HELPDESK: Using AI to Improve Customer Service

Debra Logan and Jeffrey Kenyon, Carnegie Group, Inc.

HELPDESK is a software solution that enables a customer service organization to expand its first-tier problem-resolution capabilities without adding additional personnel or requiring additional training. It is an integrated approach to customer support, encompassing a diagnostic expert system, a hypertext reference facility, and a trouble-ticketing database system. The effectiveness of the system in meeting its goals has been verified through user surveys, showing its acceptance in everyday use. Its success is attributed to the tight integration of its modules, a phased deployment strategy, and local maintenance.

Introduction

In many organizations, the task of supporting a product or service is performed by a hot line or help-desk support group. All these groups share certain characteristics. They receive calls from internal or external customers and must track and resolve these calls. The calls are resolved by either the support staff or referral to another organization. Most help desks use some type of trouble ticket, either on paper or computer, to record calls and maintain a central status board listing the calls that have not been resolved and any information of general interest to the group. Many help desks support multiple domains, sev-

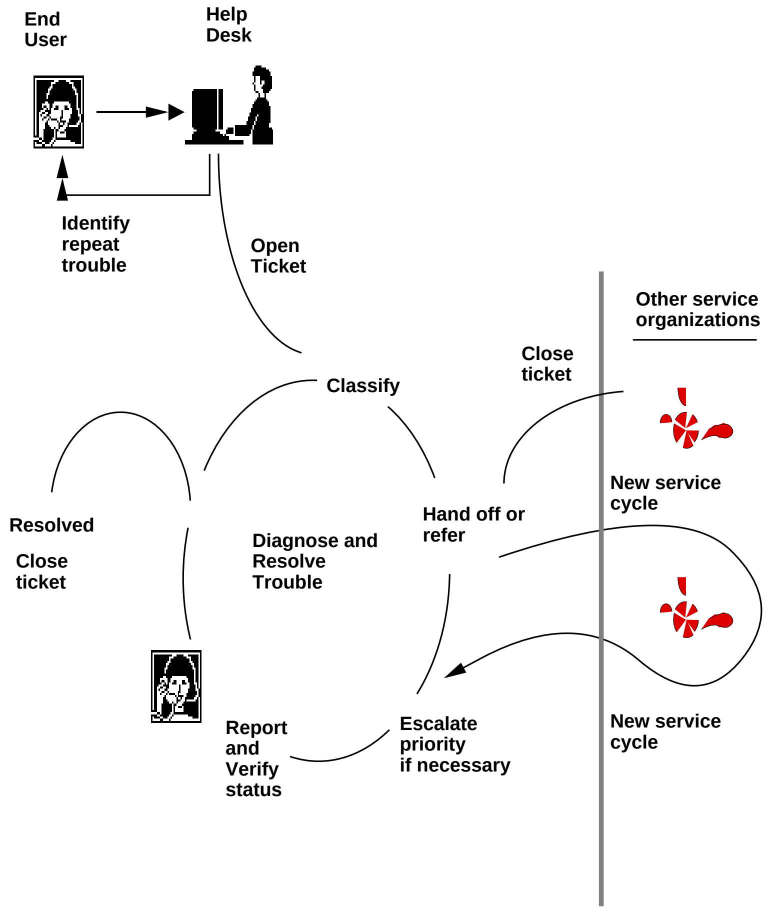


Figure 1. HelpDesk Service Cycle.

eral software programs, a variety of hardware, or a family of products. Just as most help desks share a set of characteristics, they also share a set of problems:

First, because many help desks support multiple domains, individuals develop expertise in particular domains over time, and the help desk comes to depend on this expertise, referring all difficult problems in the domain to a particular expert. If the expert is unavailable, then so is the expertise. The customer must then wait for an answer or rely on a less experienced individual. If the expert leaves the group or the company, much of the expertise is lost.

Second, help desks often have high turnover rates because of the stressful nature of the job. This turnover has the effect of draining the group of expertise. The experienced staff members remaining must take on the additional task of training replacement staff members.

Third, the number of users supported by a help desk often increases, but the number of help-desk staff members remains constant because of economic constraints.

Fourth, help desks are being asked to support increasing numbers of products and services. Learning everything there is to know in today's help-desk environment is a virtual impossibility for the help-desk staffers.

To alleviate these problems, U S West, a large telecommunications company,[1] began exploring ways of increasing their operational effectiveness while maintaining service quality, maintaining or decreasing staffing levels, and adding to the user base. AI technology suggested itself as a natural way of solving some of the problems that the help desks at this company were experiencing.

In studying the problem of help desks, it was discovered that they all carried out three basic functions: administrative, diagnostic, and reference. Two of the functions, administrative and reference, were determined to be functions of conventional technology; the third, diagnosis, required the application of AI technology.

A software solution tightly integrating these functions was then designed and deployed. A phased approach was taken to development, allowing staffers to begin using the software within a week after the start of development and realize benefits. At the conclusion of the development phase, the responsibility for maintenance was transferred entirely to local control.

In this chapter, we present a complete description of the HELPDESK software solution, its development and deployment within U S West Communications, and its emphasis on user empowerment.

Problem Description

The function of a help desk is to resolve the technological problems encountered by product users. These problems are usually reported verbally by telephone, although electronic mail, paper mail, and personal visits can also be used. The help-desk personnel must then track the information given by the customer until the problem is resolved. The service cycle is illustrated in figure 1.

In the service cycle, three basic job functions can be identified: information tracking, reference, and diagnosis. Each function is problematic.

To track information, the help-desk staffers are required to fill out a trouble report. These reports, or *trouble tickets*, can either be on paper or online. Tickets are used to track a problem until it is resolved. They can also serve as input to daily status postings and monthly status reports.

This trouble-ticketing function can lead to several kinds of difficulties. If the tickets are of a paper variety, they can be easy to fill out, but paper makes it difficult to disseminate critical information among members of the group. In addition, the data on paper tickets do not make ready input to reporting mechanisms or status posting. In the paper environment, writing the information on the status board is a manual process, and it is only useful if (1) everyone checks the status board regularly and (2) everyone can see the status board.

Online tickets present problems of their own. Internal Carnegie Group assessments found several types of online tickets in use, maintained with older databases on large mainframe computers in a central location. The ticket format tended to be inflexible, and using the data-storage and data-retrieval facilities was time consuming (Logan, Kenyon, and White 1991).

The reference function also proved to be complex. The information that help-desk staffers need to resolve problems is often paper based. In some environments, there can be several thousand pages of paper documentation. Keeping the material current is a problem; updates are frequent and difficult to track. Telephone numbers, contact names, product release data, and procedures are other examples of information that the help-desk staffer must track and have available for users. Finding a way to efficiently organize and access all the diverse types of information that a typical help-desk staffer must have available proved to be a challenging problem.

As a third job function, a help-desk staff member must diagnose complex problems. Typically, help desks support many different products; end user computing is one example. One help desk Carnegie Group worked with supported office automation software on a network of minicomputers in different geographic locations. When a user calls the help desk with a log-in problem, the problem can potentially be in the host computer, the network, the application software, or the user's procedure. Diagnosing such a problem requires a high degree of expertise. Generally, expertise in particular domains tends to develop around certain individuals; learning the details of troubleshooting over 20 different applications is beyond the scope of most people. When users call, problems are referred to the expert on a particular application. The difficulty arises when an expert is temporarily or permanently unavailable (Logan, Kenyon, and White 1991).

These three functional areas represent the staff member's perspective on the help-desk problem. From the manager's viewpoint, there are different problems associated with running a help desk. The manager's primary concern is to maintain a consistently high level of support to customers calling the organization, using people with varying levels of experience or ability, possibly in multiple locations.

All help desks face economic constraints. If the number of customers calling the help desk is increasing, the manager might be faced with the need to add new support people, who will need to be trained, or help the existing staff members to manage an increasing work load. The training of a new staff member can take between 1 and 12 months, depending on the complexity of what is being supported (Logan, Kenyon, and White 1991).

Managers must also deal with the problem of replacing staff members. Often, the most experienced members of the help desk have grown beyond their support roles and are ready to move on to other assignments. In addition, burnout is common because the stress of listening to customer problems all day takes its toll. The loss of accumulated expertise can be devastating; with the experienced support people gone, training replacements becomes even more difficult.

Application Description

After studying the operations of help desks, an architecture for computerizing help-desk support operations was developed using a mix of conventional and AI technology.

The HELPDESK software solution consists of three integrated modules. The first is a trouble-ticketing system for reducing the administrative overhead associated with paper trouble tickets. The trouble-tracking system is a straightforward database application. The second HELPDESK module is the diagnostic adviser, for increasing the range and number of problems solvable by the support staff without assistance. This component of the software is an expert system based on the Carnegie Group product TESTBENCH. The final module of the software suite is a hypertext reference system designed to simultaneously consolidate and distribute a standard body of information needed by the support staff. These modules are collectively referred to as the HELPDESK software system. Figure 2 shows these components in graphic format.

Application of AI Technology: THE Diagnostic Adviser

The diagnostic adviser addresses the issue of scarce, distributed, or vanishing expertise within a help-desk organization. It also addresses the problem of ever-increasing levels of complexity of products and ser-

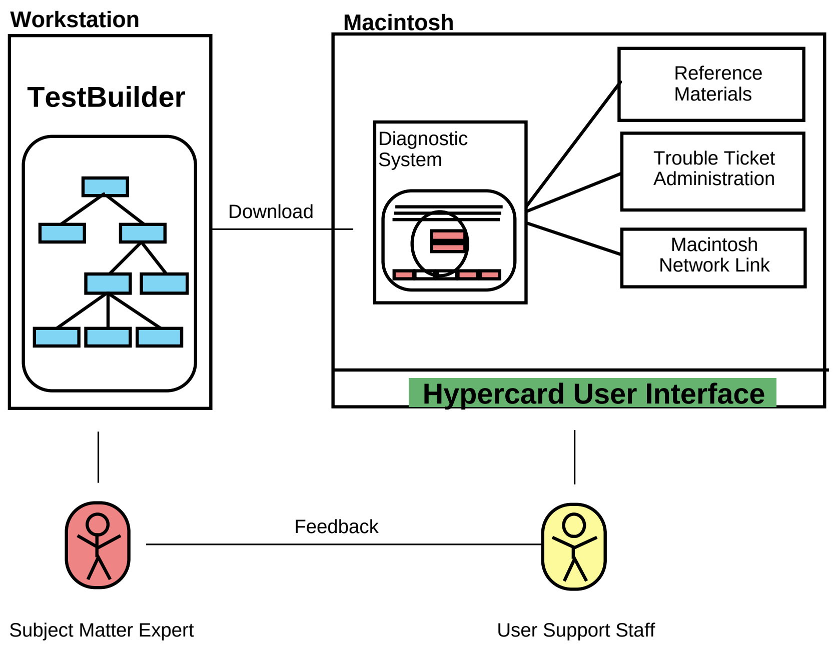


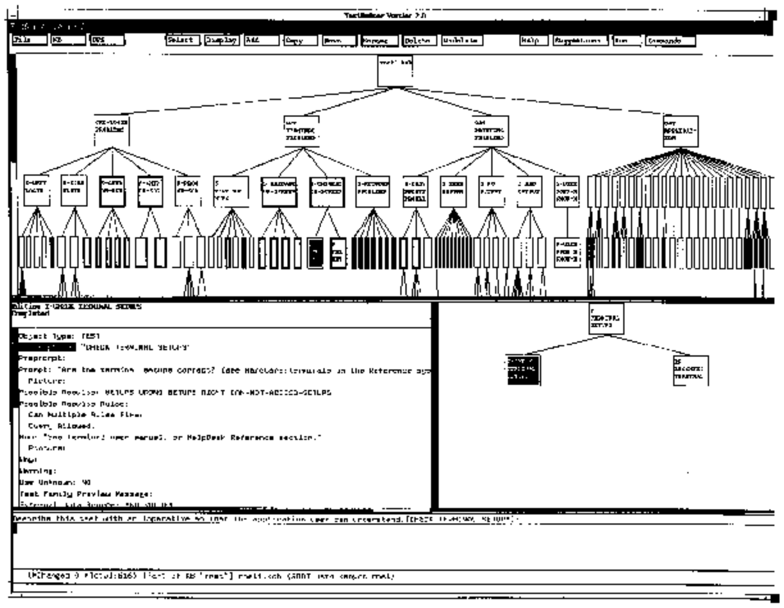
Figure 2. HELP DESK *Components.*

vices to be supported. By capturing knowledge in an expert system, it remains available even when the individuals who possess it are not.

The adviser, built using TESTBENCH, offers a method of capturing expertise as a permanent asset of the group and distributing this expertise to all the support staff. It was chosen because it matched the task of software and hardware diagnostic problem solving faced by help desks.

TESTBENCH (Carnegie Group 1991) is made up of three modules. TESTBUILDER is the workstation-based development environment; it comprises a graphic knowledge editor (figure 3) and an inference engine called the DIAGNOSTIC PROBLEM SOLVER (figure 4). When ready for deployment, the knowledge base is moved to the delivery environment and compiled into binary form using the TESTBRIDGE module. The third module is TESTVIEW, the delivery diagnostic environment available on a number of platforms, from which end users can run the compiled knowledge base.

The value of TESTBENCH is in its knowledge representation. Instead of programming languages or rules, knowledge in TESTBENCH is captured in objects specific to the diagnostic domain. At the top level are *category objects*, which are used to logically group the *symptom objects* the

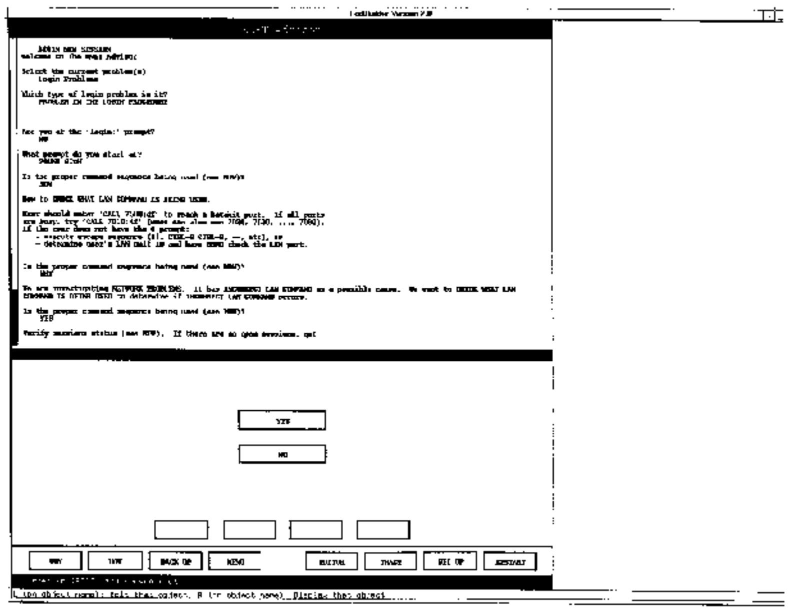


```
Editing T-CHECK TERMINAL SETUPS
[illegible]

Object Type: TEST
[illegible] "CHECK TERMINAL SETUPS"
Preprompt:
Prompt: "Are the terminal setups correct? (See HelpDesk:[illegible] in the Reference sys
  Picture:
Possible Results: SETUPS WRONG SETUPS RIGHT CAN-NOT-ACCESS-SETUPS
Possible Results Rules:
  Can Multiple Rules Fire:
  Overr. Allowed.
How: "See terminal user manual, or HelpDesk Reference section."
  Picture:
Why:
Warning:
Use Unknown: NO
Test Family Preview Message:
External Data Source: [illegible]
Describe this test with an imperative so that the application user can understand.[illegible]

[illegible]
```

Figure 3. TEST BUILDER *Knowledge Editor.*

system is capable of diagnosing (for example, in the category Printer Problems might be the symptoms No Printer Output and Printer Error Message). Attached to each symptom are the *failure objects* that are possible causes; when a symptom is selected by an end user for diagnosis, these failures are each considered in turn and either confirmed or disconfirmed based on data gathered from the end user. Data are collected through the use of *question objects* and *test objects*, presented to the user as required. Questions and tests can be combined into logical families (so that all tests on a subject are asked at one time) and in And or Or relationships. When a failure is confirmed, it is either repaired (if it is a component-level failure) using the procedure contained in the *repair object* attached to the failure, or the system continues to isolate the failure down to the component level by investigating those failures that are possible causes for the confirmed failure. Although the diagnostic behavior is established by the developer, it can be altered at run time, using rules to change the order of failure investigation or recommend a different repair.

Development proceeds by acquiring the necessary knowledge from the domain expert, programming the knowledge into TESTBUILDER, demonstrating the diagnostic behavior to the expert, then refining the



```
BEGIN NEW SESSION
Welcome to the [illegible] Advisor

Select the current problem(s)
    Login Problems

Which type of login problem is it?
    PROBLEM IN THE LOGIN [illegible]

Are you at the 'login:' prompt?
    NO

What prompt do you start at?
    [illegible]

Is the proper command sequence being used (see [illegible])?
    [illegible]

How to CHECK WHAT LAN COMMAND IS BEING USED.

[illegible] should enter 'CALL [illegible]' to reach a [illegible] port. If all ports
are busy, try 'CALL 7010:[illegible]' [illegible] 7020, 7030, ..., 7060).
If the user does not have the [illegible] prompt:
  - execute escape sequence ([illegible], CTRL-[illegible], --, etc], or
  - determine user's LAN unit ID and have [illegible] check the LAN port.

Is the proper command sequence being used (see [illegible])?
    [illegible]

We are troubleshooting NETWORK PROBLEMS. It has [illegible] LAN COMMAND as a possible cause. We want to CHECK WHAT LAN
COMMAND IS BEING USED to determine if [illegible] LAN COMMAND occurs.

Is the proper command sequence being used (see [illegible])?
    YES

Verify [illegible] status (see [illegible]). If there are no [illegible] [illegible], [illegible]
```

Figure 4. TEST BUILDER DIAGNOSTIC PROBLEM SOLVER.

behavior as needed. Figure 4 shows the TESTBUILDER DIAGNOSTIC PROBLEM SOLVER, which is used by the developer to examine and demonstrate the diagnostic behavior. The DIAGNOSTIC PROBLEM SOLVER uses the knowledge base to engage the user in a question-and-answer dialog. This approach is natural for the help-desk environment, in which the user is often at the other end of the telephone, and the help-desk staffer is trying to determine what the problem is by querying the user.

There are several alternatives to TESTBENCH's fault-isolation strategy, but none met the needs of the HELPDESK application. The most obvious alternative is the classic *rule-based system*: Using rules and data gathered from the user, the system attempts to reason to a conclusion. The difficulty of maintaining a large rule base, especially for a nonprogrammer, was a major factor in deciding against this approach. The use of a deep causal model was also ruled out because of the difficulty in creating the number of models required (some help desks offer service over a dozen or more discrete systems). Decision trees, another option, lacked the needed flexibility and were seen as too difficult to maintain, especially in a large knowledge base. The last alternative, case-based reasoning, was assessed as too slow for the requirements of HELPDESK (response within 2 to 3 seconds) and the diagnostic behavior too difficult to control. The advantage of TESTBENCH is that it allows a diagnostic knowledge base to be built by a nonprogrammer in a format that facilitates maintainability.

HELPDESK'S diagnostic adviser is a TESTBENCH application, developed and maintained locally by a knowledge engineer working with one or more experts within the help-desk group itself. Once the diagnostic behavior is verified and validated on the development platform, a runtime version that cannot be modified is delivered on the MACINTOSH platform. The ability to maintain the diagnostic adviser locally, without the help of a programmer or a knowledge engineer, is vitally important; help desks, as a group, are dynamic environments, with knowledge assets requiring regular updates. Rather than turning the responsibility for maintenance over to a programmer, the task can instead be given to a senior member of the customer support team, who modifies the system in response to his/her colleagues. By keeping maintenance under local control, the probability that the system will evolve over time to meet the needs of the users increases dramatically.

A traditional approach was inadequate for help desks because of the many different interacting components of the environment. Most help desks must have multiple experts to support the multiple areas of their responsibility. AI's approach of separating knowledge base from inference engine was a natural choice; multiple knowledge bases could be constructed, but the same diagnostic inference engine could be used for each one. The ability to incorporate the knowledge of multiple domain experts was a critical element in the success of the software. In addition, because domain knowledge is different for each help desk, the paradigm of a general problem-solving method (diagnosing) used over a variety of different domains matched the nature of the environment.

TESTBENCH was chosen for its proven record of success in building diagnostic expert systems, its knowledge representation, and its graphic user interface.

The diagnostic adviser alone was not sufficient to solve the problem. The other technologies (database and hypertext) used to solve the administrative and reference lookup function are described in the following sections.

Application of Other Technology: Trouble-Ticketing System and Reference

The Trouble-Ticketing System (TTS) replaces paper trouble tickets with computerized versions of the same tickets. These electronic tickets are easily passed between members of the help-desk staff; are used to automatically maintain a central, electronic status board; and can be used to provide a variety of administrative reports. The software is MACINTOSH based, with a hypercard interface to an OMNIS 5 database.

TTS is based on a client-server architecture, with the server containing a central database of tickets. Any number of client workstation users can log onto the server over local area networks (LANs) or wide-area networks (WANs). These clients are able to create and store unopened tickets on their own machines, but once a ticket is opened, it is saved on the central server and appears on the central status board.

The *central status board* is a computer-based version of the standard chalk or white board maintained by most help-desk operations. It is automatically updated and is instantly available to all users on the system, anywhere on the LAN or WAN. Users are able to select tickets directly from the status board and continue call resolution from the point where the last support person left off without having to re-request basic information from the caller.

The REFERENCE DESK module of the HELPDESK software is a MACINTOSH-based hypercard stack developed from the ground up according to the needs of each help-desk organization. In addition to containing detailed descriptions of various procedures routinely performed by the help desk (for example, changing a password) or lists of often-used telephone numbers, reference information can include such items as maps, tables, or textual or graphic information on hardware or software. The designer of the reference material is limited only by the development software.

A hypertext approach (Conklin 1987) was seen as superior because of the eclectic nature of the information to be captured. Each organization would require the ability to include any type of information, textual or graphic, and establish unique links between items to navigate through the information effectively.

A fourth module, the TTS ADMINISTRATOR, is used only by developers and maintainers of HELPDESK systems. The TTS ADMINISTRATOR is used primarily to create and modify the electronic versions of the tickets and the layout of the central status board in an icon-based, WYSIWYG (what you see is what you get) interface.

Technology Integration

Only by integrating these three software components was it possible to deliver a total solution that met the needs of diverse groups of users doing similar jobs. The help-desk problem was solved by integrating three shell technologies (diagnostic expert system, database, and hypertext). Each shell was selected to match the demands of a specific job function and then populated with domain-specific knowledge. This integration of conventional and AI technologies is an innovative way of solving a real and pressing business problem.

To be successful, it was felt that a tight integration of the modules was key. It would not be sufficient to have separate applications handling each of the needed functions; instead, the movement between expert system, database, and hypertext modules had to be as seamless and natural as possible. The HELPDESK software is not viewed as an AI application; it is a solution where AI has a key role but is working in concert with other technology.

Application Development and Deployment

The average software life cycle proceeds along a linear course through the preliminary exploration and requirement analysis to development, validation, implementation, and maintenance (Tuthill 1990). Although true in the case of the individual modules of HELPDESK, the design of the software invites a parallel and sometimes overlapping development process. It is precisely this process that allows the system developers to introduce change as an evolutionary (rather than a revolutionary) process and to build user acceptance at each phase.

Typically, development of a custom HELPDESK begins with the application assessment. After determining that the HELPDESK software is appropriate for a support group, the appropriate hardware is purchased and deployed. The first software module to be developed is the trouble-ticketing software because its design is copied (at least initially) from an existing paper ticket. The ticket can be designed, implemented, and the users trained within the first week of development. The support staff becomes acclimated to both the MACINTOSH and the trouble-ticketing software while development on the other modules continues.

The REFERENCE DESK is easily implemented in stages as needs become apparent, and time becomes available. It might begin as nothing more than an electronic list of telephone numbers, then expand to cover a number of basic procedures. As the help-desk staffers gain familiarity with the system, they will often be vocal about what they would like to see included in the reference materials. The REFERENCE DESK, more so than any other module, continues to expand and grow over the software life cycle.

The diagnostic adviser is the most difficult portion to develop because it often involves learning both new hardware and new software. It also involves knowledge acquisition, verification, and validation to ensure that the system meets the demands of the real world.

The initial development of the TTS system took approximately one person-year and occurred over a period of six months. The development staff consisted of two developers and a project manager. Averaged over the five subsequent deployments, the cost for each deployment was roughly the annual cost of one help-desk staff member. Because the system has allowed at least one group to expand the hours of coverage and the number of systems they support without adding to the head count, the system has proven its ability to pay off for U S West (Logan, Kenyon, and White 1991).

Deployment

Deployment of the HELPDESK software required the purchase of MACINTOSH II computers for the help-desk staff members, an additional MACINTOSH as the central file server, and cable sufficient to connect the clients and server on the APPLE TALK network.

Training on the HELPDESK software was minimal, consisting of an initial 4 hour session with all the help-desk staff members and short one-on-one sessions (usually 15 minutes or less) when new functions were added to the system.

At the initial site, the system was deployed with approximately 15 MACINTOSH workstations. For a subsequent deployment, the system was installed on approximately 10 workstations. In addition, the diagnostic adviser was made available to all end users through the UNIX server used for a variety of office automation tasks. At the conclusion of a diagnostic session, if the outcome of the session is unsatisfactory, the end user has the option of mailing a log of the session to the help desk's electronic mail address along with a text description of why the resolution was unsatisfactory.

Deployment of these systems has become a short process, taking from one to four weeks. The variation is accounted for by whether the site has hardware already in place or whether the hardware must be installed as a part of the process of deploying the software.

Maintenance

The concept of local maintenance is a key element in the design philosophy of the HELPDESK. Using the TTS Administrator software, TESTBUILDER (the development environment for the adviser system), and hypercard, each module of the HELPDESK system is maintainable on a local level by domain experts with little or no programming expertise. One or more staff members of the help desk perform all HELPDESK maintenance, and their training is an important part in the delivery of each HELPDESK application.

Local maintenance also allows a faster resolution of errors or gaps in the domain knowledge base. Help-desk environments are constantly changing, and releases are made on an as-needed basis; if needed, the domain knowledge can be updated as often as once a day.[2] Local control of the maintenance process significantly reduces the development and testing cycle for changes and extensions. A feeling of ownership is another advantage stemming from the emphasis on local maintenance. A software system that is maintained by a local administrator quickly becomes tailored to the group's environment and becomes their software rather than an externally imposed software package that they must use and conform to.

The drawbacks of local maintenance are few and can be avoided by the group's management. The problems observed were primarily because of the assignment of maintenance responsibilities to a staff member with a variety of other, more pressing responsibilities. The diagnostic system was not updated in response to the staff's suggestions, and as a result, it became outdated quickly. Staff members stopped consulting the system once they had learned its contents, and eventually it became little more than a training tool for new staff members. In a changing environment, the HELPDESK domain knowledge must be updated on a regular basis; otherwise, it becomes an outdated snapshot from the past.

Before releasing new versions, the domain knowledge must be verified and validated. *Verification*, or confirming that the knowledge in the system is in agreement with the expert's knowledge, is a simple matter because it is the expert who placed the knowledge into the system. *Validation*, or ensuring that the knowledge is in agreement with the real world, is a more complex affair. For the TTS and REFERENCE DESK modules, validation is carried out by recruiting a user (if possible, the user who requested the modifications) to evaluate the changes. For the diagnostic adviser, the preferred way of performing validation is to induce the failure in the real world and then ask a third party to use the adviser to solve the problem. In cases where induction of a failure is too dangerous or costly, a third party reviews the transcript of the user-adviser dialog. The preexisting knowledge should also be reexamined to ensure that the new knowledge has not corrupted the existing knowledge in some way.

The validation of the preexisting domain knowledge was facilitated by the use of the TESTBENCH diagnostic shell. With TESTBENCH, a library of user-adviser dialogs can be developed and rerun automatically, providing an effective means of automated regression testing.

Application Use and Payoff

The HELPDESK software developed over the course of these projects is in use by five different user groups; approximately 50 people use the software on a daily basis to perform their job function. A help desk responsible for office automation support has deployed the diagnostic adviser module on a regional network, enabling hundreds of users to troubleshoot some of their own problems. HELPDESK systems at U S West have been deployed and in use since September 1989 and have produced multiple benefits.

In the first deployment, at a help desk supporting a number of software applications, the PREDICTOR application with approximately 3000 end users was targeted for HELPDESK development; all calls regarding this application were automatically referred to the domain expert. The HELPDESK software was delivered and deployed in 20 person-weeks. The diagnostic adviser enabled the help-desk personnel to respond effectively to 80 percent of the incoming calls in the first 4 months of its deployment in which the help desk received, on average, 30 calls each month. For this 80 percent, the time required to resolve the problem was reduced by approximately 30 minutes; this savings was the result of eliminating the average time between the referral to the domain expert and the expert returning the call and resolving the problem (Logan, Carey, and Hayes 1990).

A second benefit that resulted from the introduction of the HELPDESK software had to do with the increase in support capabilities that the help-desk personnel were able to provide. In the first help-desk organization where the software was deployed, there were 10 help-desk analysts supporting 7 computer systems. The help desk operated from 6:00 A.M. to 8:00 P.M. Over the course of a year after the HELPDESK software was installed, the number of systems supported rose to 12, and the hours of operation increased from 14 to 20. According to U S West management, the increase in support capabilities was accomplished without an increase in head count because of the deployment of the HELPDESK software. With the preexisting manual procedures, this expansion of service would have required a minimum of one to two additional support personnel.

The HELPDESK system also improved the quality of service provided by the help-desk staff. Daily users of the HELPDESK software were asked the question, Do you feel that the HELPDESK software has improved the quality of customer service that your group provides? Of the 21 respondents, 17 said yes. They also reported that the HELPDESK software had decreased call turnaround time and attributed this decrease to the software itself. Finally, users of the expert system portion of the system re-

port that it has decreased the number of calls that they must refer to domain experts (Logan, Kenyon, and White 1991).

Conclusions

The success of the HELPDESK projects at U S West is the result of three factors. First, the application modules provide a total software solution for help-desk groups. By mixing conventional and AI technologies, the users were given what they needed to become more efficient at their jobs. In the original vision of automated help-desk support, the expert system was the only component that was needed. In gathering user requirements as part of the original application assessment, however, it soon became apparent that there were other needs, such as an online trouble-tracking system, that were far more pressing than the need for expert system technology. By listening to the users and providing them with all the component pieces that they needed, user support was gained as well as user acceptance of the technology.

Managing the initial introduction of new software technology for help-desk groups and working toward final deployment and acceptance of the system is a difficult task in the best of times. The evolutionary approach toward software development is a positive influence on

this process. Introducing components individually enables users to familiarize themselves with the software more gradually and allows the organization to realize benefits sooner. The sequence of deployment of the components can be based on ease of use, required development time, and potential for high visibility and most dramatic effect on productivity. For example, the HELPDESK REFERENCE DESK is the easiest to understand and use of the three components. Although the TTS system involves some pain—that is, it is not as fast as paper—the payback (in terms of increased efficiency) is large and readily appreciated by a group that might have been laboring under a paper system or a conventional system with cumbersome restrictions.

Finally, the notion of local maintenance is critical. The most common complaint that users had about older trouble-ticketing systems and, indeed, telephone company software in general was its inflexibility and often its inability to meet their needs. Older, monolithic systems still widely used in the telecommunications industry are not easily changed. Users are forced to adapt to the software, not the other way around. Teaching the expert to maintain the diagnostic system using TESTBENCH allows the expert system to be constantly changed and updated as the parameters of the environment change. The same philosophy is extended to the ticketing and reference components of the HELPDESK, allowing users to add new ticket families and new reference materials as the need arises. In the dynamic environment of the help desk, this approach was the only way to ensure that the system would continue to be useful after the developers were out of the picture. Further, when a system can be maintained locally and when changes and extensions proposed by the group are quickly realized, the pride of ownership enforces the acceptance climate.

Notes

1. All the data in this chapter were obtained in working with different help-desk groups at U S West.

2. On average, REFERENCEDESK is updated once a month, but both the trouble-ticketing system and diagnostic adviser are updated every six months.

References

Carnegie Group. 1991. TESTBUILDER User's Guide, Software Version 2.0. Pittsburgh, Penn.: Carnegie Group, Inc.

Conklin, J. 1987. A Survey of Hypertext, revision 2, Technical Report STP-356-86, Microelectronics and Computer Technology Corporation, Austin, Texas.

Logan, D.; Kenyon, J.; and White, J. 1991. Help-Desk Support Systems to Improve Service Quality. Paper presented at 1991 NEC ComForum: Customer Service: Strategy for the '90s, Orlando, Florida, 9—12 December.

Logan, D.; Cary, J.; and Hayes, S. 1990. The PREDICTOR HELPDESK Assistant: Software Problem Diagnosis and Resolution. In Expert Systems Conference and Exposition Proceedings, 35–46. Detroit, Mich: Engineering Society of Detroit.

Tuthill, G. S. 1990. *Knowledge Engineering: Concepts and Practices for Knowledge-Based Systems*. Blue Ridge Summit, Pa.: Tab Books.