Proceedings of the Fourth World Conference on Engineering Education
St. Paul, Minnesota; October, 1995.

STRUCTURING ON-THE-JOB TROUBLESHOOTING PERFORMANCE TO AID LEARNING

Brian Minsk
Center for Human-Machine Systems Research
School of Industrial & Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332
E-mail: minsk@chmsr.gatech.edu

Harinarayanan Balakrishnan and Ashwin Ram
College of Computing
Georgia Institute of Technology
Atlanta Georgia 30332
E-mail: {harib,ashwin}@cc.gatech.edu

This paper describes a methodology for aiding the learning of troubleshooting tasks in the course of an engineer's work. The approach supports learning in the context of actual, on-the-job troubleshooting and, in addition, supports performance of the troubleshooting task in tandem. This approach has been implemented in a computer tool called WALTS (Workspace for Aiding and Learning Troubleshooting). This method aids learning by helping the learner structure his or her task into the conceptual components necessary for troubleshooting, giving advice about how to proceed, suggesting candidate hypotheses and solutions, and automatically retrieving cognitively relevant media. WALTS includes three major components: a structured dynamic workspace for representing knowledge about the troubleshooting process and the device being diagnosed; an intelligent agent that facilitates the troubleshooting process by offering advice; and an intelligent media retrieval tool that automatically presents candidate hypotheses and solutions, relevant cases, and various other media. WALTS creates resources for future learning and aiding of troubleshooting by storing completed troubleshooting instances in a self-populating database of troubleshooting cases. The methodology described in this paper is partly based on research in problem-based learning, learning by doing, case-based reasoning, intelligent tutoring systems, and the transition from novice to expert. The tool is currently implemented in the domain of remote computer troubleshooting.

Introduction

Sophisticated devices and systems, such as computers, are widespread and growing in use. With the growth of complex technology an imperative need for efficient and effective troubleshooting skills has developed. Troubleshooting technologically sophisticated devices is often a difficult, complex process to carry out and learn: efficient, effective troubleshooting generally requires intimate knowledge of complicated devices and proficient reasoning skills and strategies.

Computing technology has advanced to the point where tools supporting both the performance and learning of complex engineering tasks such as troubleshooting are now possible. Many troubleshooters employ advanced computing technology in the course of their jobs. These computers are often used to help troubleshooters *carry out* their jobs, but the same computers are typically not used to help troubleshooters *learn* to do their jobs better. While these computers could be used to provide training separate from the performance of troubleshooting, a more effective approach may be to use these computers to support learning "on-the-job," while professionals are engaged in actual troubleshooting. Learning from direct experience, or "learning by doing," has been acknowledged as an effective educational approach [6].

This paper describes a methodology for aiding the learning of troubleshooting tasks in the course of a professional engineer's work. This methodology has been realized in a computer tool called WALTS (Workspace for Aiding and Learning TroubleShooting). WALTS is currently implemented in the domain of remote computer troubleshooting. Each component of this methodology and its implementation in the WALTS tool is described in the following sections. The Summary section provides a synopsis of the methodology involved in producing WALTS and its relation to previous work.

Structuring Troubleshooting to Aid Learning

Because of the complexity and constant innovation of sophisticated technology, professional troubleshooters must perpetually be learning about their domain. To become more efficient and effective performers, troubleshooters must also continually add to and refine their reasoning skills and strategies. A particular challenge for aiding on-the-job learning is to

support learning without harming performance. Ideally, such a methodology should produce benefits for both performance and learning. The methodology described in this paper is based on structuring the performance of troubleshooting such that both performance and learning are enhanced in tandem.

The WALTS system includes three major components for structuring troubleshooting (see Figure 1). One, the Cognitive Troubleshooting Workspace and the Causal Links Workspace allows troubleshooters to represent knowledge about the troubleshooting process and the device being diagnosed. Next, the Facilitator is an intelligent agent that facilitates the troubleshooting process by offering advice. Finally, an intelligent media retrieval tool called the Media Retrieval Workspace automatically presents candidate hypotheses and solutions, relevant cases, and various other media.

Representing the Conceptual Structure of Troubleshooting

The methodology described in this paper involves representing knowledge in a form that is functional for troubleshooting. Representing troubleshooting knowledge entails separating the knowledge used in troubleshooting into conceptual components, specifying supporting and disconfirming relations between the components, and specifying causal relations between the components. In the Cognitive Troubleshooting Workspace of the WALTS system, a troubleshooter can document, structure, classify and display the knowledge used in the troubleshooting and create and display supporting or disconfirming relationships between objects. In the Causal Links Workspace troubleshooters can create and display causal relations between objects.

The knowledge used in the troubleshooting process is separated into its conceptual components. Five conceptual components are identified and described below: symptoms, configuration data, state data, hypotheses, tests, and solutions.

- *Symptoms*. Symptoms are the set of system behaviors someone considers problematic that the troubleshooter is trying to correct and, perhaps, find causes for. Example symptoms for a computer system might include the

system's inability to make a network connection or format a disk. For a human, symptoms might include a rash or chest pains.

- *Configuration data*. Configuration data includes the relatively stable parts of a system. For a computer system this might include hardware and software. For a human, this might include sex, weight, or age.
- *State data*. State data is the set of relatively inconstant system behaviors (not symptoms) and the settings or values of changeable system variables. Examples for a computer system might include a condition when a computer is connecting to some devices on a network but not others or the setting of some variable in a configuration file. Examples for a human might include heart rate, blood pressure, or blood alcohol level.
- *Hypotheses*. Hypotheses are the possible causes of a set of symptoms. For a computer, this might be a broken piece of hardware or a wrong setting of some system variable. For a human, this might be a tumor, disease, or dietary problem.
- *Tests*. Tests are the actions taken to demonstrate the validity of a hypothesis or solution. Tests can be implemented by changing the value of a system variable or adding or subtracting components from the system. The result is a set of (hopefully) observable system behaviors. If a troubleshooter suspects a network card is the cause of a symptom in a computer, a test might include removing the network card to see if the symptom behavior is still displayed by the computer. An example test on a human might include implementing a low salt diet to validate a hypothesis that high sodium levels are the cause of an individual's high blood pressure.
- *Solutions*. Solutions are the actions taken to correct a set of symptoms. Solutions can be implemented by changing the value of some system. A solution may be the same as a test: however, solutions are specific to changing the behaviors of the system that are expressed as the set of symptoms. An example solution for a computer system includes replacing a bad network card when a computer system is not connecting to the network. A solution for a human condition might include removing a malignant brain tumor to restore normal brain function.

These conceptual components explicitly structure the knowledge used to carry out the troubleshooting task in a way that supports the development and evaluation of hypotheses and solutions. Each fragment of knowledge used in the troubleshooting process is classified with these conceptual components into “knowledge objects”. An explicit representation of the knowledge and process for troubleshooting supports learning [4].

To evaluate hypotheses and solutions in this methodology, “supporting” or “disconfirming” relations and causal relations can be established between the knowledge objects used for troubleshooting. A supporting relation indicates that a knowledge object is confirming evidence for a hypothesis or solution. Similarly, a disconfirming relation denotes negative evidence for a hypothesis or solution. With supporting and disconfirming relations between knowledge objects, a troubleshooter links and assesses hypotheses and solutions directly with system data. Causal relations between knowledge objects are used to represent the causal chains involved in generating symptoms. Understanding the causes of symptoms is important for generating hypotheses and solutions and for developing causal knowledge about a device.

Troubleshooters can create and classify their knowledge by creating text windows in the Cognitive Troubleshooting Workspace. Each text window identifies the type of knowledge object it is in the title bar, the knowledge itself (typed in by the troubleshooter), and any supporting or disconfirming relationship with an active text window. Supporting and disconfirming relations are created by dragging and dropping between text windows and specifying the type of relation in a dialog box. The bar at the bottom of the text window is displayed as green if it has a supporting relationship with the active window, red if disconfirming, and gray if no relation has been specified by a troubleshooter.

Troubleshooters can use another workspace in WALTS called the Causal Links Workspace to create causal links between objects. Users drag objects from the Cognitive Troubleshooting Workspace into the Causal Links Workspace. In the Causal Links Workspace users can drag and drop objects to other objects to create causal relations that are represented by arrows.

Intelligently Facilitating the Troubleshooting Process

Learning reasoning skills and strategies involved in troubleshooting complex devices is also supported by "facilitating" troubleshooters' reasoning to make it more efficient and effective. Facilitating involves actively offering timely advice while troubleshooters carry out the troubleshooting task. For example, if a troubleshooter generates a large number of hypotheses, advice to try to eliminate hypotheses or to evaluate the most likely hypotheses is given immediately. Or, if a troubleshooter tries to implement a solution that has disconfirming evidence, the system suggests the troubleshooter avoid a solution that has negative evidence.

Advice can and sometimes should be ignored by troubleshooters. As troubleshooters learn to make their reasoning more effective and efficient, advice should become less obtrusive. In the present methodology troubleshooters can control how obtrusive the advice is such that it can go from being totally intrusive to the troubleshooting process to providing no advice at all.

In the WALTS system, advice about how to make the troubleshooting process effective and efficient is given by an intelligent agent called the Facilitator. The Facilitator actively observes the Cognitive Troubleshooting Workspace. When it encounters a condition in which it can give advice, it offers this advice to the user. Advice to the user can be presented with a beep, with a flash of the screen, in a dialog box that the user must clear from the screen, in a simple text window (as displayed in Figure 1), not at all, or any combination of the above as controlled by users.

Intelligently Retrieving Relevant Information

The methodology described in this paper embodies four major goals for providing information for the learning troubleshooter. First, information furnished by the tool to troubleshooters serves their specific problem-solving and learning goals [4]. Next, access to manuals, technical specifications, instructions, and past troubleshooting cases is provided and involves little or no effort on the part of the troubleshooter. Third, retrieved media are labeled with a "cognitive media type" which classifies the media's potential

utility for problem-solving and learning [5]. Finally, hypotheses and solutions based on past troubleshooting cases are suggested when appropriate.

The Media Retrieval Workspace in WALTS provides the troubleshooter's interface to media retrieval. The Media Retrieval Workspace automatically retrieves media based on the current contents of the Cognitive Troubleshooting Workspace. Users can also initiate manual searches in the Media Retrieval Workspace. WALTS has access to large free-form text databases of previously completed troubleshooting cases relevant to its domain and various multimedia, on-line manuals, instructions, and technical specifications. WALTS also has access to previous troubleshooting cases accomplished with the WALTS tool. Retrieved media is labeled with its cognitive media type, its physical media type (e.g. text, picture, sound or movie), the type of WALTS object or the database from which it comes, a short description, and a relevance metric.

The structured knowledge furnished by the Cognitive Troubleshooting Workspace allows for efficient, directed information retrieval and storage. The Media Retrieval Workspace's automatic retrieval uses the structure of the knowledge contained in the Cognitive Troubleshooting Workspace to weight and prune searches. These directed searches help overcome a traditional problem of retrieving more media from a large database than a user can handle. Each retrieved media includes a display of the Cognitive Troubleshooting Workspace categories from which it was retrieved. Relevance metrics are computed and displayed to represent the quality of the matched media. In addition, previous WALTS cases are stored in a structured format allowing for much more efficient and useful access than is normally allowed with free-form text databases. If a Hypothesis or Solution is retrieved from a previous WALTS case, the user can simply double-click on the the retrieved object in the Media Retrieval Workspace to add it to the Cognitive Troubleshooting Workspace. Users can also display a complete previous WALTS case retrieved with the Media Retrieval Workspace.

Summary

In the domain of WALTS, the troubleshooters' learning goals are to acquire knowledge about the devices being diagnosed and to improve reasoning skills and strategies associated with troubleshooting. The problem-solving

goals are to generate a successful explanation (hypothesis) and remedy (solution) for a device's problems. The methodology described in this paper attempts to provide support for both sets of goals in tandem by providing a structured workspace to document the troubleshooting process, an intelligent "facilitator" with troubleshooter-controlled obtrusiveness to make the process more effective and efficient, and automatically furnished information that is useful and relevant.

The emphasis in the methodology used to produce WALTS is to structure the "doing" of troubleshooting such that performance and learning of troubleshooting is enhanced. Expert problem-solvers represent knowledge according to abstract principles relevant for problem-solving [2]. With the structure of the Cognitive Troubleshooting Workspace, WALTS attempts to "jump-start" this type of abstract classification for novices. Like the SHERLOCK system [3], WALTS provides an intelligent facilitation environment for learning anchored in experience. Unlike SHERLOCK, in which the training is separated from performance, WALTS is embedded in the actual performance situation. As in problem-based learning environments, the methodology described in this paper grounds knowledge about the devices being diagnosed in the performance situation itself and supports the development of diagnostic reasoning skills [1].

Acknowledgments

This research was supported by a grant from the Digital Equipment Corporation and by the Edutech Institute. The authors thank Greg Heathcock, Chris Hill, and Anil Rewari for their contributions to this research and Alan Chappell and Jennifer Turns for their helpful comments on an earlier draft of this manuscript.

References

[1] Barrows, H. S. (1986). A taxonomy of problem-based learning methods. *Medical Education, 20*.

[2] Chi, M. T. H., Feltovich, P. J., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science, 5*.

[3] Lajoie, S. P. & A. Lesgold (1989). Apprenticeship training in the workplace: computer-coached practice environment as a new form of apprenticeship. *Machine-Mediated Learning, 3*.

[4] Ram, A., S. Narayanan, & M. T. Cox (1995). Learning to troubleshoot: multistrategy learning of diagnostic knowledge for a real-world problem solving task. *Cognitive Science, 19*.

[5] Recker, M. M., A. Ram, T. Shikano, G. Li, & J. Stasko (In press). Cognitive media types for multimedia information access. *Journal of Educational Multimedia and Hypermedia*.

[6] Soloway, E., M. Guzdial, & K. E. Hay (1994). Learner-centered design: the challenge for HCI in the 21st century. *Interactions*, April, Association of Computing Machinery.

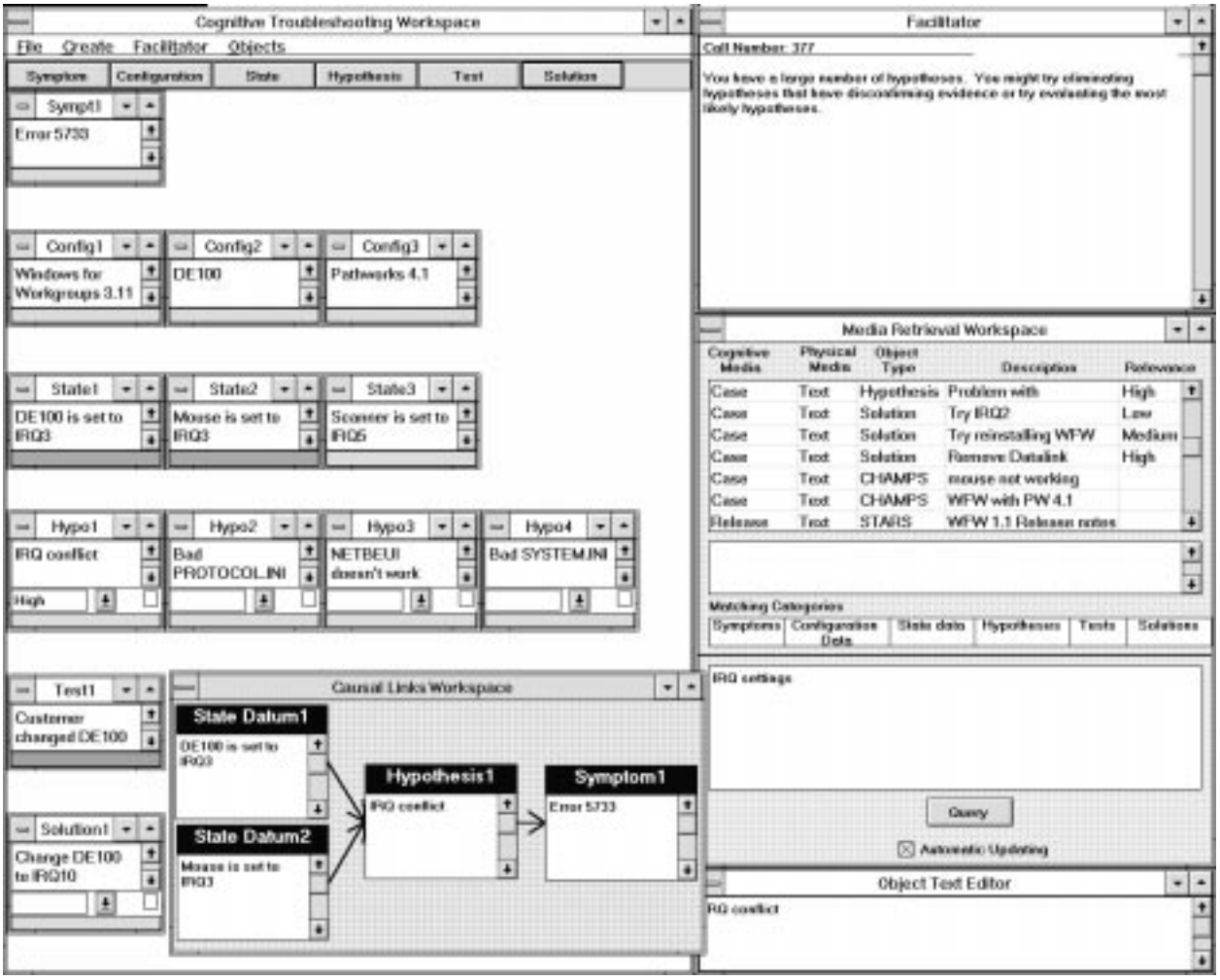

Figure 1: Sample WALTS Screen