CODA: a Concept Organization and Development Aid for the research environment

by James A. Dewar and James J. Gillogly [1]

Introduction

In 1982, the authors and colleague Morlie Graubard were led by their experiences to wonder what the role of computers might be in policy research (our primary occupations). The role of computers in analysis, particularly quantitative analysis, is extensive and well documented. Computerized text editing (as in the preparation of this manuscript) is quickly replacing the typewriter throughout the industrialized world. Artificial Intelligence researchers at Rand and elsewhere are exploring the possibility that computers might do some of a researcher's thinking. Data retrieval systems are bringing vast stores of data within the reach of a researcher's specialized interests. But it seems as if there ought to be other ways that the power of computers could help in the research process.

We focused on the part of the typical research project that involves reading vast amounts of information related to a research topic and remembering those parts which are pertinent to one's current thesis. This part of research predates computers and is currently supported, to some extent, by a variety of file management and data base management systems. The idea of the computer acting as a long term memory for the researcher seemed a natural combination of the speed and memory of the computer with the more subtle, synthetic powers of the researcher.

[1]Paper presented at the 1986 IASSIST Conference, Santa Monica, Calif., May 22–25, 1986.

In looking at several extant storage and retrieval systems, however, they seemed to be directed either toward very large data bases or at personal computer applications. As such, each seemed to have important limitations when applied specifically to the problems of the individual researcher. This led us back to take a look at the research process itself in an attempt to define those aspects of search and retrieval that most suited the policy researcher's environment.

The key seemed to be that the research process of an individual was essentially an interactive process characterized by both a growing data base and a series of failed attempts at organizing the data into a coherent whole (followed ultimately by a successful attempt, of course). This led us to a hypothesis about the utility of computers "optimized" for the policy research process, and from there to a list of desirable characteristics for the associated computer aid.

HYPOTHESIS:

Computers can aid the policy research process by acting as a long term memory (storage and retrieval facility) for the researcher's growing data base and changing concepts.

The realization of this hypothesis in the form of computer software specifications required constant referral back to the research process and an appreciation of the limitations of modern computers. The following list of desiderata reflects the results of that effort along with our justifications for each.

Desiderata

1. Quick Boolean tag searches.

   In retrieving data, we have found with other retrieval systems that even a 1 to 2 second delay between the request and the results was very distracting to a researcher's thought processes. Boolean searches involve the use of the logical connectives AND, OR, and NOT; require longer than single search requests, in general; yet should respond just as quickly as single search requests if the researcher is not to be distracted from the problem at hand. It should be emphasized that the requirements here are placed on TAG searches—full-text searches are exempt from this specification because we thought that the primary search method would be by tags and that the researcher would not mind delays for the occasional full-text search.

2. Flexible tagging rules.

   The process whereby a human retrieves data from his/her own memory is ill understood, except that it seems to involve a variety of mental processes. Retrieving data from a computer is more limited, but we wanted the researcher to be able to "mark" or tag data for retrieval with as much flexibility as possible. S/he shouldn't be restricted to words that appear in the text or single word tags or ...

3. Powerful tag changing capability.

   This would seem to be the key for a policy researcher. As one's concepts change, the relevance of one's data to the new concept usually changes also. We wanted the researcher to have the ability to make wholesale changes to the markers or tags on data. This seemed to imply two capabilities: 1) The ability to do both full-text and tag searches and then to "prune" or shape the resulting list of data records, and 2) the ability to then make changes in the tags of those records all at once.

4. Recall by data capability.

   This is a common capability in storage and retrieval systems and seemed a useful adjunct for a research environment that depends heavily on dated articles. This also includes the ability to do comparative operations on dates (such as, all dates from date 1 to date 2, etc.).

5. Data entry from keyboard or file.

   This would seem to be useful for two reasons: 1) Many researchers already have an electronic data base that would benefit from this capability, and 2) Rand (among others) now has the capability of obtaining the results of large data base searches in electronic form, which could then be easily entered in that form.

These, then, formed the basic requirements that we thought defined a computer capability "optimized" for the needs of the policy researcher. With these specifications in mind, we again looked at available software for file and data base management. Without claiming to have done an exhaustive search, we did talk to a variety of data base specialists, search the on-line literature, and visit computer shows. In the end, we were sufficiently disappointed with lack of matching between our desiderata and the available software that we decided to build our own. As a point of interest, the common mismatches were either slow response times (typical of microprocessor based systems) or insufficient capability to do easy, wholesale changes to the tags in the system (typical of systems aimed at very large data bases)

The resulting system was called CODA (for Concept Organization and Development Aid) and that system is the topic of this paper. In the following sections we will describe the prototype system we built in order to test our hypothesis, the system's capabilities and limitations, some of the details of its user interface, what we have learned both from the building and testing of the system, and, finally, some thoughts on further capabilities that appear to be amenable to computer implementation and that might aid the policy researcher.

The Prototype

The CODA program most properly qualifies as a file management system aimed at small data bases and a very limited number of users. The specifications followed in its development were the desiderata listed above, and the reason for developing it was to test the hypothesis that the specific task of policy research could be improved by an appropriate computer aid. It is a system designed and implemented by users (policy researchers) for testing some concepts about the users' environment. As such, there are some specific things that CODA is NOT. It is not a full data base management system for general use, it is not particularly suited for large data bases or numerical processing, and it is not a product that the Rand Corporation is trying to sell.

With those caveats in mind, we were interested in describing CODA to this conference both as a way of highlighting the data base needs of one special segment of the user community, and as a way of eliciting feedback from the professional community of data management specialists.

Before proceeding with a description of CODA, it is important to discuss briefly the major "buzz words" that we have come to use in our description of the system.

tag: This is a user-supplied word or phrase that is typically used in CODA for retrieving a data item. In other systems this is called a keyword. 'Tag' is used in CODA because it need not be something that actually appears in a record. A given record can have many tags.

record: This is another word for datum and refers to an individual recallable item of data in the system. In CODA, a record is any (unformatted) text that is of interest to the researcher. This is intended to include anything from a single word to several paragraphs, from a table of numbers to a collection of symbols, etc.

hit: This is any record that contains a specified search tag (or tag combination). If, for example, CODA were commanded to return all records having 'important' as a tag, CODA would return that it had found, for example, 34 'hits' or records that contained 'important' as a tag.

index: This is undoubtedly the ugliest of CODA's buzz words. There are two kinds of indices in CODA: date indices and others. Date indices are a way of grouping different kinds of dates for recall. In this way, the user is able to differentiate between, for example, the date on which the material in a record was published and the date on which it was entered into the system. The principle behind the other indices is much like that of the indices in a books. It is a way of grouping tags both for display and for search purposes. For display purposes, the intention was to give the user a tool akin to the Author index found in some books. It is much easier to look up a half-remembered author in a smaller author index than it is to do so in a large full index. In addition, one can retrieve, specifically, an author's works, for example, rather than retrieving his works as well as anything written about him.

Technically, the prototype CODA system has been written under UNIX (4.1 BSD) and is running on a VAX 11/780. It is written in C and uses the Curses screen management package. Data entry from the keyboard uses the Rand editor, a full-screen editor used by most of Rand's researchers and secretaries. CODA provides a menu-driven interface to users with a variety of terminal types, including Rand's standard Ann Arbor terminals (in several models) and personal computers connected to the VAX via modems. By servicing all of Rand's terminals, we were able to enlist a variety of Rand researchers to use CODA and feed back their comments as to its utility.

To achieve the recall speed specified in the desiderata, the tags are loaded into a hash table in memory with linked lists pointing to the data associated with each tag. The tags themselves are C strings, which allows for a wide variety of things the system will recognize as legal tags.

Capabilities and Limitations

CODA is a menu-driven system because it was our feeling that the typical non-computer-oriented researcher would find such a system the quickest and easiest to learn and the most comfortable to work with. The capabilities and limitations of CODA are best demostrated directly through the menus that constitute the user interface. Below are the nine CODA menus arranged in a hierarchy based roughly on the connections between the menus.

```
MAIN MENU
        DATA ENTRY Options
        DATA RECALL Options
                HIT LIST Options
                        RECORD Options
                        CHANGE HIT TAG Options
                        REFINE HIT LIST Options
        TAG CHANGE Options
        TAG INDEX CHANGE Options
```

Figure 1. Basic CODA Menu Structure

Capabilities

In each menu, typing '?' will access a help file that describes the options in that menu, and typing '<ESC>' will get the user out of CODA entirely. The MAIN MENU has an introduction to CODA for the first time user, and is the main access path to the four menus in the first indentation in Figure 1. Of those four, two (TAG CHANGE and TAG INDEX CHANGE Options) relate to tag and index changes throughout the entire data base. The other two (DATA ENTRY and DATA RECALL) are the "guts" of the system, and, along with their submenus, they deserve more detailed mention in order to give the reader a good feel both for what CODA is and what it is not.

** DATA ENTRY Options **

a. Set up session tags
b. Enter data (from keyboard)
c. Transfer data (from file)
d. Back to MAIN MENU (*)

Option?

Figure 2. Data Entry Menu

Figure 2 is the Data Entry menu, exactly as it appears in the bottom window of the user's CRT. Data entry from the keyboard is done in a full-screen two-window editor. An example of a record and its tags (as they have been entered in the two-window editor) is shown in Figure 3.

AGM-86 ALCM

The AGM-86 air-launched cruise missile is a small unmanned winged air vehicle capable of sustained subsonic flight following launch from a carrier aircraft. It has a turbofan engine and a nuclear warhead, and is programmed for precision attack on surface targets. When launched in large numbers, each of the missiles would have to be countered, making defense against them both costly and complicated. Additionally, by diluting defenses, the ability of manned aircraft to penetrate to major targets would be improved. Small radar signature and low-level flight capability enhance the missile's effectiveness. Production is expected to total 3,418 missiles between FY '80 and FY '87, with deliveries to be completed in FY '89. Initial funding for 225 AGM-86B ALCMs was provided in FY '80; 480 more were approved in FY '81, 440 in FY '82, and procurement of 330 is planned for FY '83. SAC's 416th Bombardment Wing at Griffiss AFB, N.Y., became the first Air Force unit to attain operational capability with ALCM in December 1982, with 12 missiles fitted externally to each of its 14 B-52Gs. It has been followed by the 379th Wing at Wurtsmith AFB, Mich. Other units to receive ALCMs are at Grand Forks AFB, Ark. Ultimately, each B-52G is intended to be modified to

have a bomb-bay rotary launcher

#General: weapon system; ACM-86; ALCM; nuclear; nonnuclear; guidance;
inertial; TERCOM; speed; range; cruise missile; funding; air-to-surface;tech data
#Journal: Air Force Magazine
#Author:
Date: 5/31/83

Figure 3. Data Entry Example in a Two-Window Editor

The first line of each record is printed out on the hit list in data recall, so it is commonly used as a summary line for the record. In the tag window, the indices are identified by '#' or '' and ':' and the tags are separated by semi-colons. These are the only reserved symbols in CODA.

Session tags (option a. in Figure 2.) give the user the ability to set up tags to be put on all data items that are entered until the session tags are changed. This saves the user time in cases where several items to be entered will have tags in common. After being set up, the tags will automatically appear in the tag window in succeeding data entry calls. Session tags are also useful in data entry from non-CODA files. In order to enter data into CODA from a file, the file must have records separated by a single delimiter that doesn't appear elsewhere in the data; the records will be entered into the data base with the session tags set up for that purpose.

** DATA RECALL Options **

a. Enter tags for search
b. Look at all system indices.
c. Look at all tags (for an index)
d. Back to MAIN MENU (*)

Option?

Figure 4. Data Recall Menu

The Data Recall menu, shown in Figure 4 as it appears in the bottom window of the user's screen, leads into the hypothetical heart of the system — data retrieval and manipulation. In any menu involving tags, the user has the option of looking at the current system indices and a glossary of the current tags. Searches can be done by tag, by full-text search, or by Boolean (AND, OR, and NOT) combinations of the two. After the search expression has been entered, CODA looks up the pieces

of the expression in the tag hash table and offers to do a full-text search if it can't find them. It will also do a pure full-text search. A sample CODA response is shown in Figure 5 (in this case the special expression 'all' was entered and CODA returned all records).

```
There are 278 hits for this expression.
Expression: all
-------------------------------------------------------------------------
 1. LGM-30F/G MINUTEMAN
 2. MGM-118A PEACEKEEPER (MX)
 3. AGM-69 SRAM
 4. AGM-86 ALCM
 5. BGM-109G GLCM
 6. AGM-45A SHRIKE
 7. AGM-65 MAVERICK
 8. AGM-78 STANDARD ARM
 9. AGM-88A HARM
10. GBU-15
11. AGM-109H TOMAHAWK (MRASM)
12. ALMV
13. AIM-120A (AMRAAM)
14. GBU-15s DESTROY SIMULATED MISSILE SITES
15. CRUISE MISSILE: WONDER WEAPON OR DUD?
16. BUSINESS OUTLOOK: CRUISE MISSILE PROGRAM
17. INGLORIOUS FAILURES PLAGUE PERSHING-2
18. MaRV: KEEPING A NEW NUCLEAR GENIE IN THE BOTTLE
-------------------------------------------------------------------------
                        *** HIT LIST Options ***
a. Look at specific record          e. Change tags on these hits
b. Output specific recore(s)        f. Refine this hit list
c. Delete specific record(s) from hit list   g. Back to DATA RECALL (**)
d. Expunge specific record(s)       h. Back to MAIN MENU (*)
Option?
```

Figure 5. CODA Hit List Example

In some sense, the posited utility of CODA should be measured by the percentage of time the user spends in and around the Hit List menu. It is here that one's ability to rearrange and shape one's "long term memory" is most evident. It is here that we thought the researcher would spend time accessing, modifying and re-marking data in light of new information or insights. Three of the Hit List processing options (a, e, and f) lead to separate menus. The others lead to further prompts from CODA and will be described first.

The records in any subset of hit list records can be output (option b) in their entirety or to the first blank line (allowing for summary outputs), to a file or directly to the printer, with or without the associated tags. Any subset of the hit list can be deleted from the list (option c) as a way of shaping the list for output or tagging, or can be expunged from the data base entirely (option d). Paging through the hit list can be done with the +page and -page keys to be found on most terminal keyboards.

Any record can be seen in its entirety (option a) by entering its numerical identifier. This leads to a separate menu (Figure 6). In additional to mimicking the output, delete and expunge options of the Hit List menu, the user can edit the record and its tags (as in data entry), page through the record (if it is greater than one screen-full) with the +page and -page keys, or page through the full records on the hit list and its options.

**** RECORD Options ****

a. Edit this record (and tags)
b. Go forward to next hit list record
c. Go back to previous hit list record
d. Output this record
e. Delete this record from hit list
f. Expunge this record
g. Back to HIT LIST Options (***)
h Back to DATA RECALL OPTIONS (***)

Option?

Figure 6. Record Options Menu

Option e on the Hit List menu (Figure 5) allows the user to make tag changes on all records in the hit list simultaneously. This leads to the menu shown in Figure 7. Adding, deleting, or renaming a tag takes place online and is reflected thereafter in any functions that involve the tags.

**** CHANGE HIT TAG Options ****

a. Add a tag to these hits
b. Delete a tag from these hits
c. Rename a tag on these hits
d. Look at all system indices
e. Look at all tags (for an index)
f. Back to HIT LIST Options (***)

Option?

Figure 7. Hit Tag Change Menu

In addition to shaping the hit list by deleting individual hits, option f in the Hit List menu (Figure 5) allows the user to refine the hit list using the Boolean operators AND, OR and NOT. This leads to the menu shown in Figure 8.

**** REFINE HIT LIST Options ****

a. Current hits OR those with...
b. Current hits BUT ONLY those with...
c. Current hits BUT NOT those with...
d. Look at all system indices
e. Look at all tags (for an index)
f. Back to HIT LIST Options (***)

Option?

Figure 8. Hit List Refinement Menu

One of the major purposes of options a through c is to allow users unfamiliar with Boolean operations to use them nonetheless, by having them translated into "natural" English. Option a corresponds to the Boolean OR, option b to AND, and option c to AND NOT. With these three, the user familiar with logic can build up any desired Boolean refinement and the novice should, with successive refinements if necessary, be able to do the same thing. With each option, the user enters a search expression (as in data recall). CODA then performs the appropriate operation and gives the user back the refined hit list, the refined search expression and the Hit List menu.

While CODA has others, those major capabilities listed above are those most relevant to the hypothesis that we hoped to test with the system. In addition to the capabilities mentioned, there are limitations worth mentioning also. Some are a direct result of the design chosen to meet the desiderata and others have more subtle origins.

Limitations

Perhaps the most serious philosophical limitation on the system is that, while it is "optimized" for an individual policy researcher, its assets become increasing liabilities as the number of researchers using the same data base increases. This phenomenon is well known to large data base systems with a large number of users. The more users there are, the more important it becomes to limit both the number of people that can make changes to the system retrieval parameters as well as the frequency with which any changes can be made. CODA is specifically directed at, and can be used profitably by, only a very small number of users PER DATA BASE.

Although not necessarily a limitation, it is true that CODA is not a very "smart" system. Again, it was the intent of the design to leave the "smarts" in this man-machine system in the "man" since that is what "he" does best. The machine has been directed to do what it does well — store and retrieve data and make changes to the data on command.

The most noticeable limitation to the user is that the system can take a long time to load. As currently implemented, the hash table is built each time CODA is operated. Hence, the larger the data base and the heavier the machine processing load, the longer the system takes to load. With a 2000 record data base and several tags for each record, the loading can take several minutes on a heavily loaded machine. This is a "start-up" cost and is mainly due to the time required to link the data to their respective tags. While this could be done more efficiently, doing it in C will not be as efficient as it would be in a list processing language such as LISP.

In addition to loading torpidity, the hash table implementation leads to fairly large RAM requirements and to some inefficiencies in truncated searches. While a virtual machine is not overly sensitive to the RAM requirements of any program, the current CODA system, set to handle 10,000 records and a hash table set up of 10,000 tags, takes up about 700K in the VAX 11/780. Again, some storage requirements could be gained and efficiency sacrificed by putting the hash table on disk.

Another limitation from the hash table approach is that truncated searches (which aren't currently coded) can't be efficiently coded. An example of a truncated search is to look for all records that contain any work starting with 'bomb'. This would retrieve records with the words 'bomb', 'bomber', 'bombing', 'bombardment', 'bombast', 'bombazine', etc. In full-text search mode, CODA can do this quite well, but very slowly. In indexed tag searches, however, it requires searching through the entire hash table, which defeats the purpose of the hash table approach and takes decidedly longer than a single tag search. This is really only a limitation on the tag searches, and tends to be a problem with most other approaches that are geared for speed.

The above capabilities and limitations are basically the ones that were designed into the system. They are basically the things that can be said about the system AS SOFTWARE. It is important to reflect upon them in terms of trying to understand the tool that was developed to test our original hypothesis. In order to determine if this tool, CODA, was indeed of significant use of policy researchers, however, we went to the researchers, had them use it and asked them to give us their opinions on it. In the following section we tell...

What We've Learned

First, it is important to understand that CODA has basically been a "hobby" written on a shoestring budget. These constraints show up in the CODA code as a paucity of "gorilla proofing" and shortage of pre-release testing. They show up in testing as a lack of a rigorous hypothesis-testing methodology and a small number of "beta test sites." What we have learned about the system comes from the generous assistance of seven Rand researchers (including one of the authors [Dewar]

— from whose work the examples in this paper have been drawn) and a total of nine data bases of various types. As examples, one data base contained data on long range non-nuclear weapons (about 300 records), another contained interview data from Russian emigres (about 2400 records), a third contained data on terrorist incidents world-wide (about 1800 records), yet another contained information on reviews for the Rand Journal of Economics (about 400 records), and one contained information on current data bases in the Rand data base library (about 100 records).

Perhaps the most interesting finding for us concerned the size of the data records. The system seemed most useful on records that were small. The two largest data bases had existed previously and were already organized as "bite-sized" data records with a small number of tags. In CODA, the number of tags on these data bases grew slowly as the data were retrieved, sorted and tagged in different ways, but there was no movement toward combining records. In contrast, in one of the smaller data bases that grew with time, there was also a slow move toward splitting large records into smaller, reasonably disjoint records or compacting large records by summarizing them. Records that were a screen-full or less in size were easiest to scan and absorb quickly.

CODA, as implemented, does indeed appear to be cumbersome for large data bases (which in this case should be taken to mean 1000 records or more). Loading time for the two largest data bases on a loaded VAX 11/780 could be several minutes; looking at the glossry of tags was cumbersome (particularly on the terrorist data where practically every incident had its own unique data identifier); and hit lists tended to be long and cumbersome to wade through while sitting at the terminal. Somewhat surprisingly, however, the users with large data bases have been happiest with CODA. Our guess is that this says much more about the utility of data base and file maangement systems in general than it does about CODA in particular.

The main "choke point" in the CODA process is definitely data entry and tagging. The ability to enter data from files was the only thing that made test with the largest data bases possible, but it is a slow and painful process in general to enter data into the system and to tag it. It was something of a surprise to find that it appeared to be more satisfactory to have a secretary put several records into the systems at a time (with a tag such as "untagged") and then to add tags to them by doing full-text searches (on the entire system), than it was to tag the records one at a time.

Originally, non-data indices were a way of grouping tags FOR DISPLAY PURPOSES ONLY. They were intended to function in much the same way that the Author index in the back of a book does. We found this to be of questionable use, and found increasingly that it would be nicer to have the ability to specify at times not only the tag, but also the index with which it was associated. That capability is now implemented, and, in our ongoing tests, is being evaluated.

Finally, we arrived at some very unscientific estimates of the point at which the CODA system stops being a corroboration of one's own memory and starts to function as a long term memory for things that have been forgotten. There seem to be two kinds of threshold — one in terms of the number of data records in the system, and the other in terms of the time that has elapsed since the first record was entered into the system. According to our informal survey, it took one to two hundred records in the system before the first records had receded far enough in a researcher's memory that they appeared fresh when recalled. For data bases smaller than that, it took a few months of elapsed time before the computer's memory was clearly superior to the researcher's. These estimates do not claim to be scientific, but they do illustrate the "delay" a researcher can expect before a

system like this can be expected to begin paying noticeable dividends.

In using the prototype of the CODA system and getting feedback from others doing the same thing with different kinds of data bases, one not only learns things about the system as it is, but one also starts to get a feel for the general class of improvements that would enhance or establish its utility. It is to this set of reasonable (and not so reasonable) future system possibilities that we now turn.

Wish List

There were three general capability enhancements that occurred to us during the testing and, in decreasing order of practicability, they can be identified as 1) a bibliographic formatting capability, 2) thesaurus capability, and 3) optical data entry.

In at least two of the data bases there were a large number of direct quotes. While it is possible to put enough tags on these records to enable one to construct a bibliographic reference, there are computer programs available that will construct an appropriately formatted bibliographic reference. Data entry in these systems is of the "fill-in-the-template" form, and CODA allow one to set up user templates. While significant work would be required to build a bibliographic formatting capability for CODA, the effort would appear to be worthwhile in future systems of this type.

The second area of improvement that seemed reasonable from our test experience was to build some type of thesaurus capability into CODA. The desirability of a thesaurus that could provide "synonyms" for a given tag seems to increase as the size of the tag glossary increases. This would be a slight departure from the philosophical notion that it is best to rely on the researcher for the intelligence in this kind of man-machine system. Nonetheless, giving a CODA-like system some capability to remember or to look for similarities among tags might be a useful "advisory" capability. The details of such a thesaurus capability must remain vague at this point, but some notions of such a capability can be described. One possibility would be to "wire in" a thesaurus, in which case the researcher would be responsible for creating and maintaining the thesaurus and CODA would only respond with synonyms upon request. Another possibility would be to "teach" CODA concepts of similarity and have it constantly review the tag glossary and, on request, suggest synonyms to the user.

The most desirable improvement comes directly from numerous confrontations with the data entry bottleneck. The most obvious data entry mechanism would be an optical character reader about the size of a light pen that one could use much the way one uses a highlighting marker to mark for recall passages in a text. Entering them directly into CODA in this manner would be much more satisfactory than current data entry methods. While such a capability is an easier technological problem to solve than the more obvious voice entry capability, the feasibility of such a hand-held mechanism is still, sadly, beyond the current state-of-the-art. In fact, ANY better mechnism for entering data would measurable improve the utility of systems like CODA.

But back to reality. There are some things to be said for the utility of CODA as it is currently constituted and they are the topic of the final section of this manuscript.

Conclusions

Our original ponderings about computer-aided policy research led us down a somewhat tortuous path to the development of CODA. The prototype system was designed to test our hypothesis that computer-aided policy research could be improved, and to determine if our specific set of desiderata for such a system was a path to such improvement. CODA was built roughly to specifications implied by our desiderata, we enlisted Rand researchers to use and comment on it, and we have learned from the process.

As to whether a CODA-like system serves a useful purpose in the community of data base users, our work only leads us to suggestive conclusions. The eagerness of our volunteers to use CODA for more traditional file management purposes and for larger-than-we-envisioned data bases suggested the ever-growing recognition of the utility of computer-aided data management. This, in turn, suggests that specialized communities such as the policy research community are still awakening to the possibilities of computer-aided data management in a variety of forms.

Among the researchers who used CODA as we intended, there was a growing appreciation for and dependence on CODA's capabilities. This growing appreciation corroborates the earlier note that there is an inevitable time lag between the beginnings of a CODA data base and the appreciation of its long term memory utility.

Several of the desiderata built into CODA appeared indeed to have recognizable utility in the policy research world. The most controversial of these was the on-line ability to change tags simultaneously on large subsets of data. As mentioned earlier, in the general data management world this capability has serious drawbacks. In the community of data bases that have a small number of users, however, this becomes a very powerful tool for reforging the long term memory to conform to the current concerns and theses of the users.

While it is a very subjective judgement at best, search retrieval under one second are a definite improvement over systems with turnaround times only slightly longer. This appears to be much akin to satellite telephone conversations in which a one-second delay in conversational responses is distractingly noticeable.

The ability to enter data into CODA electronically from a file was very useful in transferring extant data bases into the CODA system. In addition, this ability led to some serious musings on the integration of CODA-like capabilities with larger data base management systems that have on-line capability for retrieval from very large data bases.

By way of improving the policy research process, the one currently feasible desideratum the CODA prototype seemed to lack was a bibliographic formatting capability. With the addition of this item to

the list of specifications, the general design goals of a useful policy research computer tool would seem to be complete.

In summary, our work with CODA leads us to believe that there is room for improvement in the area of computerized data management aids specifically designed for the policy research and related communities of users. While CODA may not be the optimal realization of that goal, the desiderata that led to its creation, along with the addition of a bibliographic formatting capability, form an excellent foundation upon which to build such a system.□