories, then use a form that is based on index terms described below. Within a matter of months, you should have a substantial case library.
4. The final step in the analysis process is to index the stories. Indexing stories is the primary analytic activity in the process of constructing case libraries. Schank (1990) has argued that the "bulk of what passes for intelligence is no more than a massive indexing and retrieval scheme that allows an intelligent entity to determine what information it has in memory that is relevant to the situation at hand, to search for and find that information" (pp. 84–85). We tell stories with some point in mind, therefore the indexing process clarifies what that point is for a given situation. Indexing is the process of assigning labels to cases at the time that they are entered into the case library (Kolodner, 1993). These indexes are used to retrieve stories when needed by comparing the problem being solved to those stored in the case library.

For each case, identify the relevant indexes that would allow cases to be recalled in each situation. Probable indexes in a troubleshooting case library include:

- Specific fault description
- Initial symptoms observed
- Frequency of occurrence
- Actions, procedures required to isolate faults
- Hypothesis tested
- Results of various tests
- Topographic component
- Functional purpose
- Solution strategies

The result of the elicitation and indexing process is a database of troubleshooting stories. Most of the existing case-based reasoning systems retrieve cases based on a string-match, a full-text search, a rule-based, or a nearest neighbor search algorithm. However, the case-based reasoning system we developed at the University of Missouri has a more robust and accurate retrieval engine using relevance feedback to fine-tune the retrieval engine (see Fig. 4).

Each story (case) is indexed by a case vector that contains a set of attributes. Each attribute has a set of members (values) that function as

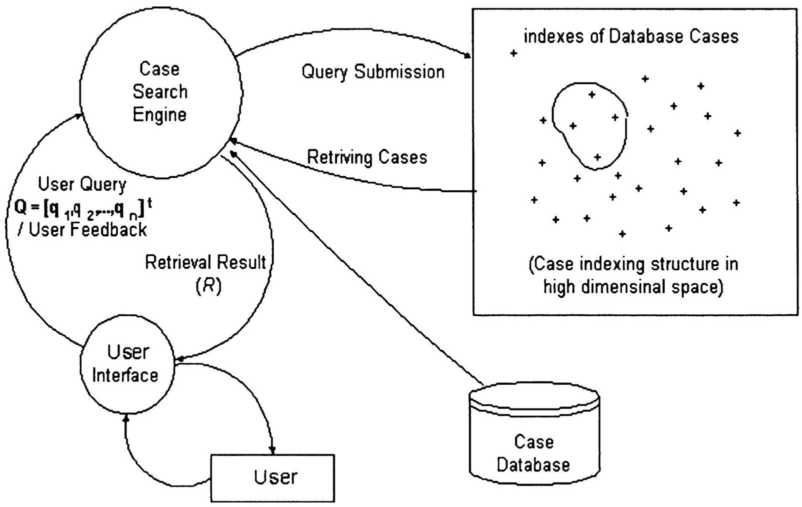

Fig. 4 Case-based reasoning architecture

options for case archival and retrieval. These members are listed in a pull-down menu bar. For some attributes, multiple selections are allowed. The system dynamically creates a query interface for the users by compiling the information stored in an Oracle relational database. It converts the inputs from the users into a query case vector (Q) that is then forwarded to the case search engine. The engine retrieves cases using an advanced nearest-neighbor algorithm. A meaningful distance measurement is the key to the search engine. When two cases are close to each other, a small distance is expected. Therefore, the search engine first computes the distances between a query case and all database cases. It then ranks the distances to determine the order of retrieved cases so that users are prompted with the best matched case first.

Instructional components of TLE

In order to help learners use the TLE, we recommend two essential instructional supports: Worked examples and practice.

Worked examples

Worked examples illustrate how to use the TLE and also model different troubleshooting strategies. If the TLE is entirely online, a pedagogical agent reads the problem symptoms and models strategies for identifying the fault state and symptoms, constructing a model of the problem space or accessing the system model, examining the faulty subsystem, recalling previous cases, ruling out least likely hypotheses, generating and testing hypotheses, interpreting results, and so on. The agent also models how to relate the problem symptoms to system components and relate system components in the troubleshooter to system components in the system model.

Worked examples reduce the heavy cognitive load imposed by the TLE. Integrating multiple representations in the systems model with the experiences of others while also manipulating the simulator imposes heavy demands on working memory (Paas *et al.*, 2003). Worked examples are useful for several reasons. First, splitting attention between multiple information sources interferes with students' acquisition of schemas representing domain concepts (Mwangi and Sweller, 1998; Tarmizi and Sweller, 1988; Ward and Sweller, 1990). Integrating those representations in a multi-layered model reduces that effect. Second, effective worked examples should highlight the subgoals of the problem (Catrambone and Holyoak, 1990). In the case of troubleshooting, those subgoals include identifying fault symptoms, constructing a system model, diagnosing the fault, generating and verifying solutions, and adding experiences to the personal library. This latter subgoal is a form of self-explanation that reduces the need to look back at examples and improves performance (Chi *et al.*, 1989; Chi and Van Lehn, 1991). Worked examples should be used more heavily in the initial stages of skill development (Renkl and Atkinson, 2003). In the latter stages, problem-solving practice is superior because intrinsic cognitive load decreases. Cognitive load decreases as learners develop solution schemas or scripts. As these schemas are constructed, learners better index knowledge and reduce cognitive load even more.

Practice

Practice consists of using the simulator to troubleshoot new problems. During practice, new problems are presented to the learner, who uses the simulator to isolate the cause of the fault. The learner may access the system model or case library in order to understand a system function, determine normal states, or get advice form an experienced troubleshooter. The number of practice problems required to develop different levels of troubleshooting skill is not known. That will depend on the complexity of the system being troubleshot, the abilities and dispositions of the learners, and a host of individual differences. It is worth noting that every action that learners take during their practice can be captured and assessed. The purpose of that assessment may be to track progress during learning or merely to see if the learner is mindfully engaged in the learning process.

Normally, a simple-to-complex practice sequence is recommended. When troubleshooting problems are practiced in a random order, causing high inter-task interference, far transfer improves but not near transfer (De Croock *et al.*, 1998). Learners constructed richer schemata for the system they were troubleshooting, which provided faster, more accurate diagnoses because the learners invested more mental effort during practice. Van Merrienboer *et al.* (2003) recommend two kinds of whole task scaffolds, simple-to-complex versions of the task in order to decrease intrinsic cognitive load and starting with worked examples in order to decrease extraneous cognitive load.

EVALUATING THE TROUBLESHOOTING LEARNING ENVIRONMENT

Because the TLE represents a new approach to troubleshooting instruction, its efficacy can be evaluated only through implementation and research. Prior to that, we provide a rational analysis of system functionality and discuss potential advantages and disadvantages of the architecture.

Table I summarizes a functional analysis of the architecture. Early in the article, we described all of the research-based kinds of knowledge and reasoning required to solve troubleshooting problems. Table I summarizes how each of those knowledge types and capacities are addressed by the TLE architecture. For each TLE component, we identify the nature of the instructional support provided for each kind of knowledge. Architecture components can provide information about a kind of knowledge or skill, engage that knowledge type or skill, scaffold that knowledge type or skill, or model the use of that knowledge type or skill. Note that different architecture components provide nearly every kind of instructional support to every kind of knowledge or skill. That is, there are at least three kinds of instructional support provided for each kind of knowledge or skill required to learn how to troubleshoot problems. That indicates a high level of integration among the components of the environment.

The primary advantage of the TLE is the level of integration in the design of the environment. That integration enables learners to construct conceptual understanding and strategic knowledge through practice. Most researchers have alluded to the necessity of integrating experience, conceptual understanding (system knowledge), and strategic activity. Poor troubleshooters generate more incorrect hypotheses and pursue incorrect hypotheses longer than good troubleshooters; they are less likely to recognize critical information, they make fewer useful tests and more useless tests; they are ineffective in generating hypothesis; and they are poor in executing and verifying the results of their work (Morris and Rouse, 1985). These weaknesses result from poor conceptual understanding of the system they are troubleshooting and from a lack of integration among hypothesis

Table I Functional Analysis of TLE Components

	Theoretical domain	System/ device	Procedural	Strategic	Experiential	Working memory	Causal reasoning	Analytical reasoning
System model	I	I	I	I	I	S	I	
Diagnoser	E	E	I	E	S	S	E	E
Case library		I	I	I	S	S		
Worked examples	D	D	D	D	D	D	D	D
Coaching		E		S			S	S
Practice	E	E	E	E	S	S	E	E

Note. I: Informs, E: Engages, S: Supports, D: Demonstrates.

generation, information gathering (testing), and thinking about the problem. The multi-layered conceptual model provides the conceptual framework for the troubleshooter, in which learners must integrate information with hypotheses and strategies in order to proceed.

The analysis required to construct the TLE is case-based or experience-based. Troubleshooting knowledge is best acquired from experienced troubleshooters. As described before, their job is to recount stories of troubleshooting experiences they have encountered and reflect on the diagnosis and solution. That is a simpler and more reliable cognitive task than trying to convert those experiences into production rules, neural nets, or other formalisms for representing knowledge in intelligent tutoring systems. Often, advisors used to represent expert knowledge in ITSs have no direct experience in troubleshooting the systems they are describing (Johnson, 1988a). Their lack of experience often inhibits their troubleshooting skill (Flesher, 1993). Also, experts can often easily decide what to do, but they are less able to provide explicit rules about similar situations (Means and Gott, 1988).

A potential disadvantage of the TLE architecture is the responsibility that it places on learners. We predict a fairly steep learning curve in the initial stages of learning. Learning to transfer troubleshooting skills really depends on invested mental effort (De Croock *et al.*, 1998). This is the transfer paradox: Instructional strategies that lead to better transfer require learners to work harder or longer before initial performance is acquired. How many cases must be troubleshot before the learning curve begins to level out depends on the complexity of the system and the causal, analytical capacities of the learner.

SUMMARY

Troubleshooting is a cognitive process in which novices begin to learn by constructing and applying conceptual knowledge about a system. With experience troubleshooting real cases, competent practitioners encapsulate relevant domain knowledge and contextual information into high-level diagnostic scripts (Schmidt and Boshuizen, 1993). Diagnosis for proficient performers and experts becomes a classification activity where troubleshooters search their event schemas in order to recognize different fault states. Learning to troubleshoot represents a shift from conceptual understanding of the system to an experiential understanding of the process.

In this article, we have described an architecture for developing environments to support learning how to transition from conceptual knowledge of a system to experiential knowledge. This architecture formalizes the essential role of experience in learning to troubleshoot using a case library of troubleshooting stories and a case-based reasoning engine to access descriptions of relevant troubleshooting experiences. The environment also integrates a diagnostic simulator with a multi-layered conceptual representation of the system being troubleshot. As we have shown, the knowledge that is constructed by troubleshooters in training moves from domain to device to experiential knowledge. This architecture is unique because it integrates all of those knowledge representations and the activities that help construct each kind of knowledge. Although this architecture has not yet been empirically validated, it provides a unique blueprint for developing online troubleshooting learning environments and performance support systems, and for researching the relative importance of those conceptual components across different kinds of troubleshooting problems.

REFERENCES

Aamodt, A., and Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications* 7(1), 39–59.

Allen, J. A., Hayes, R. Y. T., and Buffardi, L. C. (2001). Maintenance training simulator fidelity and individual differences in transfer of training. In Sweezey, R. W., and Andrews, D. H. (eds.), *Readings in Training and Simulation: A 30-Year Perspective*, Human Factors Society, Santa Monica, CA, pp. 272–284.

Allen, J. A., Terague, R. C., and Carter, R. E. (1996). The effects of network size and fault intermittency on troubleshooting performance. *IEEE Trans. Syst. Man Cybern.* 26(1): 125–132.

Axton, T. R., Doverspike, D., Park, S. R., and Barrett, G. V. (1997). A model of the information-processing and cognitive ability requirements for mechanical troubleshooting. *Int. J. Cogn. Ergon.* 1(3): 245–266.

Bereiter, S. R., and Miller, S. M. (1989). A field study of computer-controlled manufacturing systems. *IEEE Trans. Syst. Man Cybern.* 19: 205–219.

Besnard, D., and Bastien-Toniazzo, M. (1999). Expert error in troubleshooting: An exploratory study in electronics. *Int. J. Hum.-Comput. Stud.* 50: 391–405.

Boshuizen, H. P. A., and Schmidt, H. G. (1992). The role of biomedical knowledge in clinical reasoning by experts, intermediates, and novices. *Cogn. Sci.* 5: 121–152.

Brown, J. S., Burton, R. R., Bell, A. G. (1975). SOPHIE: A step toward creating a reactive learning environment. *International Journal of Man-Machine Studies* 7(5): 675–696.

Brown, J. S., Collins, A., and Duguid, P. (1989). Situated cognition and the culture of learning. *Educ. Res.* 18: 32–42.

Catrambone, R. C., and Holyoak, K. J. (1990). Learning subgoals and methods for solving probability problems. *Memory Cogn.* 18(6): 593–603.

Chi, M. T. H., Bassock, M., Lewis, M. W., Reiman, P., and Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cogn. Sci.* 13: 145–182.

Chi, M. T. H., Feltovich, P. J., and Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cogn. Sci.* 5: 121–152.

Chi, M. T. H., and Van Lehn, K. A. (1991). The content of physics self-explanations. *J. Learn. Sci.* 1(1): 69–105.

Cooper, G., and Sweller, J. (1987). Effects of schema acquisition and rule automation on mathematical problem solving. *J. Educ. Psychol.* 79: 347–362.

David, R. (1983). Reasoning from first principles in electronic troubleshooting. *Int. J. Man-Mach. Stud.* 19: 403–423.

Davis, J. K., and Haueisen, W. C. (1976). Field independence and hypothesis testing. *Percept. Motor Skills* 43: 763–769.

De Croock, M. B. M., van Merrienboer, J. J. G., and Paas, F. G. W. C. (1998). High versus low contextual interference in simulation-based training of troubleshooting skill: Effects of transfer performance and invested mental effort. *Comput. Hum. Behav.* 14(2): 249–267.

deKleer, J. (1985). How circuits work. In Bobrow, D. G. (ed.), *Qualitative Reasoning About Physical Systems*, MIT Press, Cambridge, MA, pp. 205–280.

Dreyfus, H. L., and Dreyfus, S. E. (1986). *Mind Over Machine*, The Free Press, New York.

Elstein, A. S., Shulman, L. S., and Sprafka, S. A. (1978). *Medical Problem Solving: An Analysis of Clinical Reasoning*, Harvard University Press, Cambridge, MA.

Ericsson, K. A., and Smith, J. (1991). Prospects of the limits of the empirical study of expertise: An introduction. In Ericson, K. A., and Smith, J. (eds.), *Toward A General Theory of Expertise: Prospects and Limits*, Cambridge University Press, New York, pp. 1–38.

Feurzig, W., and Ritter, F. (1988). Understanding reflective problem solving. In Pstoka, J., Massey, L. D., and Mutter, S. A. (eds.), *Intelligent Tutoring Systems: Lessons Learned*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 435–450.

Flesher, J. W. (1993). An exploration of technical troubleshooting expertise in design, manufacturing, and repair contexts. *J. Ind. Teach. Educ.* 31(1): 34–56.

Frederiksen, N. (1984). Implications of cognitive theory for instruction in problem solving. *Rev. Educ. Res.* 54(3): 363–407.

Frederiksen, J. R., and White, B. Y. (1988). Implicit testing within an intelligent tutoring system. *Mach.-Mediat. Learn.* 2: 351–372.

Frederiksen, J. R., and White, B. Y. (1993). The avionics job-family tutor: An approach to developing generic cognitive skills within a job-situated context. In Artificial Intelligence in Education: *Proceedings of AI-ED 93*, *World Conference on Artificial Intelligence in Education* (pp. 513–520), Edinburgh, Scotland.

Gaba, D. (1991). Dynamic decision making in anesthesiology: Cognitive models and training approaches. In Evans, D. A., and Patel, V. (eds.), *Advance Models of Cognition for Medical Training and Practice*, Springer-Verlag, Heidelburg FRG, pp. 123–147.

Gitomer, D. H. (1988). Individual differences in troubleshooting. *Hum. Perform.* 1(2): 111–131.

Gitomer, D. H., Steinberg, L. S., and Mislevy, R. J. (1995). Diagnostic assessment of a troubleshooting skill in an intelligent tutoring system. In Nichols, P. D., Chipman, S. F., and Brennan, R. L. (eds.), *Cognitively Diagnostic Assessment*, Lawrence Erlbaum Associates, Hillsdale, NJ.

Gott, S. P., Hall, E. P., Pokorny, R. A., Dibble, E., and Glaser, R. (1993). A naturalistic study of transfer: Adaptive expertise in technical domains. In Detterman, D., and Sternberg, R. (eds.), *Transfer on Trial: Intelligence, Cognition, and Instruction*, Ablex Publishing, Westport, CT, pp. 258–288.

Hall, E. P., Gott, S. P., and Pokorny, R. A. (1995). *A procedural guide to cognitive task analysis: The PARI methodology* (Tech. Report AL/HR-TR-1995–0108). Brooks Air Force Base, TX: Human Resources Directorate.

Hegarty, M. (1991). Knowledge and processes in mechanical problem solving. In Sternberg, R. J., and Frensch, P. A. (eds.), *Complex Problem Solving: Principles and Mechanisms*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 253–185.

Heller, L. C. (1982). An exploration of the effect of structure variables on mathematical word problem-solving achievement (Doctoral dissertation, Rutgers University, 1982). *Dissert. Abstr. Int.* 44: 416.

Henning, P. H. (1996). *A Qualitative Study of Situated Learning by Refrigeration Service Technicians Working for a Supermarket Chain in Northeastern Pennsylvania.* Unpublished Ph.D. dissertation, The Pennsylvania State University, Pennsylvania.

Hernandez-Serrano, J., and Jonassen, D. H. (2003). The effects of case libraries on problem solving. *J. Comput.-Assist. Learn.* 19.

Hoc, J. M., and Carlier, X. (2000). A method to describe human diagnostic strategies in relation to the design of human-machine cooperation. *Int. J. Cogn. Ergon.* 4(4): 297–309.

Hung, W., and Jonassen, D. H. (2006). Conceptual understanding of causal reasoning in physics. *International Journal of Science Education* 28(5): 1–21.

Johnson, S. D. (1988). Cognitive analysis of expert and novice troubleshooting performance. *Perform. Improv. Q.* 1(3): 38–54.

Johnson, S. D. (1989). A description of experts and novice performance differences on technical troubleshooting tasks. *J. Ind. Teach. Educ.* 26: 19–37.

Johnson, S. D. (1991). Training technical troubleshooters. *Tech. Skills Train.* 27(7): 9–16.

Johnson, S. D., Flesher, J. W., and Chung, S.-P. (1995, December). *Understanding troubleshooting styles to improve training methods*. Paper presented at the Annual Meeting of the American Vocational Association, Denver, CO. (ERIC Document Reproduction Service No. ED 389 948).

Johnson, S. D., Flesher, J. W., Jehng, J. C., and Ferej, A. (1993). Enhancing electrical troubleshooting skills in a computer-coached practice environment. *Interact. Learn. Environ.* 3(3): 199–214.

Johnson, S. D., and Satchwell, R. E. (1993). The effect of functional flow diagrams on apprentice aircraft mechanics' technical system understanding. *Perform. Improv. Q.* 6(4): 73–91.

Johnson, W. B., and Norton, J. E. (1992). Modeling student performance in diagnostic tasks: A decade of evolution. In Regian, J. W., and Shute, V. J. (eds.), *Cognitive Approaches to Automated Instruction*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 195–216.

Johnson, W. B., and Rouse, W. B. (2001). Training maintenance technicians for troubleshooting: Two experiments with computer simulations. In Sweezy, R. W., and Andrews, D. H. (eds.), *Readings in Training and Simulation: A 30-Year Perspective*, Human Factors Society, Santa Monica, CA.

Jonassen, D. H. (2000). Toward a design theory of problem solving. *Educ. Technol. Res. Dev.* 48(4): 63–85.

Jonassen, D. H., and Henning, P. (1999). Mental models: Knowledge in the head and knowledge in the world. *Educ. Technol.* 39(3): 37–42.

Jonassen, D. H., and Hernandez-Serrano, J. (2002). Case-based reasoning and instructional design: Using stories to support problem solving. *Educ. Technol. Res. Dev.* 50(2): 65–77.

Jonassen, D. H., Mann, E., and Ambruso, D. J. (1996). Causal modeling for structuring case-based learning environments. *Intell. Tutor. Media* 6(3/4): 103–112.

Kelley, H. H. (1973). The process of causal attribution. *Am. Psychol.* 28: 107–128.

Kieras, D. E., and Bovair, S. (1984). The role of a mental model in learning to operate a device. *Cogn. Sci.* 8: 255–273.

Kolodner, J. (1993). *Case-Based Reasoning*, Morgan Kaufman, New York.

Konradt, U. (1995). Strategies of failure diagnosis in computer-controlled manufacturing systems. *Int. J. Hum.-Comput. Stud.* 43: 503–521.

Kurland, L. C., Granville, R. A., and MacLaughlin, D. B. (1992). Design, development, and implementation of an intelligent tutoring system for training radar mechanics to troubleshoot. In *Intelligent Instruction by Computer: Theory and Practice*, Taylor & Francis, Washington, DC.

Kyllonen, P. C., and Shute, V. J. (1989). A taxonomy of learning skills. In Ackerman, P. L., Sternberg, R. J., and Glaser, R. (eds.), *Learning and Individual Differences*, Lawrence Erlbaum Associates, Hillsdale, NJ.

Lajoie, S. P., and Lesgold, A. M. (1992a). Dynamic assessment of proficiency for solving procedural knowledge tasks. *Educ. Psychol.* 27(3): 365–384.

Lajoie, S. P., and Lesgold, A. (1992b). Apprenticeship training in the workplace: Computer-coached practice environment as a new form of apprenticeship. In Farr, M. J., and Psotka, J. (eds.), *Intelligent Instruction in Computer: Theory and Practice*, Taylor & Francis, Washington, DC.

Larkin, J., McDermott, J., Simon, D. P., and Simon, H. A. (1980). Expert and novice performance in solving physics problems. *Science* 208: 1335–1342.

Lesgold, A., and Lajoie, S. (1991). Complex problem solving in electronics. In Sternberg, R. J., and Frensch, P. A. (eds.), *Complex Problem Solving: Principles and Mechanisms*, Lawrence Erlbaum Associates, Hillsdale, NJ.

McGuinness, C. (1986). Problem representation: The effects of spatial arrays. *Memory Cogn.* 14(3): 270–280.

MacPherson, R. T. (1998). Factors affecting technological trouble shooting skills. *J. Ind. Teach. Educ.* 35(4): 5–28.

Mayer, R. E., and Moreno, R. (2003). Nine ways to reduce cognitive load in multimedia learning. *Educ. Psychol.* 38(1): 43–52.

Means, B., and Gott, S. P. (1988). Cognitive task analysis as a basis for tutor development: Articulating abstract knowledge representation. In Psotka, J., Massey, L. D., and Mutter, S. A. (eds.), *Intelligent Tutoring Systems: Lessons Learned*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 35–57.

Moran, A. P. (1986). Field independence and proficiency in electrical fault diagnosis. *IEEE Trans. Syst. Man Cybern.* SMC-16(1): 162–165.

Morris, N. M., and Rouse, W. B. (1985). Review and evaluation of empirical research in troubleshooting. *Hum. Factors* 27(5): 503–530.

Mwangi, W., and Sweller, J. (1998). Learning to solve compare word problems: The effect of example format and generating self-explanations. *Cognit. Instr.* 16: 173–199.

Newell, A., and Simon, H. A. (1972). *Human Problem Solving*, Printice-Hall, Englewood Cliffs, NJ.

Paas, F., Renkl, A., and Sweller, J. (2003). Cognitive load theory and instructional design: Recent developments. *Educ. Psychol.* 38(1): 1–4.

Park, O. K., and Gittelman, S. S. (1992). Selective use of animation and feedback in computer-based instruction. *Educ. Technol. Res. Dev.* 40(4): 27–38.

Patrick, J. (1993). Cognitive aspects of fault-finding training and transfer. *Le Travail Humain* 56(2/3): 187–209.

Patrick, J., and Haines, B. (1988). Training and transfer of fault-finding skill. *Ergonomics* 31(2): 193–210.

Perez, R. S. (1991). A view from troubleshooting. In Smith, M. U. (ed.), *Toward a Unified Theory of Problem Solving*, Lawrence Erlbaum Associates, Hillsdale, NJ.

Perkins, D. N., and Grotzer, T. A. (2000, April). *Models and Moves: Focusing on Dimensions of Causal Complexity to Achieve Deeper Scientific Understanding*. Paper presented at the American Educational Research Association Annual Conference, New Orleans, LA.

Pokorny, R. A., Hall, E. P., Gallaway, M. A., and Dibble, E. (1996). Analyzing components of work samples to evaluate performance. *Mil. Psychol.* 8(3): 161–177.

Prochaska, J. O., DiClemente, C. C., and Norcross, J. C. (1992). In search of how people change: Applications to addictive behaviors. *Am. Psychol.* 47(9): 1102–1114.

Rasmussen, J. (1984a). Strategies for state identification and diagnosis in supervisory control tasks, and design of computer-based support systems. In Rouse, W. B. (ed.), *Adv. Man-Mach. Syst. Res.*, 1: 139–193.

Rasmussen, J. (1984b). *Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering*, North-Holland, Amsterdam.

Reimann, P., and Chi, M. T. H. (1989). Human expertise. In Gilhooly, K. J. (ed.), *Human and Machine Problem Solving*, Plenum, New York, pp. 161–191.

Renkl, A., and Atkinson, R. K. (2003). Structuring the transition from example study to problem solving in cognitive skill acquisition: A cognitive load perspective. *Educ. Psychol.* 38(1): 15–22.

Ronning, McCurdy, and Ballinger (1984, January). Individual differences: A third component in problem-solving instruction. *J. Res. Sci. Teach.* 21(1): 71–82.

Rouse, W. B., Pellegrino, S. J., and Rouse, S. H. (1980). A rule-based model of human problem solving performance in fault diagnosis tasks. *IEEE Transactions on Systems Man, and Cybernetics*, SMC-10, 366–376.

Rowe, A. L., and Cooke, N. J. (1995). Measuring mental models: Choosing the right tool for the job. *Hum. Resour. Dev. Q.* 6(3): 243–262.

Rowe, A. L., Cooke, N. J., Hall, E. P., and Halgren, T. L. (1996). Toward an on-line knowledge assessment methodology: Building on the relationship between knowing and doing. *J. Exp. Psychol. Appl.* 2(1): 31–47.

Schaafstal, A., and Schraagen, J. M. (1993). The acquisition of troubleshooting skill implication for tools for learning. In Brouwer-Janse, M. D., and Harrington, T. L. (eds.), *Human-Machine Communication for Educational Systems Design*, Springer-Verlag, New York, pp. 107–118.

Schaafstal, A., and Schraagen, J. M. (2000). Training of troubleshooting: A structured, task analytical approach. In Schraagen, J. M., Chipman, S. F., and Shalin, V. L. (eds.), *Cognitive Task Analysis*, Lawrence Erlbaum Associates, Mahwah, NJ, pp. 57–70.

Schaafstal, A., Schraagen, J. M., and van Berlo, M. (2000). Cognitive task analysis and innovation of training: The case of structured troubleshooting. *Hum. Factors* 42(1): 75–86.

Schank, R. C. (1990). *Tell Me a Story: Narrative and Intelligence*, Northwestern University Press, Evanston, IL.

Schmidt, H. G., and Boshuizen, H. G. A. (1993). On acquiring expertise in medicine. *Educ. Psychol. Rev.* 5(3): 205–221.

Schön, D. A. (1993). *The Reflective Practitioner—How Professionals Think in Action*, Basic Books, New York.

Sembugmorthy, V., and Chandrasekeran, B. (1986). Functional representations of devices and compilation of diagnostic problem-solving systems. In Kolodner, J., and Riesbeck, C. K. (eds.), *Experience*, *Memory*, *and Reasoning*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 47–53.

Sweller, J., and Cooper, G. A. (1985). The use of worked examples as a substitute for problem solving in learning algebra. *Cognit. Instr.* 2: 59–89.

Swezey, R. W., Perez, R., and Allen, J. (1988). Effects of instructional delivery system and training parameter manipulations on electromechanical performance. *Hum. Factors* 30(6): 751–762.

Thagard, P. (2000). *Coherence in Thought and Action*, MIT Press, Cambridge, MA.

Tarmizi, R. A., and Sweller, J. (1988). Guidance during mathematical problem solving. *J. Educ. Psychol.* 80: 424–436.

Tenney, Y. J., and Kurland, L. C. (1988). The development of troubleshooting expertise in radar mechanics. In Psotka, J., Massey, L. D., and Mutter, S. A. (eds.), *Intelligent Tutoring Systems: Lessons Learned*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 59–83.

Tversky, B., Franklin, N., Taylor, H. A., and Bryant, D. J. (1994). Spatial mental models from descriptions. *J. Am. Soc. Inf. Sci.* 45(9): 656–669.

Van Merrienboer, J. J. G., Kirschner, P. A., and Kester, L. (2003). Taking the load off a learner's mental mind: Instructional design for complex learning. *Educ. Psychol.* 38(1): 5–13.

Ward, M., and Sweller, J. (1990). Structuring effective worked examples. *Cognition and Instruction*, 7(1): 1–39.

Zeitz, C. M., and Spoehr, K. T. (1989). Knowledge organization and the acquisition of procedural expertise. *Appl. Cogn. Psychol.* 3: 313–336.