SHARE

SHARE SESSION REPORT

61	A260	A NEW APPROACH TO DIAGNOSIS INFORMATION	200
SHARE NO.	SESSION NO.	SESSION TITLE	ATTENDANCE
DOCUMENTATION		STEVE WINCKELMAN	PSU
PROJECT		SESSION CHAIRMAN	INST. CODE

THE PENNSYLVANIA STATE UNIVERSITY ZIU COMPUTER BUILDING UNIVERSITY PARK, PA. 16802
SESSION CHAIRMAN'S COMPANY, ADDRESS, AND PHONE NUMBER

A New Approach to Diagnosis Information

Jeannette Mutimer

IBM Corporation
555 Bailey Avenue
San Jose, California 95150

DOCUMENTATION PROJECT

Session Number A260

Abstract

In keeping with task-oriented library design, IBM is now publishing diagnosis information to provide a basis for effective communication between the user and the IBM Support Center. Problems are diagnosed based on their symptoms. The objective of diagnosis information is to relate those external symptoms to a functional area of the program.

The speaker will describe the organization and content of two books — the diagnosis guide and diagnosis reference. These books complement each other and are intended to increase the productivity of the people involved in the resolution of program failures.

Introduction

The first set of diagnosis books were published in 1977 for IMS/VS. Now, at least twenty IBM program products have diagnosis books in their libraries. The titles and formats have evolved, but the concept is unchanged.

I have personally worked on this concept for six years — I have written several of these books, and now spend full time consulting with other writers who are developing diagnosis information. So why is the title of this session, "A New Approach to Diagnosis Information"? Diagnosis information is not new to many of you. Diagnosis information remains a "new approach" after six years because it is so very different from the information it replaces.

Diagnosis information is a "planned" method for locating the cause of a program problem. It does not require full knowledge of the program — that knowledge is built into the methodology. Diagnosis becomes just another task, much like installing or using the program.

I can still remember when I was in class learning to program. I had barely learned the assembler language, and the instructor moved on to a high-level language. It was a long time before I was comfortable that coding one statement would handle so many details. You may feel a similar anxiety about the difference in the amount of information between the program logic manual and the diagnosis reference.

I have three objectives today. First, to ensure that "diagnosis" is not "new" to you; to make you comfortable with the contents and organization of the diagnosis books; and to make sure you understand how to use diagnosis books to increase your own productivity. Second, I want to convince you to use diagnosis books to resolve program problems. And third, I want to invite you to let me know how well they work for you.

To meet these objectives, I will digress a bit to explain the concept of task-oriented libraries. I will also review with you our analysis of the diagnosis task. For it was this analysis which set the stage for packaging the information. I then will talk about the organization and contents of the diagnosis guide and diagnosis reference, and review a scenario of how to use these books to resolve program problems. I will conclude with a discussion of the merits of diagnosis information and the contribution these books can make to your productivity in performing the diagnosis task.

Task-Oriented Libraries

IBM's task-oriented libraries are geared to the activities people perform using our products. An overall goal of task-oriented libraries is to be "task-sufficient."

Writing task-sufficient information presents some interesting challenges. It means that all information must be pertinent, and information must be collected in one place. Of course, the information must be complete and accurate — but there is a degree of completeness and a degree of accuracy that **is not** pertinent to the task.

How do we know what is pertinent? First, we must understand the task. There is seldom only one way to do anything, so we must make some decisions about how the task will be performed. For example, if each of us were to give directions about the route to get to this hotel from the airport, we would not all give the same instructions, although we all accomplished the task. We could, however, combine our directions to describe a route that would work for everyone. All of us would not agree that it was the best route. What is important is that the directions are straightforward with as little room for error as possible. The directions must accurately describe the complete route, but the directions need not describe every intersection in that route. Completeness and accuracy to this degree would complicate rather than simplify the task.

The basis for designing a task-oriented library is a task analysis. In this analysis, each major task is examined to identify the significant activities in the task, and a plan is developed that describes the steps required to accomplish those activities. The task analysis becomes the criterion to evaluate appropriateness of information during the writing of the manual. If there is any doubt about the completeness, or the need for any portion of the content, that question can be answered by matching the information against the requirements of the task. Likewise, if a piece of information appears to be valuable, but does not map to the task analysis, then the analysis itself is reexamined.

The Diagnosis Task

Diagnosis is only one of the "tasks" supported by a task-oriented library. But diagnosis has three characteristics not common to other tasks.

1. Diagnosis involves the same basic activities for all programs.

2. A single problem could easily cause you to use books from several product libraries.

3. This is the only task that requires interaction between IBM and its customers.

Because all the activities are essentially the same for all program products, the information can be organized in the same way, even though the contents are product-specific. I am sure you will agree that consistency in organization is particularly helpful when books from two or more product libraries are used simultaneously.

Diagnosis Task Analysis

The scope of the diagnosis task is, in reality, doing whatever is necessary to keep a program productive. To bring this into focus it is necessary to establish boundaries.

Definition of the Task

The diagnosis task begins, by definition, whenever an IBM program is suspected of not functioning correctly. This definition assumes that there is reasonable evidence to suspect a software failure. The information in the diagnosis books does not address hardware problems. The definition also assumes that the program has been used correctly — although it may turn out that the problem was caused by a user error.

Diagnosis books deal with software problems, and software diagnosis typically begins with the symptoms, and tracks back to the cause of the problem, using the symptoms as clues. What may appear to be a software problem, may, in fact, be caused by an obscure user error, or a hardware problem. But that fact must usually be revealed by the same diagnostic processes that locate code problems.

Task Activities

Within the scope of this definition we separated the diagnosis task into five activities, as shown in the figure below. As I mentioned earlier, diagnosis is unique in that IBM shares the responsibilities with its customers. Some of the responsibilities overlap, and others belong either to IBM or to you, IBM's customers. Let's look at how the activities in this task relate to the people who will perform them.

Keep in mind while we go through this analysis, that we are setting the limits for both the information that will be included, as well as what will be excluded, in a task-sufficient library.

1. Identify the Source of the Problem

   The diagnostician is responsible for identifying the component in control at the time of the failure.

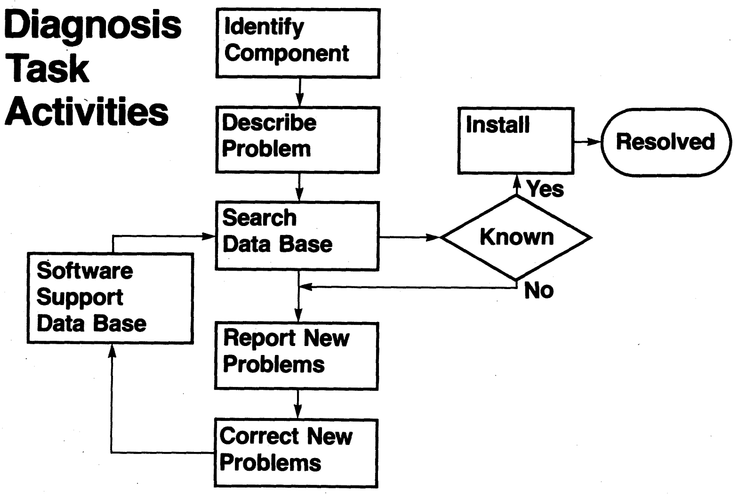
   The diagnostician is principally you, IBM's customer, although an IBM representative may also assist in that process. I use the term "diagnostician" to identify everyone who participates in the preliminary investigation of the problem.

2. Describe the Problem

   The objective of this activity is to investigate a problem sufficiently to be able to describe it with a set of keywords.

   A large percentage of program problems experienced in the field are already known, are described in the software support data base, and are corrected. The challenge for the diagnostician is describing problems consistently, so that known problems can be distinguished from new problems.

Diagnosis Task Activities



3. Search for Similar Known Problem

   The quickest way to resolve a problem is to find a ready-made solution. So the next activity is to search the software support data base for a similar problem description.

   The IBM Support Center Level 1 representative will search the software support data base, using the keyword string developed by the diagnostician. Level 1 contributes expertise on using the search argument effectively, as well as selecting the most appropriate descriptions from the list of similar problems.

   There are other methods the diagnostician can use to search the data base without contacting the support center, such as using Early Warning System microfiche.

4. Report New Problems

   If a similar problem description is not found, or the problem being investigated cannot be resolved with an existing correction, the problem is reported as a new problem. The objective of this activity is to ensure that the problem is accurately described and the necessary supporting documentation is gathered.

   The IBM Support Center Level 2 representative reviews the information obtained by the diagnostician and may suggest additional investigation. Level 2 will confirm that the keywords used to search the data base describe the problem accurately. Level 2 is also responsible for creating an APAR (Authorized Program Analysis Report).

   The primary responsibility of Level 2 is to act as a problem manager — that means applying the appropriate expertise at the appropriate time to resolve a problem expeditiously.

5. Correct New Problems

   The change team is responsible for locating the cause of the program problem and making the necessary corrections to the code. When the problem is corrected, they place a description of the problem and its correction in the software support data base. If the same problem is experienced by another user, its description will be found during the search of the data base.

   Members of the change team are IBM's product experts. For some products, the same individuals perform both the Level 2 and the change team activity, but have different responsibilities depending on which role they are performing.

Things are seldom as simple as this characterization suggests. These "activities" depict the intermediate results that lead to problem resolution. The objective of the diagnosis documentation is to direct you in achieving these results.

Information Packages

The purpose of going through the task analysis with you today is to give you a better understanding of what to expect from task-sufficient diagnosis books. You really need to know very little about the analysis in order to use the books effectively.

The conclusion drawn from the task analysis was that three types of information were involved.

1. Precise guidance information for the predictable activities.

   For each program product we can predict what information is vital for each type of failure and where that information is located. Locating this information yields a description that is suitable as a search argument.

   The diagnosis guide contains the information needed resolve similar known problems. Its product-specific procedures lead you to:

- identify the source of the problem,
- describe the problem in keywords,
- search the software support data base, and
- interact with the IBM Level 1 representative.

2. Selected reference information for free-form activities.

   Investigating new problems is a free-form activity. By that I mean we cannot predict what clue will provide the starting point for the investigation. The facts learned about the problem, following the procedures in the diagnosis guide, will determine where the investigation resumes.

   The diagnosis reference describes the program in terms of the processing it performs. Its primary function is to supply the information needed to investigate new problems, and contribute to a productive dialog with Level 2.

3. Information needed by the IBM change team to perform its task of problem correction.

   The change teams needs information that describes both the intent of the program's design and the sequence of its processing. This information need is met by the design documentation and is not part of the program's library.

Diagnosis Guide

The Diagnosis Guide contains a diagnosis methodology. Its procedures contain enough information to locate key facts to build a keyword description, but are not intended to provide an in-depth understanding of why the procedures are appropriate.

There is seldom only one way to diagnosis problems. But, for purposes of clarity and brevity, a method is selected at the time the procedures are developed. Writing procedures, actually putting the words on paper, is particularly challenging. For procedures must do more than just inform, they must lead. This sounds like a very obvious idea, but it is not uncommon for the action that needs to be taken not to be clear from the information provided.

Contents of Diagnosis Guide

The Diagnosis Guide is organized like this:

- Introduction

  The Introduction explains the concepts of using keywords to describe program failures and of using keywords to search a software support data base. The Introduction may also recommend preliminary activities to ensure that optimum diagnostic information is generated by the product.

- Component Identification Keyword

  The component identifier is the first keyword in the search argument. The order of keywords in the search argument is not critical; however, it does put you on the appropriate Level 1 queue, and is a more

  efficient search for the data base system. The release level is also used as a keyword to screen out problems reported against earlier releases.

- Type-of-Failure Keyword

  The main value of the Diagnosis Guide is its product-specific procedures. These procedures reduce the amount of product expertise required to diagnosis problems and contribute to the productivity of everyone who uses them.

  This section contains procedures for each major symptom, or "type of failure." These procedures describe what facts are important, and how to find those facts. A type-of-failure keyword describes an external symptom of a program failure, such as an abnormal termination, a wait, a loop, a message, or performance degradation.

  There is also a DOC keyword, but that is seldom needed. If it were needed, it would be used to search for descriptions of misleading or inaccurate documentation problems. Only documentation problems that are misleading in such a way as to cause lost time to other users are entered as DOC APARs.

  You can also influence the contents of a publication by sending a Readers' Comment to the writer. Errors reported through Readers' Comments are not entered in the software support data base, but are incorporated in the next update to the publication.

- Area-of-Failure Keyword

  The area-of-failure keyword associates a problem with a subset of the program.

  Procedures in this section continue the diagnosis, begun with the symptom, to narrow the area of the product that could contain the error. Ideally, the area of failure is isolated to a particular module or module equivalent. A program may have several "area-of-failure" keywords; such as, function name, module name, or modifier. A "modifier" keyword could be the command or type of statement being processed at the time of failure.

- Search Argument Procedure

  The Search Argument Procedure prepares the diagnostician to search the software support data base. It explains how to use the keyword string as a search argument and how to determine the service that has been applied to the system.

- Search Argument Techniques

  Search Argument Techniques are ways to vary the search argument — to narrow the search by making the argument more specific if the search yields too many possibilities; or to broaden the search by making the argument more general.

  This section is not a tutorial on searching the data base, but rather gives specific information relative to the search arguments developed in that diagnosis guide.

- APAR Preparation Procedure

  For most products, if an APAR is necessary, it is created by a support center representative. The purpose of this section is to make that first call to Level 2 as productive as possible. It describes the type of documentation you will need to investigate certain types of problems to prepare yourself for a productive dialog.

- Appendixes

  Frequently, there are long lists or tables of information needed to complete the procedures. This information is placed in an appendix to prevent the procedures from becoming cluttered. The titles of this reference information appear in the table of contents so that the information can be easily located.

- Index

  The diagnosis guide is not indexed because it is procedural. The table of contents provides an overview of the organization, and the page numbers indicate the beginning of each procedure. It is not appropriate to enter a procedure anywhere but at its beginning.

Diagnosis Reference

132

The contents of the diagnosis reference might sound similar to the program logic manual. It does contain many of the same types of information. However, the level of detail makes the diagnosis reference significantly different from a logic manual.

Contents of Diagnosis Reference

- Section 1. Program Overview

  Section 1 is an overview of the program. It describes how this program interacts with other programs. The overview names each function and briefly describes the type of processing performed.

- Section 2. Program Functional Description

  Section 2 describes the principal processing paths through each function and describes how the functions interact with each other. These descriptions generalize many small actions into a statement that summarizes the **results** of the actions. The level of detail of the information is quite limited when compared to that of the program logic manual.

  The information presented allows the user to follow the flow of the processing through the program, and to know when the program interacts with another program. From this section, the user can relate external input to the processing within the function.

- Section 3. Module Directory

  Section 3 shows the module-to-module communications within the program. Module information is organized alphabetically by module name and typically lists:

  - modules it calls.
  - modules that call it.
  - a brief description of the processing it performs.
  - names of the functional areas it is part of.

- Section 4. Data Areas

  Section 4 describes the data areas (or control blocks) used for communication between functions, programs, or parts of the system. Only those data areas are documented which contain fields directing flow of control from one function to another. These are the same data areas that are mentioned in Section 2 — the fields that are necessary to confirm successful completion.

- Section 5. Service Aids

  Section 5 describes the dumps and tools used to collect diagnostic information for the program. The task-oriented approach to presenting this information includes a full description of how to use the service aid, as well as when, or with what type of problem, the service aid is most valuable.

Organization of Diagnosis Reference

The diagnosis reference could be read from cover to cover to become acquainted with the program, but it is intended to be a reference manual. Each section has its own information objective and is organized to meet that objective. Each section stands alone. That means, depending on what is known about the problem — whatever piece of information you have — you can enter the appropriate section of the diagnosis reference.

Section 1 describes the functional areas in context of the type of processing they perform in the program.

Sections 2, 3, and 4 are organized alphabetically. This organization assumes you know the name of the item you are looking for.

Each section contains enough information to give you pointers into the other sections. So whatever clues you have will lead you to a full description of the related information.

The index is concise, because the entire book is organized for easy retrieval. The principal items in the index are concepts, since items arranged alphabetically within each section are not repeated in the index. This concise index contributes to its effectiveness. For example, if each mention of a module name were listed in the index you could not decide from the index which page contained the reference you were seeking. The table of contents shows page numbers for the items within the sections. When you enter a particular section, you know the type of information you will find there.

Using Diagnosis Books

The diagnosis guide and diagnosis reference complement each other. Neither are sufficient, alone, to meet your information needs. Let me take you through a scenario of how, and when, you might use these books.

- When you suspect a program failure, you first use the diagnosis guide for the component you suspect. Its procedures will guide you to investigate the failure and develop a set of keywords that describes it.

- Next, you will use the set of keywords to search the software support data base for a similar known problem. Hopefully, the search yields a similar problem with a correction which resolves your problem. If not, you may need to vary the search argument to broaden the search. This involves omitting one or more of the keywords that are particularly limiting.

- If you decide that you have searched the data base efficiently without locating a correction for your problem, you will report the problem. Your task may end there or you may do some additional investigation.

- To continue investigating the problem, you will use the keywords you selected using the procedures in the diagnosis guide to index into the diagnosis reference.

  - For example, if you have the name of a module, you would enter the diagnosis reference at Section 3. In the module directory, you would find a brief description of the processing that module performs, and the name of the function in which it participates.

  - You then would go to the description of that function in Section 2. Because you know what processing the module performs, you can determine the role the module plays within that functional area. You will also gain additional insight into what other processing was involved at the time of failure, or just preceding the failure.

- In some types of failures, you may know the nature of the processing being performed, but need to know what functional area was involved and the modules that participated. In that case, you would review Section 1, to determine the name of the functional area which interests you, and go to Section 2 to read about the processing and "footprints" that indicate successful processing.

Summary

There are many ideas about how to improve the diagnostic process. Evaluating the effectiveness of each idea in a scientific way is next to impossible — unless, of course, some of you would like to volunteer to do a comparative study. You could handle each problem in five different ways and have a contest between your teams to see which method took the least amount of time. Seriously though, there are so many variables with each problem that a time comparison by method is not feasible even if someone cared to try it. So how do we pick a method to improve the process? The answer is that we probably won't pick just one method.

I have just described a method that **can** increase your productivity while investigating problems and working with the IBM Support Center. I have described the organization and contents of two books — the diagnosis guide and diagnosis reference. This task-oriented information addresses many of the current difficulties encountered while diagnosing problems.

- Users no longer need to understand the logic of the program to understand what information is significant and how to find it.

- Users can quickly build their own expertise by following detailed procedures that locate significant facts about a failure.

- The success rate of searching the software support data base is increased by describing problems in a consistent way.

- Interaction with the Support Center is streamlined by establishing a common starting point for that interaction thus increasing the productivity of the first telephone call.

Diagnosis information is not the ultimate solution for resolving program problems. That position is reserved for automatic data capture and automatic error recovery; or better yet, error-free code. However, diagnosis information does fill the gap between where we are today and where we want to be. I like to believe that having developed this information in book form is the first step toward a more perfect solution.

I appreciate you spending your time attending this session to listen to me discuss my favorite project. I hope that you feel that it has been informative. This session is, however, just the first step. There are four more steps you should take when you return to your office to make the time you have just spent truly productive:

1. Add to your library the diagnosis books for the products you have installed.

2. Become familiar with their contents before a problem occurs.

3. Use the books to diagnosis problems and to work with the IBM Support Center.

4. Use the Readers' Comment Form to let the writer know the strengths as well as the weaknesses you found in the information.