Help Usability Enhancement Project -A Proposal

I.The Problem(s)

There are several perceived problems typically associated with user documentation in general and with on-line help in particular. Printed documents and help systems produced by Application Services have a reputation for quality, but it is valuable to examine these problems so that we can establish ways to continue to improve our product.

A.Usability - The Problem

The first problem is apparent to the application user community and concerns "helpfulness" of help systems. Even after the help system has been reviewed by developers and the representatives of the user community, the users themselves might find the help system not as useful as it could have been. This problem is generally referred to as a usability problem. A solution to this problem is presented in section III A of this document.

B.Development Effort - The Problem

The second problem relates to the amount of effort that it takes to produce a help system. That is, help systems ought to be developed in less time. There is no absolute solution to this problem, because the tendency will always be to drive the development cost of documentation to 0. This goal can only be achieved by not developing documentation. There are ways, however, to reduce development costs by a meaningful amount. An approach to this problem is presented in section III B of this document.

II.The Interrelationship of the Problems

A.The Relationship of The Usability Problem and The Development Problem

It seems intuitive that these problems are related in a mutually antagonistic way. That is to say, the more usable a help system, the longer it takes to produce it, and the less time spent in production of a help system, the less usable it is.

It is important to note, however, that the time spent producing a help system cannot automatically be judged by the size of the help system, that is, the number or topics or of words in the system. Often, it takes more time to write a concisely worded and relevant document that it does to write an unfocused and poorly organized document. The software technical writer's goal is not to turn out words in volume, but to produce written tools that allows the typical user to get his or her work done so he or she doesn't have to think about the application that much. Thus it may take longer to produce a shorter document.

But there are ways to increase help system usability and also decrease the amount of help developer time needed. To approach this problem, we must be aware of the interrelated nature of software and documentation. And we must keep in mind a user-centric view of application and help development.

B.The Relationship Between the Application's User Interface and The Documentation

Software and user documentation have always been more interrelated than most programmers and technical writers believe. In the days of command-driven software, it was impossible to operate software without documents listing commands. Today, help systems are an integral part of the user interface.

The connection between these two segments of the user's experience is often made haphazardly by programmers and technical writers. Often, the program is written based on the programmer's second hand understanding of the users' needs, and then "thrown over the transom" to the technical writer, who must then try to understand the programmer's design of the user interface, and translate it into something meaningful for the user. The technical writer may have only a vague notion of whom the user is. It is left to the user to make sense of these two parts and somehow integrate the whole package in his or her consciousness.

Because the documentation is supposed to explain the application to the user, intrinsic problems with the user interface are often perceived as a failure of the documentation. This may be expressed by the user with phrases such as "I guess the program is really powerful, it cost a lot and there are a lot of options on the menu bar, but the documentation is inadequate, because I can't find out how to do what I want to do." Never mind that the common task the user wishes to perform might require accessing menu options hidden in different submenus named in ways that would not be obvious to the intended user. A simpler and more natural user interface would enable the writer to produce a much simpler and more easily navigated help system or printed manual.

III.The Solutions (s)

In the above two sections, two interrelated but distinct problems were defined. This section is divided into two parts addressing each problem:

- Usability - Our Proposed Solution
- Development Effort - Our Proposed Solution

A.Usability - Our Proposed Solution

To produce more usable help systems (and applications) in a more timely way, there must be better contact between users, developers and writers. Developers and writers will then be better able to craft their product to meet real user needs, rather than inferred needs.

There is no magic way for software developers and technical writers to have an accurate picture of what users want and need. This knowledge can only be gained by contact with the user community. This means real users, not managers of real users, or consultants who have talked to managers of real users.

Of course, there are many ways that this contact can be carried out, and is currently being carried out. But haphazard contact is not an efficient way to extract the kind of detailed knowledge needed in a timely way. For example, it is common to find that the end-user normally only comes in contact with a new application when he or she attends a training class.

At this time the trainer may gain insight into what the development team should have done to make the product more usable. Currently in R&D CS, the feedback loop to the developers and technical writer at this point is only very rarely made, if at all. And even if the developer gets to hear second hand about a problem with the interface, it is too late to do anything about it with the current software version, because it has already been rolled out, or an immediate release date has been set.

To enhance help system usability, we propose strengthening the user-developer interface by enhancing existing points of contact. For example:

- The training classes can be used by help authors in a planned way to observe how users operate the software.
- Implementing a more efficient and structured mode of transfer of information (TOI), by instituting usability testing. Usability testing is a procedure used to quantify an applications ease of use. It measures the effort required on the part of the user to get work done.
- Gathering feedback by including a feedback form in each help system that can be used to send comments to the help system developer.
- Promoting closer integration of applications and help systems by implementing context sensitive help and performance support systems.
- Changing the SDM to encourage a more cooperative relationship between application and documentation developers within the life cycle process.

1.Training Session Observation

Training sessions provide a natural laboratory for observing actual users in action. The help author should be given notification and observation access to training classes. Classes may be held at various times during the development cycle, and are useful to help authors whenever they occur. For example, training that takes place to prepare for acceptance and validation testing would aid the help author in completing the initial help system. Also, the help author would be in a position to suggest to the developer any last minute changes to the user interface that appear as critical to system usability. Training that takes place after the product rollout can aid the help author in revising the help system, and also as a guide to the design of future help system, and the establishment of common format standards between systems.

The help author should observe several different training classes, to see if the same concerns come up more than once. Often, trainers hear the same complaints and questions by users over and over again, making the existence of a problem apparent.

The help author should take notes during all training sessions in order to produce a written record of the problems and suggestions provided by the users. The notes from these observations could be summarized into a Training Sessions Observation Report that would help the developer revise the user interface to solve problems discovered by observing training. Alternatively, a less formal set of notes can be created and shared with developers. If these problems cannot be adequately addressed by developers, the help author can explain them to the user in the help system.

2.Usability Testing

A more systematic and productive form of developer-writer-user interaction is provided by usability testing. We propose instituting a program to do usability tests on software and documentation.

But can't we improve the user satisfaction with a help system by only testing the help system? The application has already been written, wouldn't it be far simpler just to tune up the help system a little?

Here is a quote from "A Practical Guide to Usability Testing" by Joseph S. Dumas and Janice C. Redish (Intellect, 1999, p.29)

"When we first started doing usability testing, clients sometimes asked us to `just test the documentation,' because another group `owned' the software. We learned very quickly that in any usability test, even if you are focusing on the documentation, you learn a tremendous amount about the software (and vice versa). We couldn't *not* report what we had learned - and our documentation clients also realized that their software colleagues had to know what we had learned about the interface. They also realized how futile it was to think of the documentation and software separately. Good changes to both products and processes came out of those tests."

But don't the developers already know what users' want? In R&D CS, after all, developers have benefited by carefully adhering to the SDM, preparing all the required reports, and have spent time talking to the management of the average user. Dumas and Redish question this assumption (p.82) "Software engineers are not very good at uncovering usability problems, even when they are given a short lecture or report on principles of human computer interaction…people who have been developing complex products but have never systematically watched users set up, learn, or use those products find watching a usability test fascinating, shocking, sometimes humiliating and painful, but invariably eye-opening."

This type of knowledge about users can only be gotten from users. Those being tested must be from the real set of users. Managers, developers, technical writers or trainers are not an appropriate set of people to test, unless the application is aimed at these groups.

Usability tests should not be thought of a sort of validation test at the end of the development cycle. These tests must occur early enough in the development cycle to be able to impact on the completed product in more than a cosmetic way.

Application Services (the section formerly known as Technical Communications) is well placed to help make Applications Development's products more usable. Not only do we have experience in designing and implementing help systems, we also are the authors of the SDM and are therefore familiar with the entire software development cycle.

Over the last several decades, the field of usability testing has developed a methodological approach. There is quite a bit of variation in this methodology, but there is a general agreement that accurate recording and reporting of test results is essential. We should begin a usability testing program in as modest a way as possible. The testing would take place after prototyping and before validation takes place. This would entail developing a usability test plan, selecting users to be testing, and careful observation of the users' interaction with the application and help system.

We propose including in the SDM a requirement for 2 usability tests for each application, consisting of:

1.A usability test of a prototype of the application, (or a "paper" version of the user interface, if a prototype is not yet available) and a prototype of the help system. This test may be prepared by the programmer and/or the technical writer. It should involve 3-6 intended users. Each user is tested separately, and the whole test cycle can be completed in 1-2 days. These test subjects should be from the actual population of prospective users. These users should not have had previous exposure to the application. This test is designed to catch early design problems that can be corrected at this stage with little distraction to the development schedule. This test should be observed in whole or in part by both the developer and writer. The person writing up the results should be there for the entire test.

2.A usability test when the user interface is fairly stable and a complete draft of the help system exists. This test should also involve 3-6 users. The test will be more extensive than the previous test, so the test cycle may take a week. The results of this test should contribute to the modification of the user interface as well as the documentation.

Employing Usability Testing Results

The results of usability tests must be presented in a way that is immediately usable by developers and writers. A test report must be prepared that presents the results unambiguously and analyzes them in a usable way. The results of any usability test should become a required input to a project at defined points in the development cycle, after prototype development and before validation begins.

Expanding a Usability Testing Program

As we become more proficient at testing, and begin to experience the benefits, there is room to expand our efforts. Many software development organizations have established usability testing programs that use dedicated equipment to perform more useful and extensive tests.

Audio or video recorders are often used to capture usability tests, making it easier to accurately quantify results. Various types of software are available to aid in testing. Also, many software organizations have established usability testing labs that may include one-way mirrors and other features.

Our usability testers could gain proficiency in the field by attending usability testing training, given by various institutions around the country, making contact and trading learned insights. (We have already been invited to visit the usability testing lab of Trellix Corporation in Concord MA.)

Usability-Oriented Requirements Defined in the SDM

A number of modifications to the System Development Methodology (SDM) should be made to encourage users and application developers to consider what kinds of user-oriented documentation will be required and to involve help development personnel earlier in the project lifecycle. The recommended changes to the SDM are as follows:

- The **Documentation Plan** is a required document that is used in the Requirements Analysis Phase to plan, specify and record the agreement between users and developers on documentation that will be produced during the project. It currently includes *User Manual* as one of the possible documents. It should be modified to include other user-oriented documentation forms, including *Online Help*. A member of the Application Services section should be included in the required approval signatures for this document.
- The **Requirements Analysis Document** is used to document the user's requirements of the application being developed. It is developed jointly by the users and the application developers. We recommend that a section be added to this document that describes the user's documentation requirements. This section should specify not only what kinds of user documentation will be required, but also include a general description of the design of this documentation. In the case of online help, this may take the form of a topic design and description of how the help system interacts with the application. If usability tests are to be conducted, this requirement should also be included. A member of the Application Services section should be included in the required approval signatures for this document.
- The SDM should include standards and a "shell-like document" for Online Help systems. This can be based on the style sheet that Application Services has produced for use in the GCD help systems.
- Usability tests of the application and user documentation (help system, user manual, etc.) should be performed before validation begins. The development of a shell for a **Usability Evaluation Memo** will permit Application Services staff to indicate that the system has been tested in regards to usability issues. This document should include signoff approval from the user, developer and Application Services.

3.Feedback Forms

We propose the inclusion of a feedback form with every AS-produced help system. The form would be formatted like any other WinHelp topic, but will allow comments to be entered as with a web-based form. Responses would be emailed to AS. This requires the installation of a custom .dll file in the system directory of the local workstation. The local workstation must be able to connect to the WARWeb. We will develop a working prototype of this system by 4th Quarter 2000.

4.Help System Integration

We propose the tighter integration of applications and help system. This can be done in the following order:

1.Minimally, and first of all, all applications should be made context sensitive by having the developer use the programming "hooks" provided by the help author. This gives the user immediate access to relevant help information.

2.Application Services will produce a Help Guide For Developers, which will aid programmers in linking their applications with Help systems.

3.Developers should put Help buttons on all windows, to be linked to procedural topics in the help system. Developers and help authors must ensure that appropriate procedural topics are produced for each application window.

4.For high-profile applications, wizards and online tutorials authored in the help environment should be developed.

5.HTML Help can be used to create a "performance support system" that pushes critical information to the user as he or she attempts to perform a task. This requires very close coordination between developers, help authors, and the user community from the very beginning of a project.

According to Wextech technology, a leading help authoring tool developer, "With the advent of HTML Help and the ability to embed Help directly inside an application, there's been an increased interest in creating Help systems that are seamlessly integrated with their host applications. By blurring the line between the application and the Help that supports it, and by developing Help that automatically responds to user actions, application developers and Help authors now have the ability to develop true electronic performance support systems."

In the software industry, the growing reality of more closely integrated "performance support systems" requires a more tightly integrated application development-help authoring effort. To stay current, we are going to have to find ways to accomplish this within the Application Development environment.

B.Development Effort - Our Proposed Solution

To begin to think about reduction of development time, it is important to consider how a writer's time is currently spent. After months of technical writing effort, the end product of a user manual development process might be a manual of several hundred pages. The typical technical writer, with average touch-typing skills, could easily retype the manual in several days. So, clearly, the vast majority of technical writer's time on a project is not spent typing, or a least not typing the final version of a project.

So why doesn't the writer just get on the ball and type out the manual first, and save several months work? Obviously, the bulk of the time expended on a document or help system involves trying to understand the application. This is an iterative process of using the application, interviewing subject matter experts (SMEs) and developers, writing procedures, and testing the procedures. Writers discover that you never really understand all features of an application until you have to document them. Without usability testing, writers often discover

performance aspects of the user interface of an application of which even the developers are unaware.

Therefore, the most effective way to reduce development effort is to reduce the amount of time a writer needs to understand an application. Ideally, the help writer shouldn't have to spend so much time figuring out the application; the application developer should supply the bulk of the information, and the help writer should just have to organize and format the information to make it easily palatable for the application's users. Then the writer can supplement the design document with information gleaned from usability testing and/or interviews with users.

However, programmers and writers naturally perceive applications from opposite ends of the telescope. Programmers have an analytic view of applications; that is, they see them as built up from the smallest functioning units, whose internal functions need to understood. Writers have a synthetic view of an application; that is, they need to see it as a functioning system. So writers are more ideally placed to perceive the application as a whole, as do the users.

In the real world, of course, programmers are not technical writers, and they tend to resent writing the existing mandated SDM documents, let alone assisting in writing user documents. They tend to see time spent writing SDM documents as time away from their "real" job.

The solution to the problem of reducing development effort, therefore, is the same as the solution to making help systems more useable. That is, increased communication between developer, user and help author reduces the development time by giving the help author a deeper understanding of the application from the standpoint of both the developer and the end user. The help author is also more likely to be able to know where to efficiently obtain information needed to rapidly document an application.

So the same concrete recommendations given in Section III A of this paper to make systems more usable with also decrease development effort.

IV.Benefits

If this proposed program were to be implemented, R&D CS would receive the following benefits:

- Increased usability of our applications. Users would have less need to consult online help systems, and therefore have less of a chance to be disappointed with help. But more importantly, user productivity would increase within the organizations using the application. Also, users begin to recognize consistency between applications, flattening the learning curve.
- Increased user satisfaction with online help. Applications become more usable because the documentation is easier to use. Less time is spent on support efforts.
- In the long run, decreased application coding time and help authoring time. As programmers and writers increase their understanding of the user and his or her needs, less time is spent building applications and documents that must eventually be changed to gain the approval of the organization for which the application is being written. Programmers begin to be able to better anticipate future directions of an application, and design current

releases with a design philosophy more consistent with future extensions requested by the user community.

- Usability tests will help to create reliable metrics that can be used to objectively measure real productivity on the part of developers and help authors.
- The earlier Application Support is included in the development cycle, by doing usability testing, the flatter the learning curve for the documentation effort. Even if the user interface of the application needs to be changed, because of the results of usability testing or any other factor, help developers gain benefit from early exposure to the application, making it easier to understand the new interface.
- Better communications between all the players in application development, developers, users, help authors, management.
- A more open documentation process.

V.Conclusion

We believe that by bring together user, developers and help authors in a planned and systematic way as outlined in this document, we can increase the usability of our applications and help system, as well as reduce system development effort.