Domain, Task, and User Models for an Adaptive Hypermedia Performance Support System

Peter Brusilovsky
School of Information Sciences
University of Pittsburgh
Pittsburgh PA 15260
peterb@mail.sis.pitt.edu

David W. Cooper
Antech Systems Inc.
1214 Progressive Dr., Suite 101
Chesapeake, VA 23320 USA
dcooper@antechsystems.com

Abstract
Electronic Performance Support Systems (EPSS) is a challenging application area for developing intelligent interfaces. Some possible scenarios for using domain, task, and user models for adaptive performance support were explored in the context of the Adaptive Diagnostics and Personalized Technical Support (ADAPTS) project. ADAPTS provides an intelligent, adaptive EPSS for maintaining complex equipment.

Keywords
Performance support, task model, user model, domain model, adaptive hypermedia, adaptive presentation.

INTRODUCTION

Modern advanced diagnostic systems can tell a technician what is wrong or what is about to go wrong in a system; in some cases, they can even identify where the problem lies. Modern interactive electronic technical manuals (IETM) provide a wealth of information about a system: how it is constructed, operates, and what do in a case of each particular problem. The focus of the ADAPTS, an electronic performance support system (EPSS) for maintenance technicians [5] integrates adaptive guidance from diagnostics systems with adaptive access to technical information, thus supporting both sides of the process: what-to-do and how-to-do-it. ADAPTS is a comprehensively adaptive system. It adjusts the diagnostic strategy to who the technician is and what the technician is doing, dynamically adapting the sequence of setups, tests, and repair/replace procedures based on the technician's responses. New activities are planned depending on the technician's responses to current recommended activities. ADAPTS assembles information content on the fly in response to the steps of that diagnostic process. The technician receives dynamically selected technical support information appropriate for the contexts of the setup, test, and remove/replace procedure being performed. Key to the adaptive functionality is knowledge about the domain, maintenance tasks, and a user represented in domain, task, and user models. The domain and the task models provide the framework for structuring the content of IETM and representing the user knowledge. The user model determines what task to do, what technical information to select to describe the task, and how to best display that information for a given technician.

The goal of this paper is to present in detail the models used in the ADAPTS system and demonstrate how integrated task and domain models can be used to build an adaptive performance support system. ADAPTS has a number of others interesting design aspects, but they are left outside the scope of this paper. For more information about ADAPS project the reader can consult [5]

ADAPTS: THE USER'S VIEW

The cycle of work with ADAPTS consists of two main sub-processes: adaptive diagnostics and adaptive interaction with a technician performing a task. The adaptive diagnostics in the ADAPTS prototype is performed by a modified version of a toolset [6] developed by Qualtech Systems, Inc. (QSI). On each step of the diagnostic process, the diagnostic engine selects the most relevant task for the user to perform. This task is a chunk of work that is variable in size and complexity, which consists of a sequence of subtasks such as various operations on equipment and checking measurable and observable parameters. The interaction with a technician performing a task is maintained by an adaptive hypermedia interface that provides adaptive guidance though the sequence of subtasks and adaptive presentation of relevant material for each performed subtask. The user interacts with the adaptive hypermedia engine through a standard Web browser. The results of the user's work with the task (confirmation that all subtasks are completed, results of the observations, or failure to perform the tasks) are passed to the diagnostic engine. Depending on the results, the diagnostic engine dynamically selects the next task to perform and starts the next cycle of work.

The adaptive hypermedia interface consists of two main windows – the outline frame (left frame on figure 1) and the content presentation frame (right frame on figure 1). Each frame can present several types of information. The user can

IUI'02, January 13-16, 2002, San Francisco, California, USA.

select the desired type of information using named tabs on the top of each frame.

The main function of the outline frame is to provide an adaptive checklist of the task being performed. The adaptive task checklist helps each technician navigate through computer-presented maintenance information by suggesting an optimal path and indicating the current state of performing the task. It applies several adaptive navigation support techniques.

What differentiates adaptive navigation support from traditional hypermedia-based performance support is the customization to each technician's knowledge, experience, and preferences. Typical hypermedia systems identify a predefined course through technical information. ADAPTS, on the other hand, dynamically defines a unique course each time it presents technical information. In this case, the adaptive diagnostics components serves as the expert technician, driving the troubleshooting strategy based on a dynamic assessment of time, effort, payoff, resources, and a specific technician's knowledge and experience with a specific troubleshooting scenario. In determining a recommended course of troubleshooting, the expert model's consultation with the user model ensures a match between what needs to be done and what the technician has the ability to do. Not only is the recommended course of action geared toward a specific technician, but also the directions associated with performing that action under the current set of circumstances.

ADAPTS uses a collapsible checklist of steps to guide the technician through a troubleshooting procedure. ADAPTS determines how to present this checklist based on a dynamic assessment of the user's expertise with that procedure. For example, ADAPTS collapses a subtask outline if the technician is experienced with the subtask. Inexperienced technicians automatically receive an expanded outline of subtasks that reveals details. Experienced technicians may expand the outline if they choose, and are given greater flexibility to navigate within the checklist. Inexperienced technicians are given more assistance in step-by-step navigation. As a technician completes a step within the checklist, color-coding and icons identify completed, current, and remaining steps.

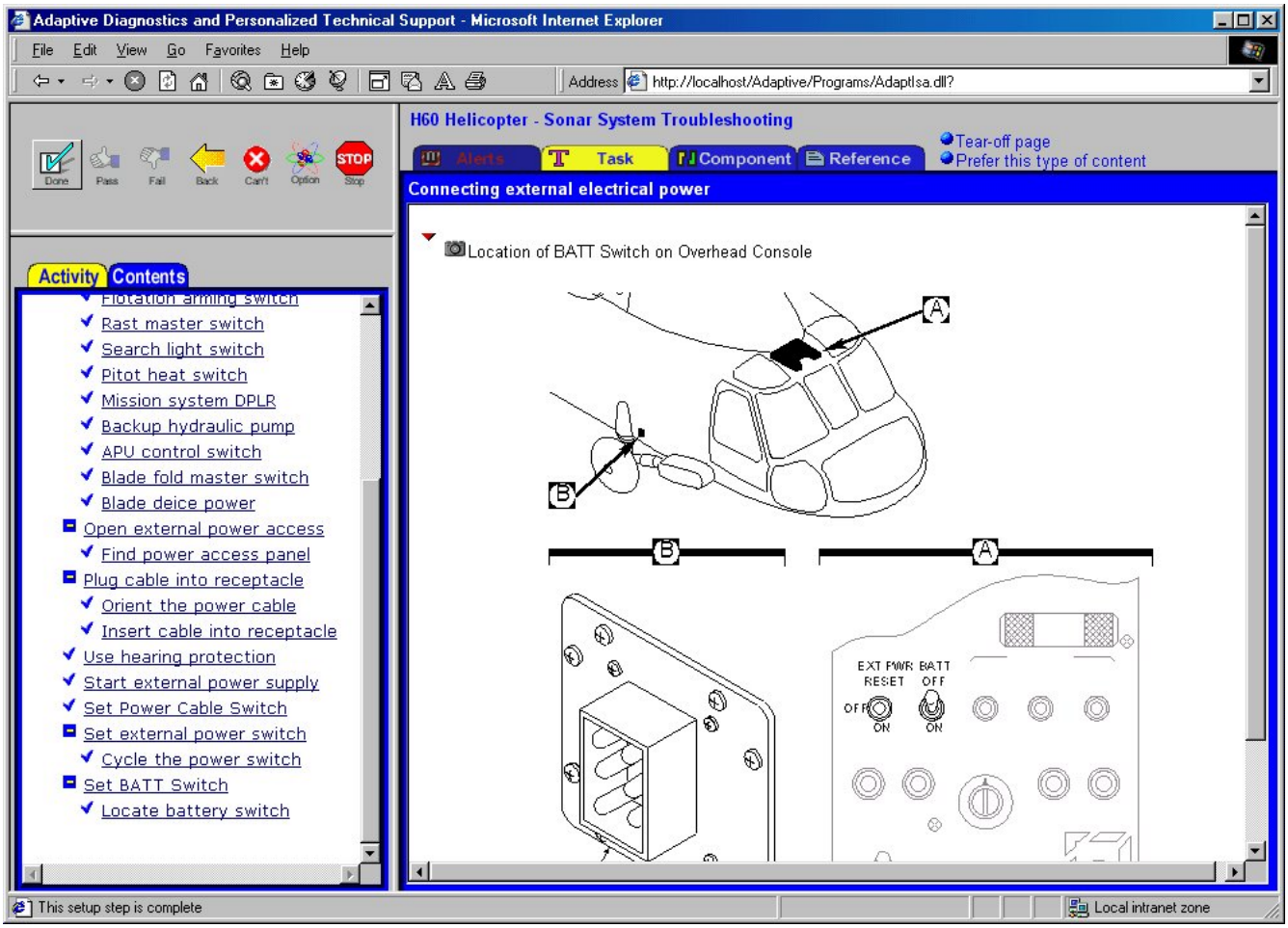

Figure 1. The ADAPTS interface consists of the adaptive outline frame (left), adaptive content presentation frame (right), and an applet for communication with adaptive diagnostic engine (top left)

The duty of the content presentation frame is to display the relevant support information for the subtask selected in the outline frame. The problem here is that the amount of potentially relevant information could be very big and it's a serious challenge for a technician to find the information that is most suitable to his experience and context of work. While several navigation "tabs" are provided to classify the support information into several types and present each type in a separate window, the amount of information in each of these windows is still potentially too big. To provide further support ADAPTS uses an adaptive hypertext presentation technique called stretchtext [4] to present a sequence of paragraphs of support information.

Stretchtext expands and collapses procedural paragraphs to reveal or hide details, similar to the expanding and contracting outline used for the procedural checklist (Figure 4). The use of stretchtext in ADAPTS is similar to its use in such systems as MetaDoc [2] or PUSH [15]. For example, a particular paragraph could be collapsed in the default presentation if a technician is familiar with the information presented in this paragraph or if this information is not very relevant to the current context. It could also be collapsed if a technician is experienced with current procedural subtasks. The technician is free to expand and contract the stretchtext at will. To support the use of procedural information, the navigation component also custom-selects links to supporting information that will be offered to each technician. Technicians who are inexperienced with a step will be offered links to fundamental concepts, background information, and training segments (such as video clips or simulations). Experienced technicians will be offered links to more concise information that omits fundamentals that have already been mastered. Because the user model is continuously updated, the navigation path continuously adapts to the technician's changing level of expertise.

ADAPTS not only custom-selects links for a technician, it also cues the technician to the relevance of the links that are offered. Cues may be visual, such as different icons or different colors; textual, such as annotations or comments; or sorting, which places the most relevant links at the top of a list. Furthermore, the tabs are used to categorize information, and the content available under each tab is adaptive. Regardless of the technique used, the goal remains the same—guiding the technician to custom-selected support information that is not only applicable to the current context, but also appropriate given the technician's expertise.

THE DOMAIN AND TASK MODELS

The key to the intelligent performance of ADAPTS is the representation of knowledge about the domain (a system to troubleshoot) and the maintenance tasks to be performed. The domain model mirrors the hierarchy of systems, subsystems, and components in the target system. The hierarchy starts with the whole system (i.e., an aircraft) and follows traditional decomposition of the system down to the elementary components.

ADAPTS distinguishes three types of components in the domain model. An *addressable unit* is the lowest level type of a component. Any part of the system that is referred directly by the manual and has to be viewed or manipulated by technicians is an addressable unit. Usually, no other user knowledge about an addressable unit may be anticipated but the knowledge where it is located.

A replaceable unit is the second level type of a component. Any component that could be replaced (i.e., removed and installed) is a replaceable unit. In addition to the knowledge about its location, we could say about a replaceable unit that a user has or has not experience in removing/installing it, has or has not experienced it broken, etc.

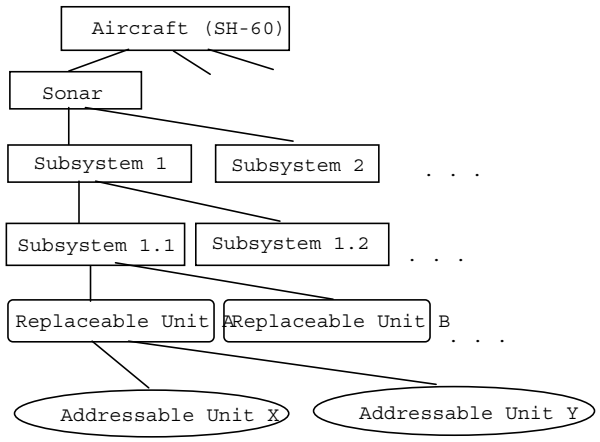

Figure 2: The structure of the domain model

A *system (subsystem)* is the highest-level type of component. The difference between a system and a replaceable unit (or a set of replaceable units) is that the system consists of several sub-components that interact with each other, so there is a theory of operation related to the system and we could measure the user's knowledge about it. A system also constitutes the lowest troubleshooting level. No troubleshooting is possible within a replaceable unit.

The knowledge about the maintenance tasks is represented in a form of task model. The task model in ADAPTS consists of a relatively large set of maintenance tasks hierarchically composed of sub-tasks and steps. The upper levels of task hierarchy are used by both components of ADAPTS: the diagnostic engine and the adaptive hypermedia interface. The lower levels of the hierarchy are used by the adaptive interface only; for the purpose of performance support some elementary diagnostics task are broken into subtasks and steps. Each subtask is a piece that is meaningful for a technician and supported by a reasonably-sized description in the IETM.

There are procedural subtasks and logical subtasks. A procedural subtask represents some part of work to be done for performing the task. A logical subtask may be required to provide references to a special kind of information. For example, if some task requires the user to do A, B, C, and D (in this order), then doing A, doing B, doing C, or doing D will be procedural subtasks. Viewing precautions, task overview, or required instruments involved (typical task components in an IETM) are logical subtasks.

The components of the task hierarchy are connected with the components of the domain hierarchy by relationship "involve" (i.e., "subtask 29 involves RU19 and RU290"). Each task or subtask is connected to all domain model components that are essentially involved in performing this task (Figure 3). In other words, if a lower or higher-level component is manipulated (tested or repaired) on a subtask, there is a link between them. Connections between tasks and components are made on the highest possible level. For example, if whole task T10 involves troubleshooting of system S78, then the connection should be made between T10 and S78, not between any subtasks of T10 and S78.

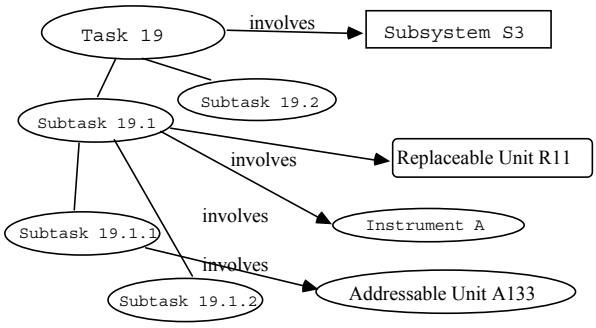

Figure 3. Task hierarchy and its connection to components

IETM CONTENT INDEXING

To support the user in performing a diagnostic task ADAPTS uses a variety of information types stored in its database (see Figure 4) which are collectively called *rich content*. In addition to textual documents and diagrams, which are typical components of IETMs, the rich content could include various pieces of multimedia: color photos, training videos, animations, and simulations. Moreover, the rich content could include variations of the same information fragment oriented to the users with different levels of experience. One of the functions of ADAPTS is to find pieces of the rich content that are relevant to the selected subtask, and to adaptively present it to the user. This functionality depends on the links between the domain model and the rich content. Establishing a connection between documents (i.e., IETM pieces) and the domain model is usually called indexing. Indexing is a key to both user modeling and adaptation. Various kinds of indexing applied in adaptive hypermedia systems are reviewed in [3].

ADAPTS applies two kinds of indexing of the rich content. The first type is role-based indexing of fragments with components. Conceptually, it means that a piece of the rich content is linked by typed links with all components involved in it, while the type of links indicate the type of involvement (i.e., its role). For example, a piece of video that shows how to remove a component is indexed with a pair (component ID, role="Component Illustrated in Removal and Installation"). Similarly, a figure that shows the location of a component is indexed with a pair (component ID, role="General Component Location").

The second kind of indexing refers to tasks and subtasks. The reason for indexing a fragment of the rich content with a task or a subtask is that the fragment explains how to perform the task or provide some other task-supporting information (i.e., "Listing of Materials Used in Task", "Discussion of Interferences"). For each piece of rich content, both the type of explanation and the level of explanation differ according to several factors. The type of explanation differs according to the material available for the content, such as text, figure, animation, or video, and its purpose. The level of explanation differs according to the estimated ability of the user to comprehend that material (for example, a reminder oriented to an expert who has done this task many times, or a complete description for a technician who has never performed the task). Tasks are usually indexed as a one-to-one relationship with a set of rich content dealing with a specific concept or topic. The specific rich content that is accessed from the set (to support each step in the task) depends on the user model. In contrast, a one-to-many indexing scheme is used with components so the technician sees many links as optional navigation paths.

In general, roles are used to identify the context within which a certain concept (component, system, task) appears. These roles are categorized in various ways so that the adaptive engine can make decisions on how and where the content will be displayed in the interface. ADAPTS uses an elaborated set of roles developed by domain experts. There are 13 component-related and 17 task-related roles. Note that roles are by nature dependent on the subject matter, and can be added as needed during authoring.

THE USER MODEL

The user model is the source for personalizing the content and navigation in ADAPTS. The core of user modeling approach in ADAPTS is estimating technician's experience with system tasks and components of various levels – from subsystems to addressable units. The experience is calculated from various evidences of user experience collected by the system. To maximize the user modeling power ADAPTS uses a multi-aspect overlay user model. A technician's experience with a component or a task is judged on many aspects, each weighted to indicate its relative influence on the decision. The user model independently accumulates several aspects (roles) of the experience and knowledge of each technician for each component or task. In total, there are 12

aspects for evaluating user experience with components and 8 aspects for evaluating user experience with tasks (Table 1). Aspects used in ADAPTS include whether and how often a technician has reviewed, observed, simulated, expressed understanding (self- evaluation), previously worked on, or received certification on specific equipment or procedures.

The aspects of the user model are designed to map one-to-one with user actions. In other words, if the user was troubleshooting a subsystem in a simulator, the corresponding simulation counter will be updated; if he removed a component, the corresponding hands-on counter will be updated; if he watched an indicator or turned a switch, yet another counter will get an increase. Whatever is done that is relevant is immediately reflected in the user model. Note that there are two different types of counters: those that record computer-based experience (i.e., user read the text or seen a movie) and those that record real experience (user turned real switch or installed a real component). Accordingly, we have two ways of watching the user and updating counters: tracing what the user is doing on the computer with the adaptive interface (requested pages, figures, movies), and "watching" what the user is doing with the real system. Both ways of user modeling are made possible due to rich content indexing presented in the previous subsection of this paper.

ADAPTS can easily trace what the user is reading or watching when working with different parts of IETM. If a page is requested, we assume it is read (with some probability). If a movie was requested, we assume the user watched it. The connections between IETM pages and the domain model enable the system to update relevant aspects for the involved components of the domain and task models. For example, if a piece of graphic material indexed with a pair "component=C88, role="General Component Location", then the review counter Au_m for C88 (seen the location in a picture) is incremented, and so on. If all IETM material is properly indexed, it is easy to update the user model after each visited page, figure, animation, or movie. For each component related to the visited piece of the IETM, the system will increment the proper model counter using the table of roles and the types of rich content. Similarly, indexing of IETM components with tasks and subtasks enables the system to update the aspect counters for the "tasks" part of the user model. ADAPTS can't really "watch" what the user is doing with the aircraft, but with ADAPTS performance support interface where the user should confirm (or reject) that he has completed a subtask or the whole task, we can reliably update the user model by watching what tasks or subtasks the user reported as done.

The update of the aspect counters for the "tasks" part of the user model is straightforward: the corresponding counter for the performed task or subtask is simply incremented (or

Table 1. User Characterization Aspects. These aspects are used to characterize the ability of users. Each aspect is weighted according to its importance in determining overall ability.

UCF ID	Applicability	Action Type	Weight	Comments
1	addressable_unit	review	1	see the location in a picture
2	addressable_unit	observe	2	find the unit in real life
3	replaceable_unit	review	2	review info about component in IETM
4	replaceable_unit	observe	3	watch someone do a task related to this component
5	replaceable_unit	simulation	4	do a simulation involving this primary component
6	replaceable_unit	hands_on	5	work on this component
7	subsystem	review	2	review info about this subsystem in IETM
8	subsystem	observe	3	watch someone work on this subsystem
9	subsystem	simulation	4	do a simulation involving this subsystem
10	subsystem	hands_on	5	work on this subsystem
11	subsystem	certification	6	user has been certified on this subsystem
12	subsystem	self_eval	4	user thinks he understands this subsystem
13	task	review	2	review this entire task in IETM
14	task	observe	3	watch someone do this specific task
15	task	simulation	4	interactive simulation of this specific task
16	task	hands_on	5	did this specific task
17	task	certification	6	have been certified on this specific task
18	task	self_eval	3	user thinks he can do this specific task
19	step	review	1	see this step in IETM
20	step	hands_on	2	did this step successfully

decremented if the user can't perform the task). The update of the historic "components" part is done using connections between tasks and components. For example: if the performed activity is a removal of component C66, a hands-on counter for C66 has to be updated. If the performed activity implies that the user watched the indicator I99, then the "observe" counter for I99 has to be updated. If the performed activity implies troubleshooting of a system S6, then a counter for S6 has to be updated. Different levels of activities are to be used to update action-based historic counters for different levels of components.

The adaptive hypermedia component does not use the multi-aspect historic model directly. Instead, it uses scalar values that estimate the proficiency of a user in locating, operating, and repairing equipment or performing each step of a recommended procedure. To move from a historic multi-aspect model to the needs of the adaptive hypermedia interface, ADAPTS uses a simple weighted polynomial. The weights represent relative importance of different components of user's experience, and were set by domain experts.

The user model continues to evolve as a technician uses the ADAPTS system, beginning with a stereotype that seeds the model for technicians with certain backgrounds. No formal test is used to initialize the model. The model grows in size as it records the technician's experience. It will follow a pattern of rapid expansion initially as new material is accessed for the first time, followed by decreasing growth rate up to a limit imposed by the extent of the domain model.

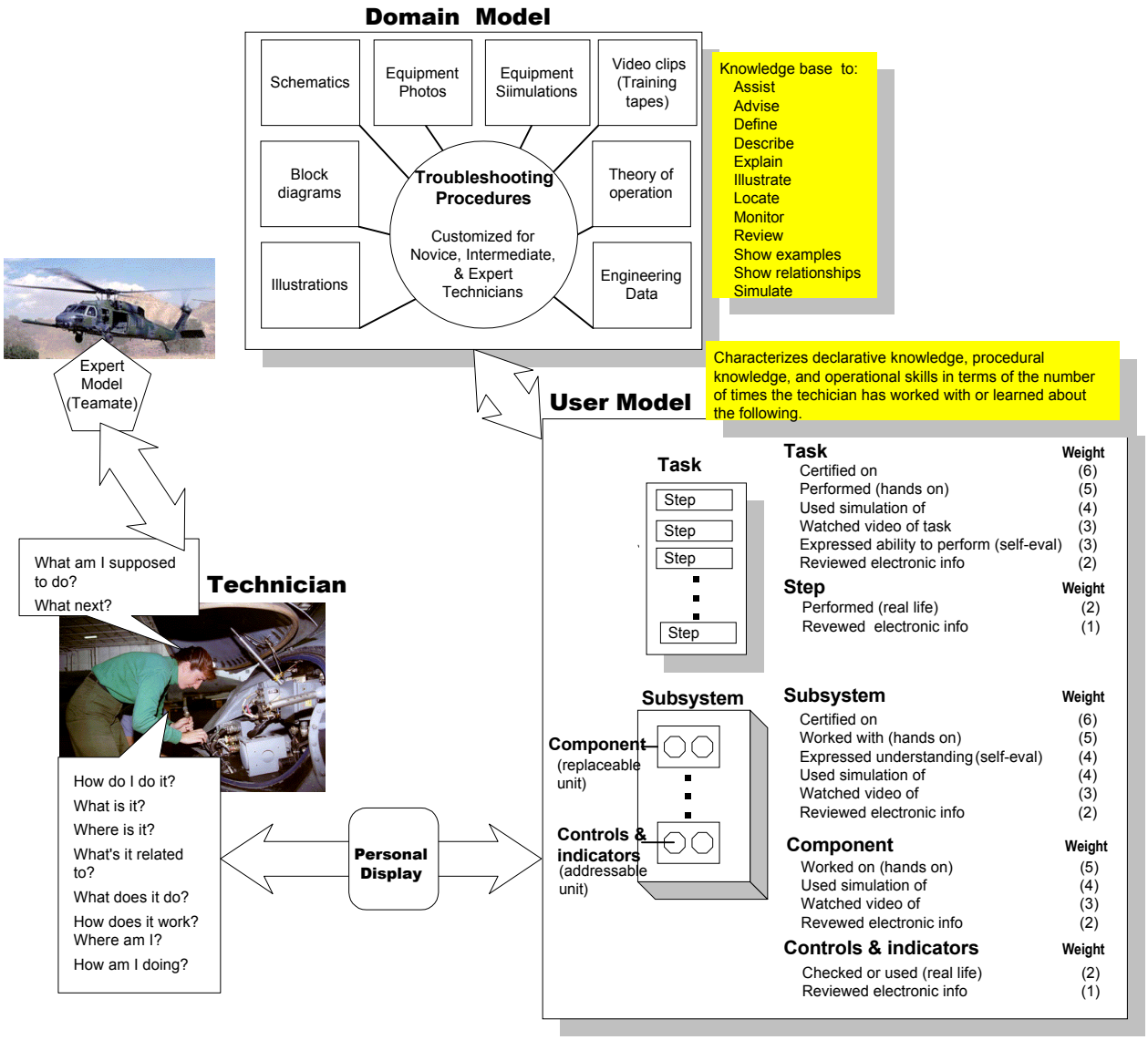

Figure 4. Adaptive hypermedia interface used information about the task and about the user to provide a personalized presentation of the sequence of steps to be performed and the supporting information for each step.

CONCLUSION

ADAPTS is an electronic performance support system that integrates an adaptive diagnostics engine with adaptive access to supporting information. Integrated performance support systems bring together an expert system-like problem solving engine and an on-line information system. ADAPTS provides comprehensive adaptive support on several stages of troubleshooting from identifying the source of troubles to determining the course of actions to guiding the user through the troubleshooting process to assembling the individualized set of supporting materials (Figure 4).

The ADAPTS system was initially developed as a proof-of-concept research project using operational technical manual data from a Navy H-60 helicopter program. During this initial phase, the focus was on developing the adaptive user interface, integrating the Condition Based Maintenance (CBM) software, and experimenting with the response of the system to variations in the user model. The first version of the system was implemented in 1999. More recently, a follow-up SBIR contract was granted to review the usability of the system in both an aiding (performance-oriented maintenance assistance) and training contexts. The results of this usability study will be reported later in a separate paper.

RELATED WORKS

The combination of structured domain models and overlay student models has been popular in the field of intelligent tutoring systems for more than 20 years and was used in dozens systems starting from early projects BIP [1] and GCAI [18]. More recently, this combination became popular in the field of adaptive hypermedia [4] and is used now in nearly every adaptive hypermedia system.

Task models have been staying in the center of the area of intelligent user interfaces for more than ten years. Traditionally, they were used by two groups of IUI systems: model-based interface development environments and intelligent help systems. Model-based interface development environments [20] applied several kinds of models including task models to facilitate the design and development of user interfaces [22; 23; 24]. Intelligent help systems used task models and various plan recognition techniques [7; 12; 14; 26] to deduce higher-level goals of the user while observing their interface-level actions [9; 16; 27; 28]. In some more recent intelligent help systems, the task model was used as a basis for the development of long-term individual user models [19; 21; 28]. There were also a few attempts to use task models in adaptive hypermedia systems [11; 13; 25]. In all these systems, task models were used to filter the potentially relevant set of links. FLEXCEL [13], an adaptive hypermedia help system for EXCEL, was able to filter the links to help information using the knowledge of the current user task. HYNECOSUM [25] and SWAN [11] provided a filtered view to the very large hyperspace of relevant documents using the knowledge about a set of tasks that are typical for the particular kind of user.

Finally, ADAPTS belongs to the class of intelligent performance support systems, an emerging type of user-oriented intelligent system. Other systems in this class known to us are described in [8; 10; 17].

The reported work inherited a lot from the works cited above and attempted to integrate together and extend several adaptation techniques investigated earlier. To provide a more reliable adaptation to the user, ADAPTS applies integrated domain and task models together with the overlay user model. In addition to simple link filtering, ADAPTS applies stretchtext-based adaptive presentation and several types of adaptive navigation support. We think that the suggested combination of models as well as presented adaptation techniques could be very useful for intelligent performance support systems in various domains.

ACKNOWLEDGMENTS

The ADAPTS project was funded by the Office of Naval Research grant to the Information Technology Branch of the Naval Air Warfare Center–Aircraft Division (NAWCAD) in St. Inigoes, MD.

This work was the result of collaboration among researchers from NAWCAD, Carnegie-Mellon University, University of Connecticut, Antech Systems, Inc., http://www.antechsystems.com, and Qualtech Systems, Inc., http://www.teamqsi.com.

REFERENCES

[1] Barr, A., Beard, M., and Atkinson, R. C.: The computer as tutorial laboratory: the Stanford BIP project. International Journal on the Man-Machine Studies **8**, 5 (1976) 567-596

[2] Boyle, C. and Encarnacion, A. O.: MetaDoc: an adaptive hypertext reading system. User Modeling and User-Adapted Interaction **4**, 1 (1994) 1-19

[3] Brusilovsky, P.: Adaptive hypermedia, an attempt to analyze and generalize. In: Brusilovsky, P., Kommers, P. and Streitz, N. (eds.): Multimedia, Hypermedia, and Virtual Reality. Lecture Notes in Computer Science, Vol. 1077. Springer-Verlag, Berlin (1996) 288-304

[4] Brusilovsky, P.: Methods and techniques of adaptive hypermedia. User Modeling and User-Adapted Interaction **6**, 2-3 (1996) 87-129

[5] Cooper, D. W., Veitch, F. P., Anderson, M. M., and Clifford, M. J.: Adaptive diagnostics and personalized technical support (ADAPTS). In: Proc. of IEEE Aerospace Conference, Aspen, Colorado (1999) Paper Number 4.602

[6] Deb, S., Pattipati, K. R., and Shrestha, R.: QSI's Integrated Toolset. In: Proc. of IEEE Autotestcon, Anaheim, CA (1997) 408-421

[7] Desmarais, M. C., Giroux, L., and Larochelle, S.: The diagnosis of user strategies. In: Bullinger, H.-J. and

Shackel, B. (eds.) Human-Computer Interaction. Elsevier, Amsterdam (1987) 185-189

[8] Fischer, G. and Ye, Y.: Personalized delivered information in a software reuse environment. In: Bauer, M., Gmytrasiewicz, P. J. and Vassileva, J. (eds.) User Modeling 2001. Lecture Notes on Artificial Intelligence, Vol. 2109. Springer-Verlag, Berlin (2001) 178-187

[9] Fox, T., Grunst, G., and Quast, K.-J.: HyPlan - a context-sensitive hypermedia help system, Arbeitspapiere der GMD No. 743, GMD, Germany (1993)

[10] Francisco-Revilla, L. and Shipman III, F. M.: Adaptive medical information delivery: combining user, task, and situation models. In: Lieberman, H. (ed.) Proc. of 2000 International Conference on Intelligent User Interfaces, New Orleans, LA, ACM Press (2000) 94-97, available online at: http://lieber.www.media.mit.edu/people/lieber/IUI/Francisco/Francisco.ps

[11] Garlatti, S., Iksal, S., and Kervella, P.: Adaptive on-line information system by means of a task model and spatial views. Computer Science Report, Eindhoven University of Technology, Eindhoven (1999) 59-66

[12] Goodman, B. A. and Litman, D. J.: On the interaction between plan recognition and intelligent interfaces. User Modeling and User-Adapted Interaction **2**, 1 (1992) 83-115

[13] Grunst, G.: Adaptive hypermedia for support systems. In: Schneider-Hufschmidt, M., Kühme, T. and Malinowski, U. (eds.): Adaptive user interfaces: Principles and practice. North-Holland, Amsterdam (1993) 269-283

[14] Hecking, M.: How to use plan recognition to improve the abilities of the intelligent help system SINIX Consultant. In: Bullinger, H.-J. and Shackel, B. (eds.) Proc. of Interact´87, the IFIP TC13 Second International Conference on Human-Computer Interaction, Stuttgart, North-Holland (1987) 657-662

[15] Höök, K., Karlgren, J., Wærn, A., Dahlbäck, N., Jansson, C. G., Karlgren, K., and Lemaire, B.: A glass box approach to adaptive hypermedia. User Modeling and User-Adapted Interaction **6**, 2-3 (1996) 157-184

[16] Hoppe, H. U.: Intelligent user support based on task models. In: Schneider-Hufschmidt, M., Kühme, T. and Malinowski, U. (eds.): Adaptive user interfaces: Principles and practice. North-Holland, Amsterdam (1993) 167-181

[17] Johnson, C., Birnbaum, L., Bareiss, R., and Hinrichs, T.: Integrating organizational memory and performance support. In: Proc. of International Conference on Intelligent User Interfaces, IUI'99, Redondo Beach, CA, ACM Press (1999) 127-134, available online at http://www.acm.org/pubs/articles/proceedings/uist/291080/p127-johnson/p127-johnson.pdf

[18] Koffman, E. B. and Perry, J. M.: A model for generative CAI and concept selection. International Journal on the Man-Machine Studies **8** (1976) 397-410

[19] Linton, F., Joy, D., and Schaefer, H.-P.: Building user and expert models by long-term observation of application usage. In: Kay, J. (ed.) SpringerWienNewYork, Wien (1999) 129-138

[20] Myers, B., Hudson, S. E., and Pausch, R.: Past, present and future of user interface software tools. In: Carroll, J. M. (ed.) HCI In the New Millennium. Addison-Wesley, New York (2001) 213-233

[21] Nessen, E.: SC-UM: user modelling in SINIX consultant. Applied Artificial Intelligence **3** (1989) 33-44

[22] Puerta, A.: A model-based interface development environment. IEEE Software **14**, July/August (1997) 40-47

[23] Schreiber, S.: Specification and generation of user interfaces with the BOSS system. In: Blumenthal, B., Gornostaev, J. and Unger, C. (eds.) Human-Computer Interaction. Lecture Notes in Computer Science, Vol. 876. Springer-Verlag, Berlin (1994) 107-120

[24] Szekely, P., Luo, P., and Neches, R.: Beyond interface builders: Model-based interface tools. In: Proc. of INTERCHI'93, New York, ACM (1993) 383-390

[25] Vassileva, J.: A task-centered approach for user modeling in a hypermedia office documentation system. User Modeling and User-Adapted Interaction **6**, 2-3 (1996) 185-224

[26] Wærn, A.: Local plan recognition in direct manipulation interfaces. In: Moore, J., Edmonds, E. and Puerta, A. (eds.) Proc. of 1997 International Conference on Intelligent User Interfaces, Orlando, Florida, ACM (1997) 7-14

[27] Wasson, B. and Akselsen, S.: An overview of on-line assistance: from on-line documentation to intelligent help and training. The Knowledge Engeneering Review **7**, 4 (1992)

[28] Winkels, R. G. F.: User modelling in help systems. In: Norrie, D. H. and Six, H. W. (eds.) Computer Assisted Learning. Lecture Notes in Computer Science, Vol. 438. Springer-Verlag, Berlin (1990) 184-193