The Space Congress® Proceedings

1969 (6th) Vol. 1 Space, Technology, and Society

Apr 1st, 8:00 AM

On-Line Conversational Information Storage and Retrieval System

A. L. Scheidegger
Apollo Systems Department, General Electric Company, Cape Canaveral, Florida

AN ON-LINE CONVERSATIONAL INFORMATION STORAGE AND RETRIEVAL SYSTEM

A. L. Scheidegger
Apollo Systems Department
General Electric Company
Cape Canaveral, Florida

Abstract

During the past several years the General Electric Company has been developing a Dynamic Automated Reporting Technique (DART). DART was conceived for application to Dynamic Situations such as exist in management of large programs.

DART fulfills the following system requirements:

1) Capable of accepting inputs which vary in content, volume and structure.

2) Capable of accommodating inputs from several remote locations.

3) Ability to react to output demands which specify output content at the time of demand.

4) Function without computer programming charges.

5) Provide a means of self reporting status outputs for improving system effectiveness.

6) Capable of user operation.

7) Uses stylized English commands.

8) Capable of quick easy updating and revision.

DART has been used and modified based on experiments in Information Storage and Retrieval for the last three (3) years. These experiments have involved four (4) different computer types, and have consisted of data bases of personnel files, test planning, ECP Change Tracking and as an aid in the evaluation of Missile Guidance and Range Safety Systems performance. As a result of these experiments, further generalized needs for the overall improvement of this User Oriented Conversational Remote Access System are being studied.

This paper explains the rationale behind the DART system design and provide examples of its stylized language, data structuring and its operation.

Introduction

Communication of information from the source to the Need Point is a vital key to the successful culmination of our large modern complex programs.

It has been widely accepted that the most effective operations are predicated upon the timely availability of complete and accurate information. It has been shown that automation can and will continue to play an important part in providing that information in a timely manner.

The approach of linking an information system to a communications network provides the capability for transmitting from the source through the information system to the user whoever and wherever he may be. (Figure 1)

The operation of such a system depends strongly on the ability of the information system to communicate with the user thru the users remote terminal facilities.

The existing telephone/teletype network can and is providing remote terminal services at many locations. Improvements in remote terminals are desirable and will undoubtedly be made available.

The problem facing us now is an available user oriented, information storage and retrieval system.

Summary

This exposition has been an attempt to develop an intuitive sense of recognition of some of the problems in information storage and retrieval, and to present a philosophy and means for these problem solutions.

It is not the intent here to represent DART as the panacea to computer problem problems, however it has been used in its present form as a satisfactory solution to many of the problems of user oriented information storage and retrieval. In addition, General Electric's Apollo Systems Department is working in other areas to accomplish improvements in information storage and retrieval.

This philosophy which we have developed into a system called DART is functionally operating and has been described herein. It is hoped that this paper will be educational and informative. Further information regarding DART may be obtained by contacting Apollo Systems Department, Kennedy Operations, 7011 N. Atlantic Ave., Cape Canaveral, Florida.

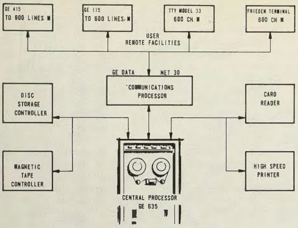


Figure 1
GE APOLLO SYSTEMS COMPUTER NETWORK

Problem Analysis

Many systems have been devised for semi-automatic and automatic storage and retrieval of information. Although most of these systems have been satisfactory, a few important ones have been either ineffective or inefficient in their missions. They have failed to support the user in maintaining cognizance over and extracting intelligence from the mass of data in the system. A review of both successful and unsuccessful systems revealed three common characteristics: (1) the input organization was specified in advance, was well defined and not changeable; (2) the data output was, likewise, specified in advance, well defined and could not easily be changed; and (3) the machine processing instructions were fixed at the data level. That is, there was no provision for data manipulation that would permit organization structure changes.

The effectiveness of these systems was a function of how well the design team could specify the inputs and the outputs to be encountered during the operational life of the system. The best systems reflected excellent design team efforts in predicting the inputs and outputs accurately and comprehensively. The important point, however, is that for a few systems even an apparently excellent system design became unusable very soon after the start of operational life, if not before.

Upon close examination, it became clear that these systems were being applied to missions which were significantly different from the missions for which they were designed. Design was based on specific programming of data relations and formats, producing systems that were not flexible. Change could be accommodated only by reprogramming, and in many missions the rate of change of requirements is many times greater than the rate at which changes can be implemented. It has been reported, for instance, that at one time the SAGE system programs were being obsoleted seven times faster than they could be rewritten.

Dynamic situations encountered in military intelligence systems, large program management, and large scale command and control systems, all present a new and challenging aspect to Information Storage and Retrieval. No longer can the usual "once and for all" approach of specifying inputs, specifying outputs, and designing the black box to perform the transformation be used. The inputs and outputs change faster than people can design or redesign the black box. The problem facing us is one of coping with a changing context. In the dynamic situation it is the context of the user which changes. Data stored in a fixed or restrictive context soon becomes uninformative and will be relevant only when reorganized in a new context. To be effective the system must be able to accommodate not only the restructuring of data but to actively aid the user in placing stored data into the new context.

The importance of context is not generally realized. Information is not derived solely from data. Data can have meaning - can be information - only when it is expressed in relation to other data organized in some pattern. Data when expressed with context is information. Context is comprised of three factors that serve to give meaning to a singular piece of data. These three factors are:

Structure: The over-all arrangement of categories into which the data may be classified and of relationships between the categories.

Interpretation: The meaning associated with each of the categories and relationships. The alternative meaning that any one category may have.

Probability: Among the alternative interpretations there will be those that are more certain than others for a given situation.

The Dynamic System

A system that is to operate continuously and effectively, in a dynamic user context, must be able to accommodate three situations that are opposite from the three situations noted for an inflexible system. Inputs will vary. Some types of data will be identified in advance of system production, but will vary in content and volume. Some data will not be identified until after system operation begins as the user context will cause changes in the structure, interpretation, and probability of alternative interpretation. There will be different relationships between new data and existing data which will continue to be used. With this changing context, outputs will change as new relationships become meaningful to the user.

To accommodate the rate of change of context evident in dynamic situations, the mechanism which is used to restructure context and to manipulate the data into the new context cannot be one of system modification. Reprogramming, the "engineering change path", will never be adequate to the response time required.

One promising solution to the problem is the dynamic system - one that provides its own restructuring capability with simple, general instructions. Three elements appear as guides for the concept:

1. An information storage and retrieval system must be an extension of the user's mind.

2. To be effective it must embody the two major properties concerned with information storage and retrieval. It must be able to store data and it must store the data within a structure that is in substantial correlation with the user's context.

3. As the user's data changes, the pertinent stored data must be easily located and changed; likewise, as the user's context changes, so must the pertinent structure of stored data be changed to mirror that context.

In the new concept any required changes are accomplished without reprogramming by holding the data in a definite structure but by making the structure independent of the machine/software configuration. The machine will hold the data and relate the data as specified by the user via the machine language; however, the language will be of such power that the structure may be changed via the same language without changing the machine/software configuration. The details of this part of the concept will be presented in the next section. It is important now to consider one serious ramification of this concept.

The Process Of Optimization

It is evident that if the user is to be able to adjust the structure of the stored data to reflect his own context changes, me must have the information criteria and tools necessary to identify the need for change and to implement the changes. It is also realized that when part of the system data file is changed, the change must be designed with great care so that other parts of the file are not affected.

To accomplish a system data file change as required to mirror user situation change and to control the system changes, it is evident that a process must be followed which is identical to the general optimization process followed in any successful development.

In the process shown in Figure 2, the developer started with a clear statement of the system mission and requirements. He then determines the state of the system and evaluates whether or not the system meets the requirements. If optimization is required, he must analyze system state to determine the cause of the inconsistency. After an analysis, he developes alternative solutions, chooses the best solution, and carefully plans implementation of the solution. Application of the plan then results in adjustment of either the system state or of the system requirements, hopefully the former.

There are several points along this process where information must be supplied and safeguards erected to insure successful adjustment of state in the proper direction. There must be an effective technique of measuring system state so that a clear, unbiased estimate of state may be obtained. Information of this kind must describe system configuration, system activity, system design parameters, etc. Evaluation of status can only be accomplished when criteria are provided to define success/failure states. Measurements of state rarely agree completely with statement of system mission and requirements. The extent of disagreement allowable depends upon many factors oriented not only towards system performance but also toward the resource expenditures required to resolve the disagreement. Thus, criteria for evaluation of success/failure must be developed before evaluation can be accomplished.

Analysis of the causes of inconsistencies requires detailed information similar to the measurement of system state. Inputs required to develop alternative solutions and to select the best solution should contain general types of solutions worked out by the system design team and specific types worked out by the user from past experience with the system. Of course, good understanding of the system design and of the various user orientations is also required at this point to preclude partial optimizing. One solution may very well improve the system in support of one user orientation while definitely degrading the system with respect to a second user orientation.

Planning the best method of implementing the chosen solution should include perspective from the system design and state, and from user administrative practices.

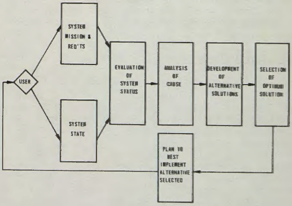

USER
SYSTEM MISSION & REQ'TS
SYSTEM STATE
EVALUATION OF SYSTEM STATUS
ANALYSIS OF CAUSE
DEVELOPMENT OF ALTERNATIVE SOLUTIONS
SELECTION OF OPTIMUM SOLUTION
PLAN TO BEST IMPLEMENT ALTERNATIVE SELECTED

Figure 2
The Process of Optimization

System Requirements

On the basis of the previous discussion, the following system requirements have been set to guide the design effort:

1. The system must be capable of accommodating inputs which vary in content, volume, frequency, and structure.

2. The system must be able to react to output demands where the content and format of the output may be specified at the time of demand. The only constraint is that the output demand must be stated in the terminology existent within the stored data.

3. No computer programming changes will be required to accomplish the system optimizing function.

4. Aids for optimizing system effectiveness shall be built-in via self-reporting status outputs and system monitor evaluation criteria.

5. System control shall be accomplished directly by the data user.

6. The system source language shall be stylized english and shall consist of a minimum number of functions.

System Design

In response to the previously defined requirements, a system has been designed and a prototype has been produced. The design depends heavily upon two techniques - the basic data organization and the source language; consequently each technique will be described in the succeeding two sections. Following this, a description of the system and examples of data and context manipulations are given.

Basic Data Organization

All data entered into the system will be in the form of a unit called an Item. Each of the many items will consist of three subdivisions, an Item Name, an Abstract, and a Text as depicted in Figure 3 (See Table 1).

The Text is the main body of the Item and consists of the various pieces of data which are closely associated with the Item Name. These pieces may be in the form of equalities and may be characterized by Attribute Names and Attribute Values or they may be statements in the form of sentences. The text may be of any length (within large bounds) and may have any format. For efficient retrieval, it will be required (where Attribute Names and Attribute Values are used) that the same Attribute Names be used in each item although they will not be required to follow identical formats.

The abstract consists of a set of Descriptors which are identifiers for the textual data. These Descriptors, chosen by the user in view of the pertinent characteristics of the data, are to establish in part the structure and interpretation of the data in the text. The Descriptors help as locators during data retrieval and provide the power to identify and retrieva classes of data.

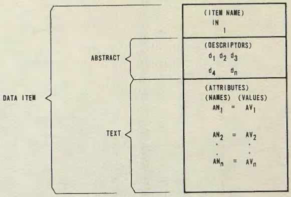


Figure 3
Basic Data Organization

The Item Name is not necessarily a unique identifier for a particular Item. Two Items may have the same Item Name even though they may not have common Descriptors, Attribute Names and Attribute Values. For example: It is completely possible to have two Items called "JOE SMITH" which have certain commonalities and differences.

Source Language

In order to provide for the versatility required of the system and to meet concurrently the requirement for minimum source language functions, a set of four functions has been found to be sufficient to direct all manipulations concerned with information storage and retrieval, and with system optimization. These functions are:

1. ADD

2. DELETE

3. CHANGE

4. DISPLAY

With these four functions the system can manipulate data selected by Item Name, specific Attribute Values, or by more general common Descriptors. The first three functions serve to enter intelligence into storage and to provide the power to optimize the system; that is, to establish a context and, subsequently to maintain the context in correlation with user context. The last function, DISPLAY, serves to retrieve data.

The system source language should allow for data organizational change as well as content change. Furthermore, it should permit retrieval as well as change of selected items of data.

The determination of Item selection should be permitted by content of an item as well as by its name. In short, change and retrieval should be permitted by implication and relation as much as possible. This requirement then allows commands to be formulated in a more associative or human fashion, rather than in an artificial or mechanical fashion where commands involve only names.

As indicated above, the system provides four basic functions with which to change data content as well as to display or retrieve data. The system language permits the user to do this through the application of a sequence of commands. A command is composed of five factors which are simultaneously interpreted to give full and widely variable meaning. These factors are:

1. Todays date
2. A <u>Function</u>
3. An <u>Object</u> of the function
4. <u>Values</u> of change or selection
5. <u>Conditions</u> of selection

Symbolically,

(command) = : (date)

(--Execute Command-------)(Under specified conditions)
(function)(object)(values)(conditions)$$

Each command is a self-contained and independent statement to the system. It results in an isolated fact about the data and does not refer to another command. However, the results of one command may be referred to by another command.
THE FUNCTION. As previously mentioned, four different functions are provided to establish and maintain context of data as well as retrieve selected data from the file. Only these four terms may appear within the function part of the command.

1. <u>DISPLAY</u> causes the selection and output of certain items or parts of items of data, depending upon the content of the remaining three parts of the command. This function also will output the contents of the descriptor pool as well as the complete data file.

2. <u>ADD</u> is the function which permits new data to be added to the file. Complete items may be added to the file or new attributes or descriptors may be added to existing items. This function allows broadening of the data context.

3. <u>DELETE</u> is the function which allows deletion of unwanted items or parts of items from the file.

4. <u>CHANGE</u> is the function which permits the substitution of a new term or value to part of an item in the place of an already existing term or value. This function permits new interpretation of the data.

THE OBJECT. The terms which appear in the (object) part of the command identify for the computer to which part of the data structure the function is to be applied. The function tells it what to do, and the object tells it on what part to operate.

The permissible terms which may appear as objects of a command with the DISPLAY function are:

1. ITEM NAME/S
2. ITEM/S
3. ABSTRACT/S
4. TEXT/S
5. ATTRIBUTE/S
6. DESCRIPTOR POOL
7. ATTRIBUTE NAME/S

The permissible terms which may appear as objects of a command with the CHANGE function are:

1. ITEM NAME/S
2. DESCRIPTOR/S
3. ATTRIBUTE VALUE/S
4. ATTRIBUTE NAME/S

The permissible terms which may appear as objects of a command with the DELETE function are:

1. ITEM/S
2. ATTRIBUTE/S
3. DESCRIPTOR/S

The permissible terms which may appear as objects of a command with the ADD function are:

1. ITEM/S
2. ATTRIBUTE/S
3. DESCRIPTOR/S

THE VALUES. The terms which appear in the VALUE part of the command amount to new data or data to be used for comparison purposes depending upon the function and object of the command. In a number of cases the content of VALUE will be empty.

THE CONDITIONS. The terms appearing in the CONDITIONS part of the command serve to define for the computer to which Items of data the command is to be applied.

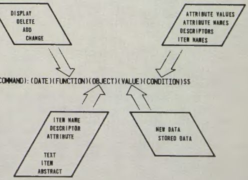


Figure 4
DART Language Summary

The conditions, therefore, allow the user to specify that data having certain common descriptors be operated on while data not having the specified commonality would be undisturbed.

The terms appearing in the CONDITIONS part of the command constitute a Boolean expression of terms. These terms may be Item Names, Descriptors, Attribute Names or Attribute Values or Text Values. It is through the interpretation of CONDITIONS that certain Items or parts of Items of data are selected from the file to be operated upon according to the FUNCTIONS. The Boolean expression which constitutes CONDITIONS is made up of one to four Boolean sub-expressions connected by logical "and".

System Functional Description

Figure 5, Overall User/Machine Activity, shows a top level layout of the system. It is composed of the user and the computer (with its Information Retrieval Language). The user acts in a dual capacity of: (1) actually using the machine as a storage and retrieval device; and (2) evaluating and optimizing the machine effectiveness. In the former role, he works with new input data and with information requests. New data is formatted into items and entered via the ADD function while requests for information are formulated as a command using the DISPLAY function. The computer responds to each command by adding data to its storage or by searching its storage for the data requested (the data which satisfied the conditions) and displaying the data at its output. The user is then free to use the data as he sees fit. Although it is hoped that at this point he will be attentive to his role of systems evaluator and express his opinion as to how well the machine satisfied his request.

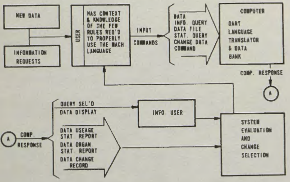


Figure 5
Overall User/Machine Activity

In his second role, the user will identify the need for context adjustment, determine the change required, and implement the change. Implementation is accomplished via the DELETE and CHANGE functions primarily, although the ADD function could also be considered in this capacity. Identification of the need for adjusting context is accomplished while evaluating system effectiveness which in turn is made possible partly by the displays of status provided by the machine. System activity reports, and configuration change records, are provided for this purpose. The additional information required to complete the evaluation - the user satisfaction - is almost automatically provided since the evaluator is the user. Selection of changes to be made should easily follow a good evaluation since the change required is that which brings the structure of data within the machine more closely in correlation with the user's context.

The machine can be any of the many computers available for data processing because there are no requirements within the new system which demand unique capabilities in the machine. Of course, the question of machine efficiency is always present and certain types of machines (disk files, serial charachter memories, etc.) have advantages over other types from this aspect of system effectiveness. Basic system functions are shown in Figure 6.

If the machine were to be examined at some time after system operation had begun, it would contain a file consisting of several items of data. The items may have abstracts of different length and content and texts of different length and content. Along with the data pool will be a descriptor pool - a list of all the descriptors used in all the abstracts. This pool becomes an important reference list for the user and is also employed in the computer for the search and compare technique of finding data.

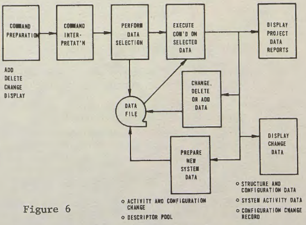

COMMAND PREPARATION
ADD
DELETE
CHANGE
DISPLAY
COMMAND INTER-PRETAT'N
PERFORM DATA SELECTION
EXECUTE COM'D ON SELECTED DATA
DISPLAY PROJECT DATA REPORTS
CHANGE, DELETE OR ADD DATA
DATA FILE
DISPLAY CHANGE DATA
PREPARE NEW SYSTEM DATA
o ACTIVITY AND CONFIGURATION CHANGE
o DESCRIPTOR POOL
o STRUCTURE AND CONFIGURATION DATA
o SYSTEM ACTIVITY DATA
o CONFIGURATION CHANGE RECORD

Figure 6
Basic System Functions

This bank of data comprises the total information used for the project upon which the system is applied, and for differentation from the following discussion this data will be termed Project Data.

In support of the user's dual role, the machine also has a dual role. There must be a second data bank which is retained as data for the system optimization and is termed System Data. Data contained in this bank must be sufficient to provide:

1. Reports upon system utilization
2. Reports upon data configuration
3. Reports on data configuration change

The first two types of data will consist of data required to manage the system. This data is geared to measurement of state and to reveal the major modes of system usage. Consequently, the data will describe daily system activity in terms of data additions such as requests satisfied and requests rejected. This activity data will also be used in the optimization effort to provide over-all profiles of system activity. As such, it will include tallies of: commands received by function type (ADD, CHANGE, DELETE, DISPLAY); number of incorrectly formulated commands; number of requests satisfactorily filled, and so forth.

The latter type of report, Configuration Change Record (System Item), will provide the information necessary to identify past changes and the change content. Configuration accounting and control is a necessary activity in any system and will be particularly required in a system in which CHANGE is a major activity. To prepare for these reports, a second set of Items (consisting of Item Name, Abstract and Text) will be set up in storage. Each configuration change command which is executed will automatically generate a new system item in which the Item Name will be the change identifier (concatenation of the contents of (function) and (object) part of the command), the Abstract will be composed of the System Descriptors signifying which data are affected by the change, and the Text shall consist of the command which caused the change and of the old values that existed prior to the change. In addition, the data of the command issuing the change will be included in the text as a separate attribute. The construction of this new Item is shown in Figure 7. These items of information are generated as part of the normal machine process and require no additional activities on the part of the user. It will be noted that change records are generated as a result of all commands which cause changes to Project Data. These commands are those that contain the function CHANGE, DELETE, or ADD since any of the three can change the configuration of the data file. The DISPLAY function, of course, only exposes the stored data and in no way changes its configuration; therefore, it is not included.

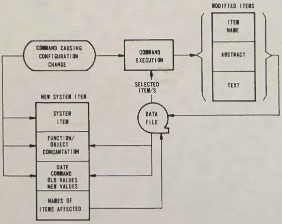


Figure 7
System Item Generation

System Model

In order to clarify many of the questions which probably have occurred by this time, it may be helpful to consider a model operation of the system. Let the system be employed as a service to a group whose interest encompasses the particulars of the top level change notification status at GE, Cape Canaveral. Their items of information would be similar to the sample in Table 1 where the Item, Name, Descriptor, and Attribute Names and Values are shown.

As the user file grows there would be stored in the computer many items similar to the sample following. Each item will have a different Item Name, will have abstracts of varying size (although many of the Descriptors from Item to Item will be the same) and will have texts of varying size. The texts will contain many common Attribute Names with differing Attribute Values. There is no requirement that the Attribute Names be ordered in the same sequence for one Item as for the next Item even though the names may be identical in both items.

```
*** ITEM NAME ***
    * 47/66ICD21Ø1

*** ABSTRACT ***
    * ORIG B
    * A HDW
    * B HDW
    * IND AREA
    * 66ICD21Ø2
    * LM 4

*** TEXT ***
    * DATE RECD AT A = 8/19/68
    * ECP NO = RC3417
    * DESCRIPTION = PROPER GROUNDING FOR THE OXYGEN SAMPLNG
 GSE
    * TWR/ECR = 9370
    * TECH APP ACT A = 9/1Ø/68
    * TECH APP ACT B = 8/16/68
    * CCB APP SCH A = UNK
    * CCB APP SCH B = UNK
    * REMARKS = RETURN FROM LEVEL II FOR CANCELLATION BEING
 HELD IN SUB PANEL    ........
```

Figure 8 - Table I

Now, suppose that a new change was initiated and that a new Item of data is available for entry on 12/13/68. The user would prepare in stylized English the data shown in Example 1 and would employ the ADD function to enter the complete item into the machine. Note that each command has a date associated with it, (generally the date of execution of the command), in this case 12/13/68.

```
(12/13/68) (ADD) (ITEM) (70/65ICD92ØØ// AS 5Ø3 * ORIG C * A DOC *

C DOC * PAD AREA * 65ICD92ØØ// DESCRIPTION = ADD WIRING BETWEEN

HYD SKID AND ANTI DRAIN BACK VALVE * DATE RECD AT A = 12/13/68 *

ECP NO = 1Ø57* TWR/ECR = 11153 * TECH APP SCH A = 12/13/68 *

TECH APP ACT C = 12/11/68 * CCB APP SCH A = 1/6/69 *

CCB APP SCH C = 1/6/69 * REMARKS = NONE ) () $$
```

Figure 9
Example I

Note that the ADD function notifies the machine which type of operation it is to perform. The ITEM entry in the object position of the command format notifies the machine that a complete item of data is to be added. The empty parentheses denote that the Conditions part of the command format is meaningless - the data is to be accepted with no regard to other stored data and the $$ denotes end of command. The Values position of the command format gives the data that is to be stored. This data, since it is a complete Item, must contain an Item Name - 7Ø/65ICD92ØØ - An Abstract (denoted by the descriptors between the next // marks) and a text (denoted similarly by the Attribute Name and Attribute Value between the next set of () marks).

The user now has an additional Item of data stored in his system. At some later time (12/15/68) let us assume that he received additional data which is pertinent to only one change 7Ø/65ICD92ØØ. In this case he would not want to add a complete item of data but would simply want to add data to an existing item in storage. To accomplish this he would again employ the ADD function, but this time would utilize the Conditions part of the command format to specify that the values were to be added only to the stored item that met the conditions stated. The command would appear as:

```
TIME OF DAY IS 17.172 ELAPSED TIME IS Ø.ØØ33
COMMAND BEING PROCESSED IS AS FOLLOWS (RDCOM)
    (12/13/68) (ADD) (ATTRIBUTE) (PRIORITY = THIS IRN HAS TP AT A) (IN(
    7Ø/65ICD92ØØ))$$ $$$$

* * * ITEM NAME * * *
    * 7Ø/65ICD92ØØ
ATTRIBUTE(S) BELOW ADDED......(ADATT)
    * PRIORITY = THIS IRN HAS TP AT A ......

CURRENT COMMAND HAS BEEN PROCESSED.
    ITEMS WERE FOUND TO SATISFY COMMAND. (MAIN)
```

Figure 10 - Example 2

The computer is thus instructed to utilize its ADD operation on the Item which is the Item Name (IN), 7Ø/65ICD92ØØ, and to add ATTRIBUTE as specified. The computer would interpret this command, search for the records containing the 7Ø/65ICD92ØØ item, and would add the specified attributes to the text. Note that this command has caused a change in the stored data configuration and that the results of this change will be recorded in the System Data portion of the system.

To further describe the model operation, assume that something in the user's context has changed and that he no longer considers the Descriptor "URGENT" to be pertinent to SC 1Ø3. He would thus desire to delete that Descriptor from all Abstracts for those Items connected with SC 1Ø3. Such a command is written as:

```
(4/8/68) (DELETE) (DESCRIPTOR) (URGENT)

    (D(SC 1Ø3))$$
```

Figure 11 - Example 3

The computer would search for all items which contained a Descriptor SC 1Ø3 and would delete the Descriptor URGENT from the identified Abstracts.

To further illustrate the manipulation power in the system, assume that it is desired to correct stored data such as changing a Descriptor from ORIG B TO ORIG C for all Change Notices which occurred between the dates 11/5/68 and 12/3/68. To accomplish this, a command using the Change function would be written. The Object portion of the command would specify that a Descriptor was to be changed while the Conditions portion would call for only those having DATE RECD AT A which occurred between the specified dates and which were relevant to SC 1Ø3.

```
TIME OF DAY IS 17.Ø57 ELAPSED TIME IS Ø.Ø17
COMMAND BEING PROCESSED IS AS FOLLOWS (RDCOM)
    (12/6/68) (CHANGE) (DESCRIPTOR) (ORIG B *TO*ORIG C)
    (D(AS 5Ø3) AV((DATE RECD AT A*BETWEEN*11/5/68, 12/3/68)))$$

* * * ITEM NAME * * *
    * 7Ø/65ICD97ØØ
DESCRIPTOR(S) CHANGED AS FOLLOWS. (CHGDES)
    * CHANGE = *FROM* ORIG B

    * TO*   ORIG C

CURRENT COMMAND HAS BEEN PROCESSED
    1 ITEMS WERE FOUND TO SATISFY COMMAND. (MAIN)
```

Figure 12 - Example 4

Note here that a double search must be accomplished because there are two conditions expressed to select the proper set of Items. First, the computer would search for those items which contained the Descriptor, AS-5Ø3, and then it would select from that group only those items received at A between 11/5/68 and 12/3/68. Since dates are not carried by this user in the Abstract, he found it necessary to search the text for the pertinent dates. Thus, he has written, *AV(DATE RECD AT A*BETWEEN* 11/5/68, 12/3/68) as a constraint which tells the computer to search for an Attribute Value, AV, which has an Attribute Name of DATE RECD AT A, and to select those items for which the value lies within the bounds of 11/5/68 and 12/3/68.

The previous examples illustrate how the input and storage processes and the manipulative power of the system are utilized. The following two examples will illustrate the tremendous retrieval power of the system.

Suppose that one user interested in the 65ICD92Ø2 desired a history of these changes. He would know that 65ICD92Ø2 was used for either of two locations - Pad or Industrial Area - and that the purpose would be keyed in the Abstract as Pad Area or Industrial Area respectively. Since the logic operation OR is not included in the source language and since there may be changes with Pad Areas only, changes with Industrial Area only and changes with both, the user must write two commands to obtain all the data.

```
(4/7/68) (DISPLAY) (ITEM) ( ) (D(PAD AREA*65ICD92Ø2))$$
(4/7/68) (DISPLAY) (ITEM) ( ) (D(IND AREA*65ICD92Ø2))$$
```

Figure 13 - Example 5

Thus, his need for information has been comprehensively satisfied.

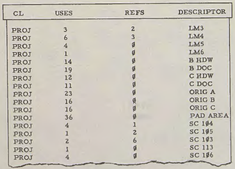
Assume that there is a second user who wants to interrogate the same bank of data from a second viewpoint. Suppose that he is concerned with evaluating the performance on a change implementation cycle of certain Change Notices. He might want to see the complete Item for all Instrumentation changes received at A before a certain date during 1968 and received technical approval at A before November 1, 1968. The resulting command would be:

```
TIME OF DAY IS 17.161 ELAPSED TIME IS Ø.ØØØ7
COMMAND BEING PROCESSED IS AS FOLLOWS (RDCOM)
    (12/13/68) (DISPLAY) (ITEM) ( ) (AV((TECH APP ACT A*LESS THAN*
    11/1/68)*(DATE RECD AT A*BETWEEN*1/1/68, 11/1/68))D(INSTR))$$
```

Figure 14 - Example 6

Thus, it is seen that the system meets the requirements set forth for the system design and that the user has significant power to use effectively large amounts of Project Data. He is not isolated from the machine by programmers or system operators, and yet is not required that he learn a voluminous set of rules and a strange language in order to use the machine himself.

With respect to System Data, the operation is oriented towards keeping the user advised of the actual contents and configuration of his data and towards providing him with measurements of system activity and direction of change. In the previous examples it may have been noted that the user will sometimes choose to search Abstracts for key Descriptors, while at other times he may specify text searches for Attribute Names or character strings (Text Values). A question probably arose in the reader's mind as to how the user is to remember which words are used as descriptors and which words appear only in the text. Earlier, in the System Functional Description, brief mention was made of the Descriptor Pool (Figure 15) which is a list of all Descriptors used in the stored Abstracts. This list is used internally in the machine as a search reference when commands are being interpreted and it is also displayed to provide the user with a reference document. With this document, therefore, he can determine if he is not sure, whether or not the words he considers key words are included in the Descriptors. If he finds that they are, he can confidently command a search of Abstracts on the basis of his Descriptors. If, on the other hand he finds that his key words are not included in the Descriptors, he may command a search by other associated Descriptors or by Attribute Names. He may also, at this point, use the ADD function to add his key words to the pertinent Abstracts if he feels that his key words will have continuing significance as Descriptors. This latter action is representative of the user identifying a need for optimizing the system and taking the appropriate action by adjusting the context of the stored data.



CL	USES	REFS	DESCRIPTOR
PROJ	3	2	LM3
PROJ	6	3	LM4
PROJ	4	Ø	LM5
PROJ	1	Ø	LM6
PROJ	14	Ø	B HDW
PROJ	19	Ø	B DOC
PROJ	12	Ø	C HDW
PROJ	11	Ø	C DOC
PROJ	23	Ø	ORIG A
PROJ	16	Ø	ORIG B
PROJ	16	Ø	ORIG C
PROJ	36	Ø	PAD AREA
PROJ	4	1	SC 1Ø4
PROJ	1	2	SC 1Ø5
PROJ	2	6	SC 1Ø3
PROJ	1	Ø	SC 113
PROJ	4	Ø	SC 1Ø6

Figure 15
Descriptor Pool

Configuration change accounting is accomplished by automatic generation of System Data Items as explained previously. In the System Model examples it will be noted that commands, except Display, caused the actual configuration of the stored data to be changed. Therefore for each of those examples here would have been a new item of data generated for configuration change accounting purposes. In Example 2, the ADD function was employed, and although a change resulted, there was no change to existing data as happens with the DELETE and CHANGE functions. Therefore, the item generated will consist of the Item Name, the date of command execution, the content of the command and the Item Name - 7Ø/65ICD92ØØ - of the affected Project Data Items.

```
*** ITEM NAME ***
    * SYSTEM ITEM

*** ABSTRACT ***
    * ATTRIBUTE ADDED

*** TEST ***
    * DATE = 12/13/68
    * COMMAND = (12/13/68) (ADD) (ATTRIBUTE) (PRIORITY=THIS
      IRN HAS TP AT A) (IN(7Ø/65ICD92ØØ))$$
    * ITEM = FILE DESCRIPTION
    * PRIORITY = THIS IRN HAS TP AT A
    * ITEM = 7Ø/65ICD92ØØ
    * PRIORITY = THIS IRN HAS TP AT A......
ITEM BELOW DELETED ...(DELITM)

    * INTERFACES = NONE
    * SYSTEM IMPACT = ADD CAPABILITY FOR FOUR MORE TRACKS ON
FR1ØØ TAPE RECORDER
```

Figure 16
System Item Generated From Example 2

In Example 3, where a general class of items were changed (all those having SC 1Ø3 in their Abstracts), there will possibly be several items changed with no evidence to the user as to which were changed. A simultaneously generated item of System Data would account for all changes; that is, its Text would contain the command wording and the Item Names of the Project Data Items which were changed by the command.

```
*** ITEM NAME ***
    * SYSTEM ITEM

*** ABSTRACT ***
    * DESCRIPTOR DELETED

*** TEXT ***
    * DATE    =    12/14/68
    * COMMAND       =   (12/14/68)(DELETE)(DESCRIPTOR)(URGENT)(D(S
   C 1Ø3))$$ $
    * ITEM     =    FILE DESCRIPTION
    * DESCRIPTOR DELETED      =      URGENT
    * ITEM      =    1Ø/661CD21Ø5
    * DESCRIPTOR DELETED      =      URGENT    .......
ITEM BELOW DELETED ....(DELITM)
```

Figure 17
System Item Generated From Example 3

Recent Efforts

In late summer of 1967, using the General Electric, Apollo Systems Department's computer capabilities in Daytona Beach, the Dynamic Automated Reporting Technique (DART) was installed on the IBM 7044. Sample data were loaded, edited, revised and retrieved. The preliminary investigation on this project indicated that the DART system was technically capable of timely and efficient transfer of management data which can be applied in a general sense to many management problems.

In October of 1967, the installation of DART on a Remote Access Batch GE 635 computer was initiated. The complete conversion of the program to the GE 635 was completed in early January 1968. Since that time, the concept of applying DART to some of the management systems problems which we face has been attempted, and has been in many ways quite successful. After a thorough analysis of certain typical systems, it was found that major problems are associated with gathering data, and getting data into a readily usable data bank. Solution of the problem of inputting data under the one described here, should enhance the operation of any system considerably. With the advent of remote access on a large scale computer, it was found that DART could be made even more cost efficient on the GE 635 than before. This is extremely encouraging to the people working with DART and who promoted the remote access capability. Thus, the program was launched to implement in a pilot mode the use of DART as a Change Statusing System. The primary purpose of the system was to create easy and quick access to status on certain changes, which are being implemented by General Electric. The remote teletype access allows free use of data entry from any teletype location; allows entry of data in the user context; and provides an input system which eliminates load forms, key punch, subsequent transmission to a computer. It also eliminates the need for voluminous reports and provides management direct and quick access to the data bank.

Turn-around on the GE 635 computer was initially estimated to be feasible in a matter of four (4) hours. Experience as of this writing has shown each case to be within this estimate. In many cases, responses were available within 5 or 10 minutes, some, before the operator of the teletype has completed his entire activity. The attractiveness of being able to structure his own context the data which he was about to enter, has shown a very appealing side. The interest and flexibility of operation which is provided by DART to the user has turned the idea of loading data into an understandable, more fascinating and challenging task. In addition, specific individuals have been called out as being responsible for certain data within their area. This individual is responsible by name, if the data is not current or needs to be updated, and has not been updated, his name is made available for direct telephone contact if required. He is also immanently aware of a direct responsibility, where prior to this time, no direct responsibility had been assigned to him. The need for this direct responsibility is felt, if he doesn't do it, no one else will. He knows that since he has direct access, there is no filtering, and exactly the data he inputs will be there, rather than some condensed coded version which might come out as a result of clearing house activities. Although the cost of operating DART is somewhat higher than operating a similar conventional system, we must recognize the cost of maintenance of a clearing house of data has been eliminated. This is one of the tradeoffs to be made, and usually results in lower overall costs. There is no question that the user of the system itself can be made more efficient and less costly by the use of devices, such as coding and abbreviating etc. However, it is not our intent at the moment to regress into this type of operation which would eliminate one of its main attractions of being simple to understand and operate.

Future Directions For Optimization

Many extensions, optimizations, spin-offs and applications of the present work have been identified at this time. They range from making the design more flexible and powerful to defining machine requirements for implementation of time sharing. Some of the more important directions are:

1. IMPROVING MACHINE EFFICIENCY
   Although actual machine efficiency has been considered in the system design phase, it has been regulated to a tertiary objective. There are undoubtedly improvements to be made in this area and any design review should be conducted from this viewpoint.

2. MORE FLEXIBILITY IN DESCRIPTOR USE
   An immediate step in this direction would be the introduction of an OR capability for selection of Descriptor level data.

   The addition of Multi-File addressing capability through the use of Descriptors to create a working file capable of being addressed by DART commands. This need will become more apparent as the size and volume and numbers of data file increases.

3. INCREASED LANGUAGE CAPABILITY
   (a) By the use of synonym tables to allow greater freedom in selection of terms.

   (b) By tolerating somewhat less than 100% fulfillment of the spelling requirement, say, 7 or 8 out of 10 matches within the file structure.

4. COMPUTATION AND MORE FLEXIBLE FORMAT CAPABILITY
   By allowing user controlled report generator flexibility and internal computer calculating flexibility to allow matrix arrays to be printed and operated on.

5. INCREASED SYSTEM CAPABILITY
   The capability to provide an Attribute Name List for purposes of system activities.

The next planned activity in furthering DART development will be to concentrate on task 4 above, namely to extend its manipulating power to perform generalized mathematical tasks and print columnar arrays of numerical and alphabetic data in order to provide concise small volume summaries to the large volume user.