1688

C/67-3

ADMINS - A Progress Report

by

David Griffel and Stuart McIntosh

Center for International Studies

Massachusetts Institute of Technology

January 1967

Preface

This document is intended to serve as an interim recording of the work the authors are currently doing, and also to serve as a back up user manual to the user documentation built into the ADMINS system. The work is being done under the auspices of the Center for International Studies here at M.I.T. and is supported by an N.S.F.* grant for which project Professor Ithiel Pool is the principal investigator, and is aided by a technical committee comprised of faculty associates Professor James Beshers and Professor Joseph Weizenbaum. Naturally, all of the mistakes the authors have made are their own, and of course, without the facilities of Project MAC the work would not have been possible.

*N.S.F. Grant - Computer approaches for handling large social science data files.

CONTENTS LIST

3.f. Location statements

3.g. Audit statements

3.h. Descriptive statements

3.i. Transformation statements

3.j. Contingency statements

3.k. Diagnostics

3.l. Organizer usage

3.m.1. Processor overview

3.m.2. Processor usage

3.m.3. Processor instructions

3.n.1. Processor output

3.n.2. Marginals

3.n.4. Error report

3.o. File inversion

3.p. Data alteration

3.q. Concatenating files

4 Analyzer

4.a. Analyzer - overview

4.b. Analyzer instruction types

4.c. File focus

4.d. Classified directory maintenance

4.e. Set operations

4.f. Summarization

4.g. Recoding

4.h. Multi-level files

4.i. Syntax documentation

4.j. Analyzer - usage

4.k. Analyzer - error comments

4.l. Analyzer - storage management

4.m. Data manipulation language

5 Dynamic loops

5.a. Organizer - processor loop

5.b. Admins binary input

5.c. Vertical horizontal processing

5.d. Multi-source files

5.e. Multi-level files

ADMINS

Administrative Data Methods For Information Naming Systems

1.a.1. Data about the environment is gathered and recorded. This is usually done according to some purpose. If many different types of record are generated, it becomes necessary to keep a record (meta data - bibliographic data) of the data records.

1.a.2. Some other institution may acquire from many different original generaters, their data records and even perhaps their meta data records. The problem confronting this institution will, dependent on its collection oriented purpose, be that of organizing the variety of data records it has now acquired by means of its own meta data record (which may or may not rest on the originators' meta data records).

1.a.3. This institution has a further problem. Any organization of the record must be capable of re-organization to support a user oriented purpose which need only be stated in general but must be supported in any particular case.

1.a.4. As records are always embodied in some media which has a constraining effect we are concerned to design procedures using a flexible media (a disk oriented, time shared computer facility) which are effective on data records and meta data records where we have no control over the input conventions, i.e. we did not gather the gathered data, our selection is restrained to that which we choose to acquire.

1.a.5. Administrative Data Methods are quite well understood in the areas of business management and public administration. Information naming methods are quite well known in the areas of Librarianship and Technical Documentation. What is not self evident is how to design an information system that does both, embodied in a third generation computer facility that will be used by socialscientists who have a dynamic view of data.

1. b As methods by which one derives a norm from data are perhaps more difficult to make completely public than are methods for classifying data under a norm, the public and explicit statements about data are ordinarily of the type that this data has been coded and categorized under that norm. Thus the systemic aspects of an information handling design is the activity that makes explicit what it means by data, e.g., actual or bibliographic. What are data entries, data categories, data items, data files. What are the ordering relations of each of these elements. What data ought to flow where under what norms. What are the functional sub systems, what different kinds of data flow do they handle. How do they relate to the overall system, and how does the information handling system interface with its environment?

1. c This abstraction called a system may be said to be workable when it can adapt to the vagaries of the data it has been designed to handle and also, when it can accomodate to and eventually assimilate the changing purposes of its users. Thus the system must not only be able to process data under its norms, it must be such that one can change its norms based on data experience. If the system is to be designed to accomodate one particular application problem, it is conceivable that the fitting together of a congerie of components on a trial and error basis will provide an adequate solution. When the application is a data management system to support user applications in areas as yet unspecified and with substantive data as yet unknown, the practical approach is to proceed on a trial and error basis only within a framework where the system relationships are made public and explicit at all levels of system function. This will enable the input from one sub system to take the output of another sub system without the aid of prayer, chewing gum and string. Even so, one cannot design an adequate data management system without some experience in adapting to many substantive problem descriptions. Nor can one accomodate a flexible data management system to a new application, unless there is an adequate substantive problem description of this new application.

1. d We are of course only concerned with systems for administering information but even so this can be quite complex. There are different types of data, different embodiments of the data, different ways of naming the data. The data can be what we will call actual data, i.e., data on the substantive and methodological aspects of the environment that the system has been designed to collect data about. The actual data can be information data where interpretations of this data are essentially linguistic, vis a vis what the data is pointing at in the environment, and ambiguities lie in the province of the philosophy of language. The actual data can also be scientific data where interpretation of this data are essentially scientific vis a vis theory, and ambiguities lie in the province of the philosophy of science. The data, however, may not be actual data but bibliographic data, i.e., data which describes the physical embodiment of the actual data record and in broadly logical terms describes the form and content of the actual data record. Interpretations of bibliographic data are essentially meta-linguistic, i.e., we have a language for pointing at actual data which is also in a language. There are of course plenty of naming confusions possible. The name 'occupation' can be the name in a catalog describing the content of a codebook prototype norm. The same name may be a category of a codebook describing a data record of actual information or scientific data. A data management system has to avoid both the following options. It can address itself to trivial clerical tasks and give a user little intellectual assist. It can address itself to intellectual tasks but the results of its operations are so esoteric that only members of the cult can live with the constraints.

1. e Actual data and bibliographic data can both be heavily structured, i.e., the norm under which the data is to be processed can be designed such that the possible entries under a category can be made explicit and the relationships of a category within an item can also be made explicit. Finally, these items, e.g., a catalog item describing say a survey code book or e.g., a questionnaire report item describing information data about a person, can also be made explicit as to the ordering relations between the items in a file. Thus the codebook prototype norm that purports to control the processing of actual information data in empirical questionnaire reports or of actual analytic scientific data in social science index construction is the same as the catalog prototype norm that purports to control the processing of bibliographic data about codebook prototype norms. That is to say is the same for a considerable amount of data processing activities because they can both be heavily structured.

1. f When it is not possible to say explicitly what entries are legal under what category, what are the precise relationships between categories, what are the categories that define the boundaries of an item, what is the threaded pattern of items in a file--we say that we have loosely structured data. Both for real data and for bibliographic data the structure may only be a little loose, i.e., some of the entries under some of the categories are open, i.e., cannot be explicitly closed in some effective consistent way. However, with feedback--post editing in an adaptive system for processing the data records--one can get on top of some of these exigencies. The linguistic problems associated with wholly loosely structured data are beyond our competence; however, we provide some clerical help for handling this kind of data.

1. g When one has different types of data in various states of process from different files, generating reports of one kind or other, one has to develop an administrative model that can organize the contingencies via a process catalog of some complexity. In effect we have backed off from the philosophy of language problems inherent in loosely structured data in favor of the philosophy of science problems inherent in the organizing and searching heavily structured data that may be used for policy and research purposes. The macro organization and search capability operates through the prototype norms. A particular norm is that organization that controls a particular body of data. The method is the procedures for processing the data under the norm which may be viewed as a data sieve. When the norm and data are not in correspondence, control audits over their discrepancies must be available so that one can either edit the norm or edit the data. It is also possible that one may have errors in the syntax of the procedure descriptions; it is therefore necessary to have control audits over these errors which must then be edited. One might also be accessing the wrong data and therefore control audits over these errors must also be available. In general, a visible sampling capability must support the audit-edit machinery. As well as control over the data process application one must have access also to controls over data format, data embodiment and data channels provided by the underlying computer (program management) systems structure.

1. h The function of designing a norm which is intended to sieve information data or bibliographic data is not fulfilled by ADMINS in so far as there are no executable procedures for norm designing. However, there is provided clerical procedures for manipulating the descriptive information used in designing a norm. The function of operationalizing a theoretical hypothesis such that correspondence may be sought with scientific data is similarly not fulfilled executably by ADMINS. However, there is provided executable procedures for manipulating the symbolic tokens which represent the hypothesis after the clerical procedures have been used to describe the information as scientific data. Indeed after one chunk of scientific data has been so handled another chunk may be processed by executable procedures. Thus the construction of a scientific data norm may get more of an executable boost from ADMINS than an information data norm, although the intellectual planning work in both cases is done by induction machines called people. However, by offering them a comprehensive clerical assist, they get the most out of this augmentation by endeavoring to make public their private ways of working which they normally do not think about when concentrating on substantive or formal issues.

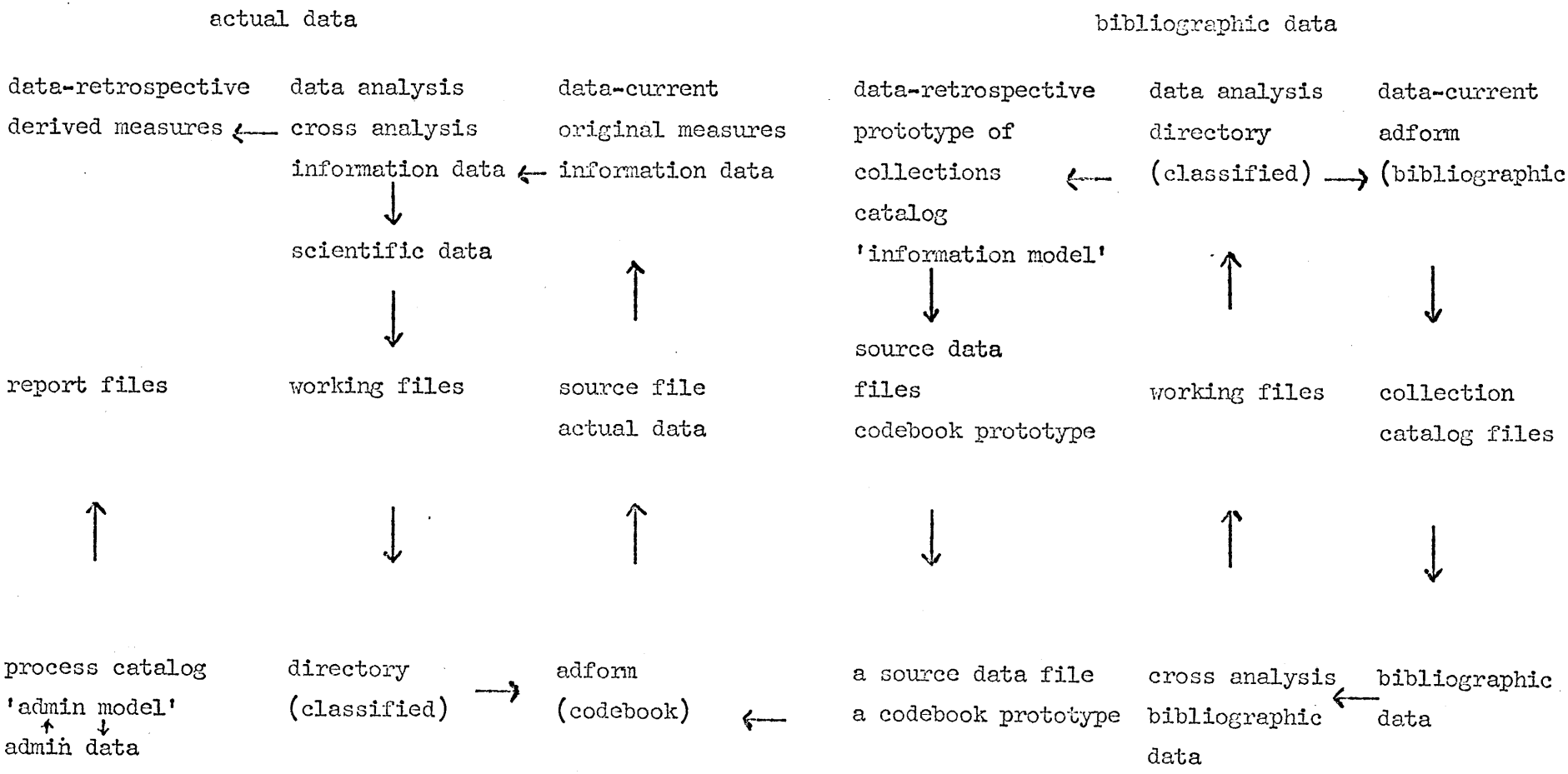
1. i Once a particular norm has been explicitly stated, if we have been flexible in our system design, we can expect the system to help us amend the norm. The function of an organizer sub system is to make it possible to get a norm into an active state so that it can be applied to the data. The function of a processor sub system is to have controlled processing of the data under the norm. The function of an analysis sub system is to accomodate to the analysis purpose of the user of the data, and to assimilate these purposes in the development of more powerful data analysis capability. The analysis sub system must also provide an interface to statistical analysis sub systems developed by scientific users. The result of the analysis of information data will be derived measures about the information system environment. The result of the analysis of bibliographic data will be derived measures about the information system collection and usage.

1. j Taking a general look from the top ADMINS can be used in the following way as outlined in Chart A. The heavily structured bibliographic data, i.e., a catalog of codebook prototype norms is analyzed. The result from analyzing this 'information model' is the selection and subsequent retrieval of codebooks and corresponding data files for some particular use purpose. The pertinent actual information data is then processed under the codebook prototype norm and analyzed according to the user scientific purpose. The processing of data under norms continues through several stages, at the end of a particular phase the data and norm is in a certain state. Reports, current and cumulative, will have been generated which describe the results of the activity, and on the basis of these reports, decisions are made, as to what to do about a certain state of data or norm. The macro organization of data source files, report files and of norm files, by means of a process catalog which will permit flexible search strategies is the administrative function of ADMINS. The purpose of ADMINS as an 'administrative model' is to provide the data management capability of accessing retrospective bibliographic data via the 'information model' and then of providing selected actual data for current processing as required by user purpose. Basically ADMINS can support three main types of user application. The cross analysis of the characteristics of bibliographic data. The cross analysis of the characteristics of information data, which can then be operationalized to a cross analysis of the characteristics of scientific data, which may then be used to support the building and testing of a social science model. The cross analysis of the characteristics of information data can also be used in the support of policy formulation. For example, the characteristics in a personnel file are original measures as to role and person which may be cross analyzed and the derived measures used in support of staffing policies.

Chart A
1.J.1

ADMINS

MACRO-ORGANIZATION



1. k ADMINS was conceived as a data management system using a computer system as an administrative tool. This means that substantive decisions can be made on an immediacy basis without being bogged down with clerical chores. Thus in order to make intelligent use of interactive checkout and update, the data processing must be such that control interrupt decision making is really decision making of an intellectual nature. Clerical decisions that one cannot routinize must be swept up on the run when substantive decisions are being made; otherwise machine aided cognition is a farce and resources are being wasted. This limits the use of the ADMINS system to users who have considerable substantive knowledge of what they want to analyze, and are sufficiently dedicated to learn to make decisions as they go rather than by searching for a needle in a haystack of tables. Admittedly this is a tough confrontation, it will, however, have to be faced sometime so it may as well be now. This agony can be made somewhat less exasperating if the designer can provide a consistent language in which the user may describe and instruct the various data processing tasks. The grammar of the organizer, processor and analyzer are all quite different to each other, however both within and across these sub systems we have made an effort to be consistent in language conventions. Furthermore we have provided on line documentation not only of 'command' syntax but of 'command' explanations.

1. 1 As the systems analysis developed and the efforts to abstract general purpose functional sub systems that could handle different types of data became easier to specify explicitly it became apparent that some parts of the task were tougher than others. The simulation of sub file construction both within and across files by means of indexing the data so that one may compare the relevant chunks to each other, further analyze within a chunk, re-group according to a theory or policy, was one such task. Another was the development of an ability to name indexes by means of token symbols and/or by names such that one had a mechanism by means of which one may begin a classification of data. An actual data (information or scientific) index structure may be very complex. The co-ordination of tokens (names or symbols), the relationship of tokens as roles or links, the inclusive hierarchical relation of tokens provides the ability to classify data indexes which can then be searched and selected by means of these tokens. This classification of an index structure, which is in effect a classified index to the partial contents of one data file, may be abstracted from a particular index structure and then applied to another file, given that it is possible to recognize and erect a similar index structure, i.e., if one could do it by hand, the classification will be able to do it also. The relevant classification code conventions could either be completely public or it could be an explicit but private user convention. The public classification technique could be applied to heavily structured bibliographic data provided the categories in the catalog items are sufficiently detailed to make the exercise interesting. However, in what way one can categorize in a heavily structured catalog the content of a variety of surveys such that the noise does not overpower the coherency of the categories in the classification syntax is a problem we have yet to tackle, along with the problem of classification of topic subject descriptions.

1. m The general considerations so far expressed usually serve two purposes in any particular study. They get the administrative systems analysis into the right area of discourse and hopefully they force the substantive problem specifications out into the, public and explicit, open. However, the form of the system design must eventually be specified in a more detailed, coherent and integrated way.

2. a The design of an administrative form whose function is to be a vehicle for expressing prototype norm specifications under which heavily structured bibliographic data and actual data, both information and scientific may be processed, requires detailed consideration. Basic data elements are possible entries under a category. The tokens used for representation are numerical or alphabetic codes sometimes even alphanumeric codes. Possible interpretations of these codes vary. Sometimes the numerical code represents a nominal code, sometimes a numerical value. The tokens may also be used in a classificatory way under a particular category. Within a category it must be possible to specify associations between the possible entries and also contingencies between them. It must also be possible to change the order to the possible entries and to combine as required. Where the entries are numerical values to perform on them numerical computation and also as required to interval the numerical values as discrete numerical codes. One must also be able to specify audits on the legality of actual entries against possible entries both before and after the required transformations. The subject description of the possible entries and of the category under which they fall must be stateable and any changes to codes reflected in changes to subject description. The format of the data must be specified unambiguously as this is a matter of interpretation as well as of existence. Finally the category must be named as well as described. In summary then, we have statements in an object language about subject descriptions of data and token codes for the data. We also have statements in a meta language describing the data relationships and transformations, and ought/is relations between norm and data. These meta language statements are named. For each category the object and meta statement are set out as sentences in a paragraph and the syntax is kept as consistent as possible. Some of the statements are executable and some only descriptive. Some statements may refer to another category.

2. b The categories, as named, may also exhibit associative and contingent relationships which must be specified. There must also be a capability of specifying audits between possible entries in different categories. The description feature must allow for descriptions that reflect a topic covering several categories and any comments as required. The administrative form may be composed of several categories for an original item. The adform must be so named such that another adform composed from the same original item may easily be related to it at analysis of categories stage. The specification of an administrative form when syntactically complete may be viewed as a computer program in a problem oriented language which serves as input to another computer program called the Organizer. This program is a translator or application compiler which outputs a file of categories specification, a file of subject descriptions, a format specifications file, a resequence table file, and a recording and audit program, subject of course to syntax error discrepancies which are reported upon at compilation stage. This may be achieved in a diagnostic mode before a real run is made.

2. c The function of the Processor sub system is that of a control program over the coding transformation process and audit control process, as applied to the empirical data. The discrepancies between the adform norm and empirical data are here checked out. A pass may be made through the Processor sub system gathering information on discrepancies without actually applying the required transformations to the data. An item can be sampled giving the details of the transformations and discrepancies for each entry as occurs in each category. A number of items can be processed and the type of discrepancy as occurs shown for the relevant category in the relevant item. A control may be set on the number of errors to be allowed before processing is to stop at the next item. Control may similarly be set as to the number of errors allowed in a category. Control results may be specified in an actual error versus control error table. It is also possible to pass through the system in silence mode, generating no descriptive information about errors, only the number of errors, which may be done in verify mode every 100 items. Naturally one will eventually pass through the system actually changing the data and get an item output file containing the relevant categories and changed entries. One may also obtain a report file containing a summary by category of error types and number of errors. Also obtained is a file of aggregates of data entries from which a file of marginals, i.e., the aggregates with their subject descriptions may be procured.

There is also the capability of processing a data file in sections such that one may append subsequent sections to previous sections. This permits two different kinds of flexibility. Some source files are continually having new items added to them, as for example a catalog of bibliographic data. Some source files are quite large, thus it would be convenient to process the file in perhaps some random or skipped way and later perhaps process another section of the file. In both cases one would like to be able to treat the appended files as one file for analysis purposes.

2. d When a user has selected all or some of the categories in a codebook prototype norm, e.g., questions in a survey codebook, e.g., combinations of responses from a variety of questions as a social science category, e.g., categories describing bibliographic data states, e.g., categories describing personnel characteristics; and these categories have been assembled in an administrative form under which the data has been processed, the result is a file of processed items in correspondence with the norm. When this processed item file is inverted we now have a file where the ordering relation is by category where previously it was by item. Each category record contains the normative description that defines the category and its legal entries plus and subject description of both plus the actual entries as have occurred in the empirical data. In essence each category record is a contents list of the possible characteristics and a file of the occurring characteristics for each category. A category record file is a file of the category records that have been chosen perhaps at different times from the original codebook for the original source file.

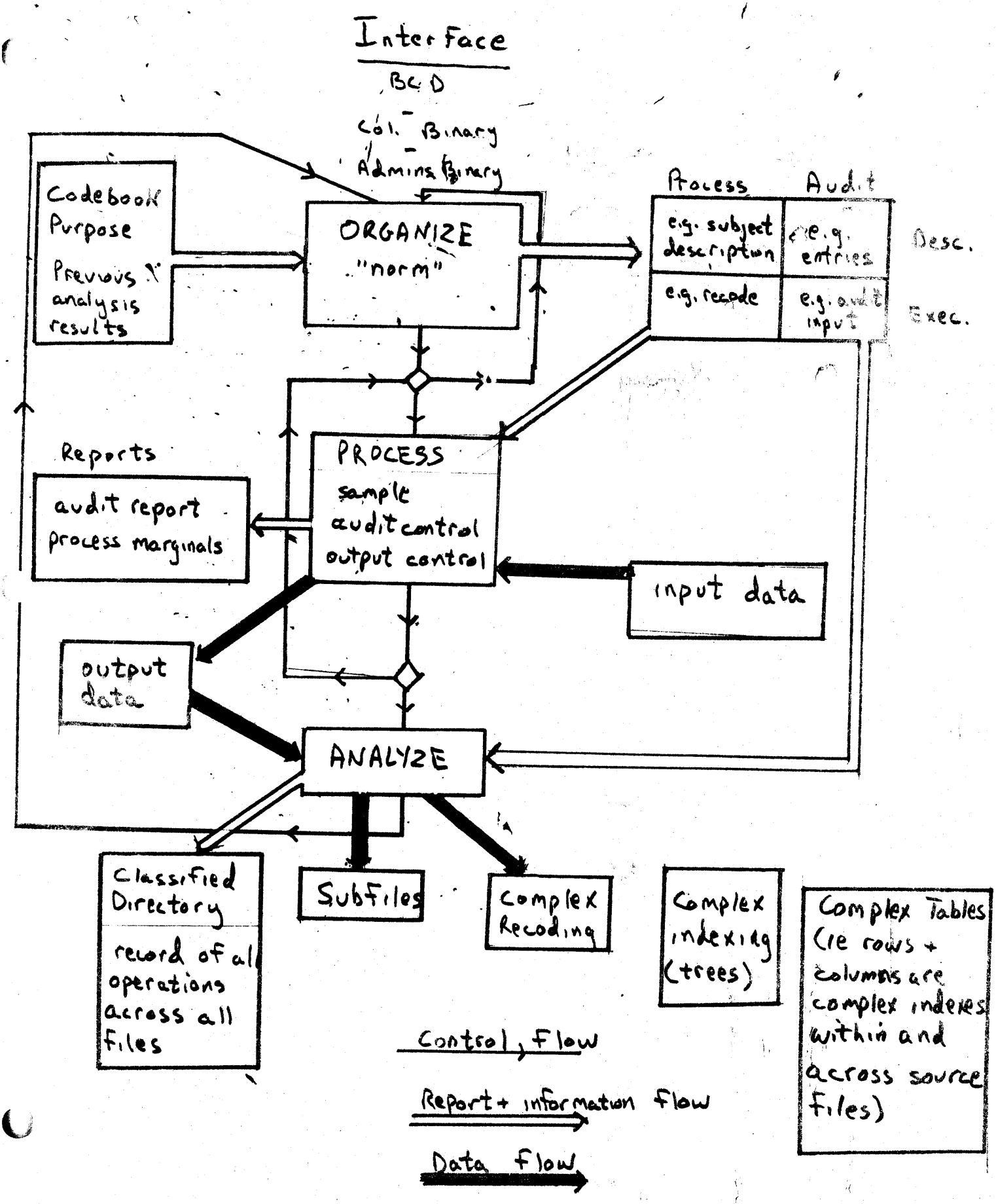
2. e The simplest example of a category record is for example when the contents list describes the possible responses to a survey question and the record contains a file of the actual responses of the population interviewed. A more complex example would be when the contents list describes a grouping of possible responses according to some social science category and the record also contains a file of the actual responses of the people who responded to these selectively grouped possible responses. The second type of category record would have been constructed from certain operations upon the first and contingent upon the results of these operations. An in between kind of construction of a category record would be that when a new category record constructed from the first type does not contain all of the information in the second type. For example, when the contents list is of the second type but the record does not contain the file of actual grouped responses but only contains the number of people who so responded.

2. f An index, in the simplest case, is an index to a particular entry in a category record. It is a list of the locations of the actual occurrences of, e.g., a particular response to a question as filed in the category record. More complex indexes may be constructed by, for example, unioning responses to a question. The actual entries in a category may be intersected with the actual entries from another category thus forming another type of complex index. In this case we would for example have a list of the actual occurrences within the same item of two different responses as filed in two category records drawn from the same source file population. Similarly one may union entries from different categories. With these tools one may build up complex information indexes which point to the occurrence of characteristics in combination from different category records about the same source file. For people who think in terms of trees, each node of the tree is an index and a path from one node to the next node is operation of constructing another level of indexing with another entry from another category. Unions are in effect the combining together of paths.

2. g Decisions as to what characteristics ought to be combined in an information index are made on the basis of what named information one requires in combination. As in the case, for example, with bibliographic data when one wants information about surveys, e.g., the country, the type of panels, the time, the size, the codebook location, the data location. However, the number of items that one obtains at a particular intersection in the case of surveys is further examined in more detail for descriptive information. If, however, the items are records about an individual in a personnel file this further examination would also follow, but in the case of a social science survey, the scientist wishes to invoke statistical tests upon the numbers of individuals with selected characteristics so that he may decide in what way he invokes further combinations of these information characteristics. These complex information indexes he is building according to a purpose such that at some point he can assert that they are in correspondence with a social science concept and thus name them as a social science index. In effect for the social scientist, information index construction is not only a search tool where interactive checkout of comparisons of summarized data appropriately described is essential to path one's way down complex trees (if so conceived) but is an analytic tool in the sense that statistical tests may be called to allow interactive decisions to be made in support of substantive knowledge.

2. h As these complex information indexes are built with reference to attitude characteristics, performance characteristics, existence characteristics, and each one is perhaps equivalent to a simple social science index, the table way of thinking may then be helpful. Several associated complex indexes may form the columns of a table and several others the rows of the table. The social scientist may be investigating some socio-psychological concept vis a vis performance in certain subject areas. He may wish to compare several socio-psychological concepts against the same subject area performance and existence criteria within one source file population. If he has several source files which are both in form and in concept compatible he may wish to run his analysis in parallel for the different populations. For example the same type panel for different time periods for one country. Similar panels for different countries at the same time. Comparisons for respondents who repeated over time. The selection of similar occupational groups from panels of a different nature for comparison of attitude and performance at roughly similar times in different countries. As discussed in the previous paragraph, information indexes may be built up, and several different files worked in parallel. The ability to work in parallel allows the investigator to depart from fundamentalist assumptions or clerically fossilized activity and massage the given information for each file into an indexing strategy that is compatible for comparing across files, meanwhile using a rational naming convention which helps keep track of the complexities involved in manipulating symbolic tokens which are open to different interpretations. In order to make this feasible one has to have a directory of the index structure that is a directory of the names or symbols used in labelling the structure. One must also be able to list the contents of the directory by file worked then index or by index then file worked. One must also be able to list the details of the index construction.

Chart B
2.M.1.


Interface
BCD
Col. Binary
Admins Binary
Codebook Purpose
Previous analysis results
ORGANIZE "norm"
Process
Audit
e.g. subject description
e.g. entries
Desc.
e.g. recode
e.g. audit input
Exec.
Reports
audit report
process marginals
PROCESS
sample
audit control
output control
input data
output data
ANALYZE
Classified Directory
record of all operations across all files
SubFiles
Complex Recoding
Complex indexing (trees)
Complex Tables (ie rows + columns are complex indexes within and across source files)
Control Flow
Report + information flow
Data flow

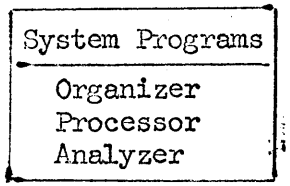
2. 1 When the investigator is combining and recombining various characteristics in his index construction he is in effect simulating recoding which he may actually effect later on if he composes another administrative form embodying new combinations. Also, if a consistent convention is adopted for the naming of indexes either with names and/or symbols, with respect to one processed source file, the recoding of another processed source file may be simulated by application of the names in the directory structure of source file A data to source file B data. The co-ordination of word forms as index terms within a name, where the index terms may be specified as role or link facets is a naming organization which may be searched for co-ordinations or for hierarchies of word forms. The information indexed thus named by a co-ordination of word form index terms may have been constructed from simpler indexes named one to one by the component word form index terms. In all probability this will not often be so, thus a mechanism for selecting names of indexes from which a more complex index was constructed, is desirable. So also is the converse ability to select the name of a complex index given the names of the simpler indexes from which it was constructed. The renaming of the names of indexes as names and/or symbols is required as is the different requirement, i.e., the renaming of indexes. When only one source file is being worked both of these renamings have the same effect, however, when two or more source files are being worked renaming of an index affects only the index concerned whereas renaming of a name affects all of the indexes thus named. Conceptually the naming of data indexes and conventions for classifying the names (and therefore the data indexes) is similar to the naming by index term word forms which describe subject descriptions (loosely structured data). Once index term word forms have been chosen (by computer programs and post editing) as naming elements for the description of subject descriptions, the machinery for manipulating the naming classification of subject descriptions can be the same as the machinery for manipulating the naming classification of information indexes (and social science indexes) for actual data and heavily structured bibliographic data.

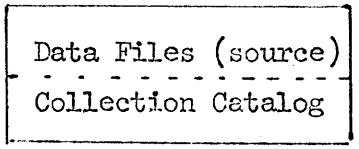
2. 3 The data management task remains essentially the same when a category record contains only aggregates of an entry, although the scientific statistical tests may be of a different kind. In the simplest case the aggregate may be the population of a town, the sample size of a survey. For demographic or ecological data the original measures are of this kind, when derived measures are the result of some computation on the original measures. The aggregates themselves may have been produced by data processing operations within the system or an aggregated data file may have been entered into the system as a source data file. When these aggregates are to be selected as control contingencies over the analysis of category records containing the entry characteristics of a number of individuals, the problem becomes one of devising a category record that contains as entries the aggregates for each of the groups within the category. For example, a category record for towns, for example, a category record for survey populations. Once constructed these category records may be analyzed using the normal indexing machinery.

2. k The flow of the instructions and user control via feedback loops is outlined in Chart B. The top level feedback is between analytic and organization-processing. Here the hunting and comparing analysis (perhaps perceived as pathing a way through trees) constructs complex indexes as rows and columns for tables. These tables are convenient artifacts for book-keeping parallel index construction within and across files by means of the directory of classified names for the index structure. This simulation of recoding by index construction may be consolidated in a new administrative form which is organized and processed against the previously processed data, and the result of the subsequent processing inverted to category records for analysis of these more cogent data categories. An intermediate level feedback loop is between organizing and processing. The discrepancies between norm and empirical data have to be settled by amending the norm or the data and re-processing until correspondence is satisfactory to the user. The micro-level feedback loops and continuous interactive checkout is mainly evident in two ways. Syntax error messages, especially in the organization of the adform; and data processing and data analysis interim results checkout as information upon which to make a decision as to how to direct the flow of control.

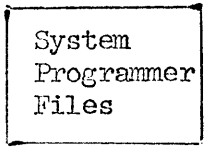
2. 1 The flow of interaction and administrative control via feedback loops is outlined in Chart C. The top level macro-organization feedback is via the collection catalog (also used by users) and the process catalog which has to keep track of all files in process and all states of each file for each phase of the process. An administrative form can be at different states of organization, there can be several administrative forms for one source file. Adforms for different source files may be siblings of each other. The organization of an adform results in a family of report files and report files from the processing of data under this adform join the family. The source data may be processed in separate sections which have to be appended. The processed items are inverted and the category record file may be composed from different processed files from the same source file. The directory and tables from analysis of category files in parallel from different source files are also report files. Working files may be saved. The function of a process catalog is that of a retrospective record of the results of operations, such that the relevant files may retrospectively be selected and retrieved. The collection catalog has the same function for unprocessed material. A system remark file allows for user suggestions and a monitoring of command usage so that further options may be developed. On line documentation is part of each program and gives details of correct command syntax and explains the function of each command. The retrospective selections and retireval capability for data, source and processed, and for codebooks, source and processed will be augmented by a more micro level feedback control in the form of a modest ability to access small chunks of loosely structured data such as subject descriptions to codebooks. The accessing capability in conjunction with accessing of heavily structured categories of file description both source and processed for empirical and normative data represents our present endeavor in computer based data management.

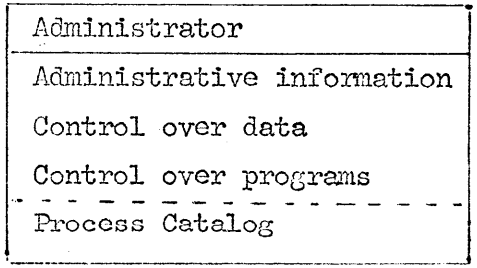
Chart C

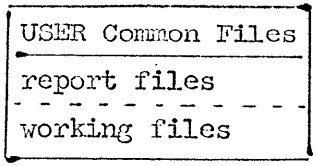


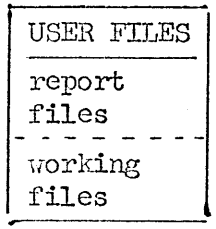


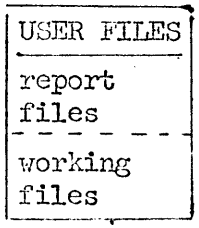
Program Maintenance











2. m As Admins is a development system resting on top of another development system, the CTSS of Project MAC, the obviously one seeks ground rules for developing an interactive data management system. Monitoring usage is helpful but not very meaty. Procedural research, i.e., the activity of regarding any particular substantive analysis from the frame of reference of the massaging of data management procedures to a better user fit within an interactive checkout philosophy, is the only really useful way to proceed. This in turn means that users who insist on using a machine aid cognition resource, which is limited, only with production of substantive results in mind, the fast turn around of batch production of tables mentality ought to be discouraged. One can do this by system design, administrative action and social pressure and perhaps better, by all three.

3.a.1 Physically, data input to Admins consists of a linear sequence (file) of items. Each item contains positionally designated headings under which codes are found. We call these headings categories and the codes entries. A category may be of the following types:

1. Nominal - Each entry records the presence of some nominal characteristic of the item. These nominal codes may be either numeric or alphabetic.

2. Ordinal - As with nominal entries, ordinal entries record some characteristic of the item, however, the alphanumeric values of the entries code an ordered relationship linking the characteristics the entries represent.

3. Interval - The entry numerically measures some characteristic of the item.

3.a.2 The physical representation of the entry codes may be one of three types.

1. Binary Coded Decimal - The entries are 6-bit codes which may be used to represent the integers, alphabetic and most punctuation characters. Sequences of BCD codes may have several interpretations, such as various length integers, alphabetic codes, etc. Most commonly BCD codes are found on IBM (Hollerith) punch cards or on tape 'images' of these cards. Clearly, BCD codes only cover a few of the possible punch permutations in a 12 hole column. This fact leads us to 2.

3.a.3 2. Column Binary - The entries are represented as holes in columns of a punch card image which exists on tape or disk; each column is imaged in a 12-bit pattern. (Column binary card images are commonly used to hold binary translations produced by Fortran-like compilers.)

Much data prepared before the late 1950's, that is under the influence of the counter-sorter, etc. attempted to code maximum information onto the punch card thereby creating patterns of holes in a column which had no BCD equivalent. As a result if such cards are read by standard input/output packages (i.e., designed to accomodate BCD) various forms of confusion ensue, none of a constructive nature. To summarize, BCD uses 6-bit codes to represent the contents of a column on a Hollerith card. Column binary 'images' a column as a 12-bit pattern.

3.a.4 3. Admins binary - The data output of Admins, a form of binary coding which individually packs each category based on the data descriptions of the category on the adform, may be used as input to Admins as well. The various uses of feeding system output back in as input are discussed further on.

3.b.1 Admins can be conceived of as a static collection of integrated functional sub systems, each embodied in computer programs, each with its own administrative languages, each with its own report generation and diagnostic abilities. Alternatively, the system can be understood as a dynamic flow of information and decisions within and across sub systems under control of the user, 'conversing' with and reacting to Admins.

In order to fully understand the latter some knowledge of the former is required. Therefore, we will first attempt a static description of the sub systems followed by a discussion of the possible administrative and man-machine feedback loops the program design is able to accomodate.

[1] CTSS Manual AH.3.07

3.c.1 The input to the Organizer sub system is an administrative form (adform). The adform contains process and audit information for each category the user wishes to access in the data file. That is, the adform describes a prototype (as well as transformation for producing this prototype) of the item record (output) file which the Processor should produce from the input data file.

The process and audit statements on the adform are either executable or descriptive. The following chart gives examples of each.

3.c.2

	Process	Audit
Descriptive	Examples: No. of entries in category, an English subject description for each entry and for the category.	Examples: The maximum permissable number of entries in a category, likewise for the minimum.
Instruction (Executable)	Examples: A recode statement, a re-sequence statement, the location of the input codes on the item record.	Examples: An executable statement which declares the 'legal' input code configuration, likewise for the output code configuration.

3.c.3 The output from the Organizer is, in the case of a 'valid organization', a group of disk files containing tables, a ring structure, and a computer program, all of which can be though of as a computer understandable rendition of the adform.

Alternatively, the Organizer may have found errors in the adform either of a purely syntactic or of a coherence nature, in which case the output from the Organizer are descriptions of the errors. The user is expected to correct these errors and re-submit his adform for organization.

3.c.4 The adform is prepared, edited and re-edited using the EDL command of CTSS, which allows an alphanumeric file to be flexibly typed onto the disk and contextually altered.[1]

3.d.1 The adform is a sequence of paragraphs, each corresponding to a category in the data. Each paragraph is broken down into a sequence of sentences. Each sentence contains a statement in a language which is understandable to the Organizer. The function of the repetoire of statements made available to the composer of the adform (the user) is to provide a language sufficiently rich for describing all necessary data audit and data process procedures dictated by the users purpose and the state of the input data file.

A statement consists of a statement identifier, followed by an '=', followed by a string of alphanumeric symbols conforming to the syntax of that statement, followed by a '.'. Blank characters are of no consequence anywhere on the adform. This permits the user any 'form' design he wishes in laying out the adform within the constraints that statements terminate with periods, paragraphs with double periods (..), the adform with a triple period (...); plus constraints imposed by the syntax of the individual statements.

3.d.2 Statements are of the following types.

1. Location - This instructs the Processor as it is filling the input positions where to locate and how to interpret the input codes.
2. Audit - These state a norm against which the data will be compared. Discrepancies between norm and data will be reported by the Processor sub system.
3. Descriptive - These describe the form and content of the category.
4. Transformation - These transform the contents of the input positions into output entries.
5. Contingency - These assert the data contingent flow of computer control thrpugh the adform during processing.

These statement types are not mutually exclusive, that is particular statements may be of more than one type. (For example, the re-sequence statements perform both audit and transformational functions.)

3.d.3 The Processor sub system (under interactive control) will be responsible for reading the input items, using the location statements on the adform to fill the input positions, applying the executable statements on the adform to these positions, placing the output entries from each category into item records, and writing the item record file onto the disk. As processing occurs (and cumulatively as well) audit discrepancies are reported.

3.e.0 Before delineating the individual statements of the organizer language let us describe certain syntactic conventions common across many of the statements.

3.e.1 Positions can be thought of in two ways; as containers holding alphanumeric values or as codes which are present or absent. Each interpretation is convenient for certain types of input data. Containers are referenced by prefixing the position number with a V, creating a simple arithmetic expression. For example, 'V3' is read 'the value in position 3'. The alternate interpretation of positions is as a boolean or logical expression which is either true (if the entry is present) or false (if the entry is absent) and is stated by prefixing the position number with a B, creating a boolean expression. For example, 'B3' is read 'boolean 3'.

3.e.2 Constants are numeric values, such as 7, 10, etc., or alphanumeric in which case they are enclosed in '$', such as a, 77, $$. Constants are simple arithmetic expressions.

3.e.3 Arithmetic operators are used to link arithmetic expressions to form arithmetic expressions. Arithmetic operators are + (plus), - (minus), * (multiplication), and / (division).

3.e.4 Relational operators are used to link arithmetic expressions to form boolean expressions. The relational operators are: L (less than), E (equal to), G (greater than).

3.e.5 Logical operators are used to link boolean expressions to form boolean expressions. The logical operators are: A (and), O (or), X (exclusive or), N (not), T (then).

3.e.6 Precedence operators - Parentheses are used to express precedence of interpretation when evaluating expressions.

3.e.7 Subroutines may be invoked by preceding the subroutine name with 'SB' for a boolean subroutine, 'SV' for an arithmetic subroutine, and following the name with arguments enclosed in parenthesis and separated by commas.

3.e.8 (As the statements using the syntax conventions thus far described are easily translatable into MAD statements, which is what is done to them, further explanations of their meaning can be elicited from the beginning chapters of the MAD manual.)*

3.e.9 Through Notation - certain statements require the assertion of sequences of constants. '+' (read 'and') is used to separate constants which are noncontiguous (i.e., do not differ by 1) and '-' (read 'thru') is used to assert a contiguous sequence of constants.

Notation	Interpretation
1-3	1,2,3
1-5 + 7+6	1,2,3,4,5,7,6
1 - 4	1,2,3,4 left-justified alphanumeric
7-4+2	7,6,5,4,2,

3.e.10 Syntax Convention Chart

Positions	V'n' or B'n'
Constants	numeric or alphanumeric (enclosed in $)
Arithmetic operators	+ - * /
Relational operators	L E G
Logical operators	A O X N T
Subroutines	Boolean and arithmetic
Through Notation	+ - numeric and alphanumeric
Precedence	( )

* MAD Manual - Galler Arden Graham, 1963.

What follows is a description of all the statements in the Organizer language. Following each statement in parentheses is its abbreviation, which may appear on the adform in its place.

Location

3.f.0 FORMAT (FMT) - There are three format statements; for BCD, for column binary, for Admins binary.

3.f.1 BCD - This states the card number (within the input item), the column on the card, and a Fortran format statement of form XYZ where X is the number of input positions to be filled, Y the interpretation (I for integer, A for alphanumeric) and Z the size (in columns) of the field; card, column, format are separated by commas.

3.f.2 Column Binary - Card number, column number, punch number. The punches specifiable are as follows on the table.

Punch number	Hole
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	0
11	X (Zone)
12	Y (Zone)
13	Blank (i.e., not 1-12)

Punches may be referenced using the through notation. Each punch goes into an input position. The card and column number are separated by a comma; the punch numbers are enclosed in parentheses. An entire column may be read as one input position containing the 12 bit pattern by not specifying any punches. More than one column may be referenced in one FORMAT statement, by repeating the pattern of card, column, punches.

Two subroutines, COLINT and COLBCD are available for interpreting full columns read into single positions as integers or alphanumeric respectively.

3.f.3 Admins Binary - the category name followed by the entries required. As in column binary entries are enclosed in parentheses, may be specified using the through notation and more than one category may be referenced.

Numerical (i.e., interval or bibliographic) categories are referenced by placing an 'N' between the parentheses.

3.f.4 In all three varieties of the FORMAT statement the user is not restricted to the original ordering in the input item, e.g., the BCD FORMAT statements in an adform may skip back and forth across the input cards and columns. As well multiple references are made to the same input fields by prefixing in the FORMAT statement a previous category name--whose input positions one wishes to reference--with a '/'. This allows one to reference the same input positions as the category 'slashed'.

Audit

3.g.1 AUDIT INPUT (AI) - The user is asserting--by stating a boolean expression in terms of the input positions--his expectation as to the state of the data in the input positions. The boolean expression is evaluated using the data in the input positions, for each item. If the boolean expression is false the data in the input positions for that item is discrepant; such occurrences are reported during processing.

3.g.2 AUDIT OUTPUT (AO) - Similar to the Audit Input statement except the boolean expression is stated in terms of the output entries.

3.g.3 AUDIT ITEM RECORD (AITR) - The boolean expression is stated in terms of categories and their entries appearing in the adform. This allows an output audit contingent on data in several different categories.

3.h.1 [redacted] NAME (N) - This is used to assign each category a 6 or fewer alphanumeric character name. All references to the category are made with this name. NAME is always the first statement in each paragraph, as all diagnostics are described with reference to a category name.

3.h.2 SUBJECT DESCRIPTION (D) - Each category and each entry within it receive an English subject description--72 alphanumeric characters or fewer. All multiple blanks in the description are squeezed to single blanks. The entry descriptions are separated by slashes. Numeric categories are only given category descriptions.

3.h.3 ENTRIES (E) - This states the number of possible entries, the maximum permissable in an item and the minimum permissable. These three numbers are separated by commas. If the category is numeric, that is its entries are numeric values, then one writes an 'N' followed by a maximum value for the output entry.

3.h.4 ASSOCIATION LOCAL (ASL) - By using a pattern of 1's and 0's the ASL statement asserts associations among the entries.

3.h.5 ASSOCIATION GLOBAL (ASG) - Same as ASL except the association is among the categories.

3.h.6 SCALE (S) - 'ON' specifies the entries are ordinal.

Transformation

3.i.1 RESEQUENCE CODES (RSC) - Input positions (interpreted as values) may be resequenced into output entries by specifying a resequence table using the through notation. The table is interpreted to mean: if the nth value of the table is present in any of the input positions, the nth output entry should be produced.

As RSC implicitly audits the data (i.e., an audit discrepancy occurs if an input position contains a value not in the resequence table) one may follow the resequence table with a permission table--separated from the resequence table by 'permit'--which contains values (again using the through notation) one does not want represented in the output entries but nonetheless are permissable input.

3.i.2 RESEQUENCE POSITIONS (RSP) - The same as RSC except one specifies positions and not their values. An alert reader will notice that the flexibility of the column binary or Admins binary varieties of FORMAT statement allows one to resequence in that statement. Since the absence of any transformation statement in a paragraph causes input to become output, one can resequence positions without a transformation statement. However, reasons of style, double resequencing of complex input, and the fact that one cannot 'permit' on the FORMAT statement make RSP useful.

3.i.3 RECODE (RC) - One states a sequence of IF's, each of which are followed by a boolean expression. Each 'IF' is evaluated producing the nth output entry if the nth 'IF' was true.

3.i.4 ARITHMETIC (ARITH) - If the output entry is a numerical value computed by an arithmetic expression in terms of the input positions, the arithmetic expression is here stated.

3.i.5 INTERVAL (INT) - This is used if the input positions contain numeric values and one wishes to produce output entries corresponding to intervals of these values. The intervals are specified by writing the boundaries separated by a '-'; the intervals are separated by commas.

3.j.1 FILTER (F) - This contains boolean expressions followed by destination paragraphs specified by their category names. Each boolean expression (stated in terms of the output entries) is evaluated. If found true control is routed to the specified destination, otherwise the next boolean expression is evaluated, and so on. If all boolean expressions in the FILTER statement are false, or if there is no FILTER statement, control continues at the next paragraph. As in RECODE the boolean expressions are preceded by IF. They are followed by GOTO and a category name. Note: FILTER affects control flow during processing <u>not</u> during organization.

3.k.0 The Organizer is programmed to recognize and clearly describe quite a variety of possible syntax or coherence errors. Let us see a few examples.

3.k.1 Syntax

A statement in category 'xxx' does not have a terminal period.

BCD Format - category 'xxx' uses column outside item.

Image Format - category 'xxx' exceeds 12 punches plus blank.

BCD Format - category 'xxx' overlays previous format.

Literal - category 'xxx' has unbalanced dollar signs.

Category 'xxx' - item record audit references non-existent entry of category 'xxx'.

Interval for category 'xxx' is incomplete.

3.k.2 Coherence

The following statements must always refer to the same number of output entries: SUBJECT DESCRIPTION, ENTRIES, RESEQUENCE POSITIONS/CODES, RECODE, OUTPUT, AUDIT. They are checked against each other and if incoherent the Organizer prints a message followed by a table specifying for each statement involved the number of entries referenced.

As these examples are but a small part of the programmed syntax and coherence checks the user can feel secure that an adform organized without error expresses at least clerically, his full intent.

3.k.0 The Organizer is programmed to recognize and clearly describe quite a variety of possible syntax or coherence errors. Let us see a few examples.

3.k.1 Syntax

A statement in category 'xxx' does not have a terminal period.

BCD Format - category 'xxx' uses column outside item.

Image Format - category 'xxx' exceeds 12 punches plus blank.

BCD Format - category 'xxx' overlays previous format.

Literal - category 'xxx' has unbalanced dollar signs.

Category 'xxx' - item record audit references non-existent entry of category 'xxx'.

Interval for category 'xxx' is incomplete.

3.k.2 Coherence

The following statements must always refer to the same number of output entries: SUBJECT DESCRIPTION, ENTRIES, RESEQUENCE POSITIONS/CODES, RECODE, OUTPUT, AUDIT. They are checked against each other and if incoherent the Organizer prints a message followed by a table specifying for each statement involved the number of entries referenced.

As these examples are but a small part of the programmed syntax and coherence checks the user can feel secure that an adform organized without error expresses at least clerically, his full intent.

3.1.1 When the user has finished preparing this adform with EDL, it exists on the disk as a disk file of name ADFORM 'basic-label' where basic-label is a 6 or fewer character name which the user has selected to identify all report and process files associated with this particular adform. The user may then give the ORGANIZE command with arguments specifying:

1. The basic label of the adform.
2. The mode of input - BCD, Column Binary, Admins Binary.
3. The size of an input item; in case of Admins binary this is irrelevant and the name of the source file of the input category records is specified instead.
4. An optional argument 'diag' which asserts to the Organizer not to produce any of the intermediary files but only to scan the adform for syntax and coherence errors. This is done whether 'diag' is specified or not, but in the latter case a valid organization produces intermediary files.

If the ORGANIZE command is given free of arguments, the console prints back the proper argument syntax. (This is so for every command in Admins; the term 'command' is used to refer to programs that exist on the disk loaded, awaiting execution, i.e., core images. The term 'instruction' is used to describe 'verbs' understandable to core images.)

3.1.2 If the Organizer finds an error in the adform, it will continue the scan of the adform seeking further errors. However, certain errors dis-orient the program, in which case the Organizer stops.

3.1.3 When the user has achieved a valid organization his disk files hold a compiled MAD program, named 'Basic-Label BSS' and 4 intermediary files containing:

1. A ring embodying the structure of his adforms.
2. A table containing the English subject descriptions.
3. A resequence table.
4. A format table.

These disk files are used by the rest of Admins as the data progresses thru the system. Their existence need not concern the user.

3.m.1 The Organizer never saw the users data. It is the function of the Processor to apply the 'organized' adform to the data under interactive control.

The function of the Process command is to load up the following into a 'process image'.

1. The BSS program produced by the organizer.
2. The appropriate interface subroutine (BCD, Column Binary, Admins binary).
3. The control program which shall apply the organized prototype to the data and produce online reports and disk summary files.
4. A subroutine 'BIBLIO BSS' which is called after each input item record is read into core with a pointer to the input buffer and is expected to return a value representing one of the following messages:
   a. The control program should proceed.
   b. The control program should halt because this item is out of sort, or for some other reason. BIBLIO may print a message if it wishes.
   c. This item should be skipped. This permit samples, random or otherwise, of the input file to be taken.

If a disk file of name BIBLIO BSS is found in the users file directory it is loaded into the process image; otherwise a BIBLIO BSS which always returns the first (i.e. the proceed) request is used.

.m.2 The user provides the following information as arguments when he invokes the PROCESS command:

1. The basic label of the adform which he wrote and now wishes to apply to his data. (A link must exist in the users directory of name 'INPTO Basic-label' which points to the input data file.)

2. The mode of the input data file - BCD, Column Binary, Admins Binary.

3. Optionally, that he is producing an output data file which he wished to 'append' to an already existing output data file.

4. Optionally, that he wishes to save his 'process image'. If this option is not taken the 'process image' is placed into core for execution.

The instructions the user may type at the console to his process image are as follows:

3.m.3 DO 'n' or 'all' - which instructs the processor to process 'n' input items or all of them. As audit errors are found a one line message of the following form is printed on the console. Item number/Category name/reason for message/contents of input positions/ output entries.

The possible reasons for the message and their mnemonic codes are:

Mnemonic	Reason
SAMP	the user requested a sample.
IAUD	input audit discrepancy.
RSP	resequence position discrepancy.
RSC	resequence code discrepancy.
PRIM	user supplied primitive flagged an error (i.e. returned 'false' when called).
MAX	user specified maximum for number of entries was exceeded
MIN	same for minimum.
OAUD	output audit discrepancy.
AITR	audit item record discrepancy.
INT	interval discrepancy, i.e. a value which fell out of all the intervals was input.
ARTH	The numerical value exceeds the maximum specified by the entries statement.

If the user has 'turned on' the silence feature all such messages are suppressed.

Unless a relevant control interrupt is brought to bear, the process image will process the instructed amount of items and print a summary line telling the total items processed thus far, and the total errors discovered. If the user has instructed the processor to do 'all', processing continues until an 'end of file' is encountered on the input data file.

3.m.4 SAMPLE - This command processes one item printing the contents of the input positions and the output entries for each category processed. In effect this produces a 'slow motion film' of the application of the adform to one item. Sample may be used at any time during processing.

3.m.5 STOP - This instruction terminates processing.

3.m.6 CONTROL 'n' - This tells the process image to control interrupt after 'n' errors are found in the input file. At a control interrupt the summary line is printed on the console and the user is free to give any command he wishes.

3.m.7 SET - This instruction puts the user in a mode where he types a category name followed by 'n' a tolerance setting; he may do so for as many categories as he wishes. A control interrupt will occur if 'n' errors in the specified category are exceeded.

3.m.8 SETOFF - This turns all individual settings off.

3.m.9 DECONTROL - This turns the interrupt features (CONTROL, SET) off.

3.m.10 SPECIFY - This prints out a small table showing for each category in which an error has occurred, the number of errors and the current tolerance setting.

3.m.11 SILENCE - This turns off the printing of error messages feature.

3.m.12 PRINT - This turns on the printing of error message feature. Initially this feature is on.

3.m.13 DUMMY - This instructs the process image not to produce an output file, that is the user is processing the file in order to generate the reports and not to produce data output. Obviously, this introduces an economy in computer time usage.

3.m.14 DATA - This instructs the process image to produce an output data file. This feature is initially on.

3.m.15 SKIP 'n' - This causes the processor to SKIP 'n' input items.

3.m.16 VERIFY 'on' 'off' - If this feature is 'on' the processor prints the one line summary message every 100th item record processed.

3.m.17 These instructions are typed on a line. The 'process image' responds to them, either by processing data (printing messages as it process), or by printing some information, or by turning some 'switch', and then prints back 'ok' whereupon the user can type another instruction.

3.n.1 The output of the 'process image' is:

1. A binary disk file of aggregates.
2. A binary disk file of errors.
3. Optionally, the output data file in Admins binary form which exists on disk file 'item basic-label' or, if appending, on 'append basic-label'.

Upon completion of processing there are two printed reports the user can obtain.

3.n.2 (1) Marginals - this consisted of a printout which contains for each category and each entry in the category:

1. The number of items with that entry (category).
2. 1 as a percentage of the total number of items.
3. 1 as a percentage of the number of items having that category.
4. The english subject description.

3.n.3 This report is composed from the binary aggregate file and the subject description table by typing the MARGINAL command followed by the basic-label. The report is placed on the disk file 'MRGNLS Basic-label' and may be printed on the console using the 'PRINT' command in CTSS* or by requesting it be printed offline on the 1401 using the 'RQUEST' command in CTSS.**

3.n.4 (2) Error Report - whereas the error messages printed out by the process image were one-to-one in relation to the error causing the message, the user may generate an error report which summarizes the errors with respect to the cause of the error; the form of the error report is as follows:

*CTSS Manual AH.5.03

**CTSS Manual AH.6.06

For each category the number of errors are specified. Under each category heading, lines of the following form appear. Error type; contents of input positions; number of errors of this type with these contents; serial numbers of the first 3 items in which the error occurred.

The report is composed from the binary errors file by typing the REPORT command followed by the basic-label. The report is written on disk file 'REPORT basic-label' and may be printed by the same means described for printing the marginals.

3.o.1 File Inversion

The data output from the process sub-system is a linear sequence of item records each containing entries under the categories specified in the adform (the prototype of the item record). This data is not usable by any further part of Admins until this item record file has been inverted, that is, until for each category required, a category record is produced. A category record is a disk file containing the entries under a particular category for all items, the english subject description of the category and its entries, the aggregates of the category and its entries as well as other descriptive information from the adform. The disk file name of a category record is 'CATG FILNAM' where CATG is the name of the category (assigned with the NAME instruction in the adform) and FILNAM is a user assigned name of the source file from which the category record was produced.

3.o.2 A file is inverted using the INVERT command, which requires the following information from the user.

1. The basic-label of the item record file.
2. The name of the source file.
3. Optionally the names of the categories to be inverted. If none are specified all categories from 'ADFORM basic-label' are inverted.

The invert command takes as input:

1. An item record file.
2. The ring structure produced by the organizer
3. The subject description table.
4. The binary aggregates file.

These four are integrated to produce the category records requested.

Data Alteration

Decisions made from the reports of the process image may require a few alterations in the data (i.e. the entries) of certain categories for certain items. This is done using the ALTER command. The command is typed followed by a category record name and the source file name. The ALTER command obeys instructions which can print or change the entries of items in the category record specified. The ALTER program automatically redistributes the aggregates to reflect the changes to individual items.

3.q.1 Appending Files

The APPEND command concatenates the item record file 'ITEM basic-label' and the item record file 'APPEND basic-label'. Both must have been produced by the same process image. As well, the binary aggregate files from both are 'added' together to reflect the concatenated file. The APPEND command is typed followed by the basic label of the appropriate 'item' and 'append' item record files.

The Analyzer

4.a.1 The Analyzer is the key sub system inasmuch as the user has organized and processed his data in order to analyze it. Generally speaking, data analysis consists of a dynamic interaction between classification and summarization according to a theory (or set of hypotheses). As the Analyzer is a tool, a perspective on the Analyzer's capabilities can be gained in a discussion of the tools required to classify and summarize heavily structured data.

Classification, clerically, is a process of combining characteristics and naming these combinations according to a plan. If the classification is to be empirical, i.e. is responsive to the empirically observed items which fall under the classification, the data must be summarized in terms of the classification, and the resulting summarization fed back into the plan. The classification is built up as the a priori (theoretical) notions interact (in the mind of the user) with empirical summarizations of the data.

As the Analyzer is the users clerical aid in this process of empirical classification, it must possess tools for combining characteristics, applying classificatory names to combinations of characteristics, and summarizing the data in the framework of the classification.

Characteristics are combined by building indexes to the items in the file. Indexes are lists of token pointers to items. The criteria for building an index are stated in terms of the existence of a category and/or (combinations of) entry(ies) in the item record file.

Indexes, once built, may be conceived of as sets of items. The basic set theoretic operations may be applied to these sets producing complex indexes. In turn, the set operations may be applied to complex indexes, ad infinitum.

Names may be given to indexes as they are built. The naming mechanisms allow both symbolic as well as mnemonic english names; these mechanisms can interpret names as expressing hierarchical or facet relationships. The names and explicit descriptions of the indexes they label are organized in a 'classified directory'. As indexes are constructable both within and across source files, the classified directory is a very complex structure, as it is responsible for the recording of arbitrarily complex index construction and classification.

Summarization involves tabulating the empirical data in the framework of the classification embodied in the classified directory. As data and user purposes are varied there is a large repetoire of summarization procedures in the Analyzer. One is the ability to build a table, whose rows and columns are complex indexes existing in different source files, and whose cells are the sizes of the intersection of the row and column co-ordinate indexes.

The Analyzer is a highly interactive program. The user types an instruction and is immediately presented with its effective result. Almost every instruction the user issues states a decision he has made as a result of information presented by the Analyzer in response to a previous instruction.

4.b.0 The instructions available in the Analyzer fall into the following types:

4.b.1 1. Maintenance of the classified directory. As the user proceeds in his analysis, he is empirically combining characteristics of the items of the data and giving these empirical combinations names. The sequence of his set operations and the names he gives are stored in a classified directory.

4.b.2 2. Maintenance of the 'file focus' - As the analyzer crosses files the user needs a group of instructions which tells the analyzer which source file he wishes to work and which allows him to continually switch back and forth between these files as he goes.

4.b.3 3. Set operations - these instructions are used to empirically construct the boolean combinations of characteristics in the data.

4.b.4 4. Summarization - The Analyzer permits many types of data summarization from simple marginals to complex co-occurrence tables.

4.b.5 5. Recoding - The Analyzer has instructions for certain necessary recoding of category records as analysis proceeds.

4.b.6 6. Multi-level files - As the analyzer permits integrated analysis of multi-level files (i.e. the data exists at different levels of aggregation or can be sub-divided into groups), instructions which allow the user to pass between different grouping levels are required.

File Focus

4.c.1 WORK - The user types the names (these names are the names of the source files the user assigned to the category records when he inverted the item record file) of the files he wishes to work. All subsequent relevant instructions will then be applied, in parallel, to each of the files being 'worked'. These files constitute the 'worklist'.

4.c.2 UNIVERSE - The user types a list of all the files in his universe of discourse. Whenever he chooses he may then type 'work universe' and all the files he declared to be in his universe go into the worklist.

4.c.3 ADD - The user types the names of the files that should be added to the worklist.

4.c.4 SUBTRACT - The user types the names of the files to be removed from the worklist.

4.c.5 WORKLIST - The Analyzer prints back the current worklist.

Classified Directory Maintenance

4.d.0 The classified directory consists of all the names assigned to all the indexes constructed across all the files that have been worked. The user may assign two 'names' to each index; one less than 6 characters (a symbol) and the others between 7 and 30 characters. The name (7-30) may be broken down into facets by using '.' as a separator.

4.d.1 LIST prints out a list containing for each index the source file name, the index name, the index symbol, the index size, the paragraph number in which the index was created. If LIST is typed alone all indexes are listed. Alternatively, specific indexes can be asked for either by name or by symbol. As well one can make implicit requests using a '*' notation: For example: LIST BR*4 lists all indexes whose symbol has B as character 1, R as character 2, and 4 as character 4; LIST BRIT*.CHINA lists all indexes whose first facet is BRIT, third is CHINA. These may be intermingled in one request.

4.d.2 LISTC is the same as LIST except LISTC explicitly documents the construction of the indexes listed, i.e. what operators acting on which categories and entries and in what sequence produced the index.

4.d.3 ALLIST - whereas LIST and LISTC operates in parallel over the files in the worklist, ALLIST gives each index requested (same conventions as LIST) and under it the files in which it exists; they need not necessarily be on the worklist at the time the instruction is given.

4.d.4 SOURCES prints the index names and the set operation from which its arguments were built.

4.d.5 RESULTS prints all the index names which have the arguments as one of their direct parts.

4.d.6 NAME is used to assign a symbol/name to a name/symbol or to change a name/symbol. It is effective across all files, not necessarily those on the worklist.

4.d.7 RECLASSIFY is the same as NAME except it only affects those files on the worklist.

4.d.8 DELETE is used to delete indexes for files on the worklist.

4.d.9 DEPAGE - The classified directory is kept in a paged memory and may be removed from one Analyzer core image and transferred to another. DEPAGE is used to write the directory into a disk file 'NAME PAGES' where NAME is specified as an argument in the DEPAGE instruction. This is very handy because a user can build a very complex classified directory and then store it on the disk in a small amount of space. This directory can then be read into an 'empty' Analyzer core-image belonging to its author or perhaps to some other individual who wished to use the complex classified directory.

The Analyzer automatically depages itself every 10 directory altering instructions providing backup protection to the user.

Set Operations

4.e.0 To name the result of a set operation instruction the arguments should be followed by an '=' and then a name and another '=' and a symbol. Symbols may precede the name, and, if the user wishes, only one of the two need be present. If neither is present the summary line for that operation is printed but the actual index is not built and therefore never saved, i.e. no directory entry is made for it.

4.e.1 INDEX is used to build an index to items possessing a specified category or an entry in a category; or one or more of a list of entries in a category; or if the category is numeric, to items whose value possess the relationship (less than, greater than, or equal to) with a specified constant. The summary line returned for each file on the worklist is file name, population, index size, index name--if assigned.

4.e.2 INTERSECT is used to build an index containing those items in <u>every one</u> of the arguments of the INTERSECT instruction. (Up to 10 indexes may be specified.) The summary lines returned contain the arguments and for each file on the worklist, the file name, argument sizes, file population, result size, Fisher Exact Text,* applied to the intersection if but 2 arguments were specified, and the result name--if any.

4.e.3 UNION is similar in form to INTERSECT except the result index contains items present on <u>at least one</u> of the arguments. And, of course, no Fisher Exact Test is applied.

4.e.4 COMPLEMENT is used to build an index to items not in its single argument. Again the summary line is similar to INTERSECT.

4.e.5 RELCOMPL - The relative complement is used to build an index to all items in its second argument which are not in its first argument. That is, RELCOMPL is the complement of the second argument relative to a universe specified by the first argument.

* The FET is a statistic which measures the randomness of the size of an intersection.

Summarization

4.f.1 SUBJECT prints the aggregates, aggregate percentages and the subject description for the category specified. If particular entries are specified, only they will be printed.

4.f.2 INTERVAL takes as argument the name of a category whose entry is a numeric value, indexes (optionally) and percentage breakpoints. It orders the entries (for only those items in the indexes if indexes were specified) and prints out the value of the entry at each percentage breakpoint.

4.f.3 DISPLAY is used to display the actual entries for items in an index. It accepts as arguments the categories to be displayed and the index.

4.f.4 PERMUTE is used to simulate the combining of entries in a category. It accepts as arguments a category, a list of entries, an index and a threshold value for the Fisher Exact Text. It then simulates the recoding of the entries supplied by trying all permutations of the entries as a single entry intersected with the supplied index. The entries which produce a FET value over the given threshold value (or only the highest one--if requested) are printed out.

4.f.5 PATTERN - As the data may include categories with multiple entries in the item the user may be interested in a summarization of the patterns as opposed to the individual entries--which can be obtained from the TABLE instruction. PATTERN accepts as arguments a multiple-entried category and (optionally) indexes. It prints out for each entry pattern in the data the number of occurrences in the total population and, if indexes were supplied, in the sub-population specified by each index.

4.f.6 VALUES prints the number of occurrences of each value in the numerical category record supplied, as well as the number of occurrences for the items in the indexes--if they were supplied.

Co-occurrence Tables

4.f.7 The tables generated are of two types.

1. The columns contain indexes and the rows contain categories, i.e. each entry in the category is a row.
2. Both the rows and column contain indexes.

Tables are either labelled (TABLES instruction) or unlabelled (FIGURES instruction). They are printed on the console and may be also printed on the disk (DISK instruction). Many types of percentages may be printed on the tables (STATS instruction) as well as the Fisher Exact Text. Users are encouraged to program their own statistical tests as well.

Tables are printed in horizontal parallel form with respect to the worklist.

4.f.8 COLS is used to specify the indexes on the columns.

4.f.9 ROWS is used to specify the rows, which are either indexes or categories (in which case each entry becomes a row).

4.f.10 DISK specifies the name of a disk file on which all tables should be written until a new name is specified or DISK is turned 'off'.

4.f.11 TABLE prints a labelled table on the console. For indexes the labels are the names assigned to them. For categories and their entries the labels are the english subject descriptions from the adform. The cells of the table contain of course the sizes of the intersection of the indexes on the rows and columns forming the co-ordinates of the cells.

4.f.12 FIGURES prints an unlabelled table. If a disk file has been specified a labelled table is printed there.

4.f.13 STATS is used to specify the statistics which appear on the table. The currently available ones are:

PC/TOT - each cell as a percentage of the total population of the file.

PC/ROW - each cell as a percentage of the row marginal.

PC/COL - each cell as a percentage of the column marginal.

PC/'N' - the ratio of the cell to the nth cell in the row as a ratio of the column marginal over the nth column marginal.

SIG - Fisher Exact Test

OFF - turn all statistics off. The STATS settings apply till they are changed or turned off.

4.g.1 RATIO is used to produce a category record which, item by item, is the ratio (over 100) of two other numerical category records.

4.g.2 COUNT is used to produce a numerical category record, where each entry is the aggregate of that item over a list of supplied indexes. It may be used to simulate 'threshold' or 'majority' logic of the kind: if an item has n out of m specified characteristics, classify that item which characteristic X.

4.g.3 RECATEGORIZE builds a category record whose entries correspond to the indexes specified as arguments. It can be used to regroup a group of related complex indexes. This can be done, as well, by writing an adform with Admins Binary input.

Multi-Level Files

4.h.0 A file is multi-level if a subfile based on bibliographic tokens in the main file exists (or is constructable). The selection of the items which will make up the subfile may be demographic, kinship, or based on any arbitrary index. The instructions for dealing with multi-level files have two purposes:

1. Construction of a subfile from a main file, i.e. the selection of the subfile bibliographic tokens.
2. Relating data in the subfile to the items in the main file and vice versa and aggregating data in the main file over the subfile.

The full power of the other Analyzer instructions can be brought to bear in analyzing subfiles in relation to the main file.

4.h.1 UNIQUE accepts a category whose entries are bibliographic tokens and produces a category record containing each unique occurrence of a token. A category record containing bibliographic tokens is called the 'basis' for the subfile.

4.h.2 SUBFILE creates a category record containing the bibliographic tokens only for those items in a specified index. This instruction (in conjunction with the UNIQUE instruction) permits the user to build a basis for any arbitrary subfile.

4.h.3 MAP - This instruction maps category records from one file into another, using the respective bases as a cross reference between files.

4.h.4 AGGREGATE - This instruction aggregates the items in an index in one file over the items in a second file, using the respective bases as a cross reference between the files.

4.h.5 SORT orders a category record of bibliographic tokens.

4.i.1 The SYNTAX instruction can be used to ask the Analyzer to print the abbreviations and syntax of the instructions specified as arguments. The syntax for all instructions are kept on a disk file 'SYNTAX CROSS' and are read in when the analyzer is initiated. That is, each analyzer core-image contains some self-documentation.

4.j.1 The user may type one or more instructions (with arguments) to the Analyzer; the instructions are separated by commas. A '/' kills the line being typed back to the last comma. An extra carriage return tells the Analyzer to execute the above instructions. When completed, the Analyzer types a paragraph number (cross referenced in the classified directory) and the user may type the next paragraph, which may be but one instruction.

4.k.1 The Analyzer is error-proof in the sense that the user cannot give an instruction in error, which causes him to lose some work inadvertantly or lose his core image. If the Analyzer detects a user error, it prints a clear explanatory message and is unable to 'harm' itself. All error messages are kept on disk file 'ERRORS CROSS' and are read into core when needed, thereby saving space and putting no storage limitation on the richness of error messages.

Storage Management

4.1.0 As there are five basic elements which require storage during analysis let us relate each separately to the storage management issue.

4.1.1 1. Programs - As the analyzer offers an ever growing repetoire of instructions, core might be exceeded someday by programs alone. Therefore, the Analyzers user-interface links up with the subprograms embodying individual or small clusters of instructions at execution time. It reads the entry points of all instructions from a disk file table. This allows the Admins administrator to make an Analyzer image tailored to a users needs, i.e. only containing those instructions a specific user requires. A misjudgment does not involve loss of work, for the user can always depage an ill-suited Analyzer image and repage his directory into a freshly prepared one.

4.1.2 2. Classified Directory - As the classified directory is paged (512 words to a page) and all address pointers in the classified directory are page and word numbers (i.e. relative), directories are relocatable across Analyzer core images. As well they are a compact form of backup and saving ones work. As the classified directory solely contains directory (i.e. prototype) information and only points to blocks of token pointers which are the actual indexes, classified directories are only a few disk records in length.

4.1.3 3. Indexes - Indexes are lists of token pointers tightly packed in blocks of storage. The information as to what an index is, i.e. the sequence of set operations which led to its construction, is in the classified directory. That is, the indexes (token pointer lists) are dispensable. They can always be rebuilt, as required, by a recursive procedure 'evaluating' the classified directory. Realizing this, the Analyzer purges (that is, returns to free storage) all unessential indexes whenever memory is full. This guarantees almost indefinitely long analysis, until the directory itself overruns memory which seems very unlikely because it would involve an extremely complex analysis.

4.1.4 4. Data - The actual data, the category records, are kept on disk until they are specifically required. Whereupon they are read into memory, used, and immediately deleted from memory.

4.1.5 5. Comment information - The error comments, which are numerous and rich, and an explanation of each command will be kept on disk until specifically required, then read, printed or otherwise displayed but not kept in core.

4.m.1 The data manipulation functions of the Analyzer have been separated from the application instructions. This gives us a language (in the sense of an integrated group of subprograms) independent of the Analyzer. As the rational reconstruction of this language is yet incomplete it would be premature to specify it precisely. When complete, however, it would offer to the sophisticated user a language, based on MAD, for referencing a data base, operating upon it, and keeping a classified directory to what he is doing. This would allow users to program their own information models (say a special purpose analyzer or a heuristic tree analyzer--as one of the authors has done with a now obsolete version of the language) or scientific models (say a data based complex simulation).

DYNAMIC LOOPS

Having described the statics of the three Admins sub system--Organizer, Processor, Analyzer--one can now relate this to descriptions of the Admins use dynamics.

Organizer-Processor Loop

5.a.1 This is the interactive process involved in preparing data for analysis. The problem is to find agreement between a normative description of the data which will support one's analysis and the actual data. Any discrepancy between data and prototype, other than that caused by a clerical error on the part of the user, e.g. mistyped adform, can be resolved in one of two ways: change the prototype (adform) or change the data (item record file). The Organizer-Processor loop is concerned with the flow of information involved in deciding whether to change adform or data and how, as well as with the tools used to implement these decisions.

5.a.2 In writing his adform the user has a hard copy description of the data and an analysis purpose. He may have, as well, aggregate information from previous analysis results, perhaps from Admins if his input is Admins Binary. He uses the Organizer language to construct an adform which expresses the data prototype as he believes it now exists and as he would have it transformed. He types the adform onto the disk using a context editing program EDL* to input and alter the adform. Then the user types the ORGANIZE command, most likely with the diagnostic option. Users being human the Organizer finds syntax and/or coherence errors. A description of each error is printed narrowing the error source down at least within the category. This information is used to change the adform, whereupon it is resubmitted for organization by typing the ORGANIZE command again. Usually diagnostic errors are clerical ones but occasionally the coherence checks pinpoint actual logical flaws in the adform.

*op cit

5.a.3 Eventually the user comes out with an 'organized adform'. The PROCESS command creates a process image which will apply the adform to the data. At least the first time through a new data file the user will run in 'dummy' mode, that is, not producing an output file. The user will sample a few input items and follow the input/output changes with his adform in front of him. This may turn up enough errors in the adform to justify quitting out of the process image and changing the adform.

If the user suspects the whole file, or perhaps just certain categories, are particularly error prone he issues the appropriate control instructions. He then may process part of the file. Error comments are printed out as the data is processed. Eventually he terminates the process, and gets the marginals and summary error report using the MARGINAL and REPORT commands, respectively. He now has produced the following information in addition to information existing prior to processing.

1. Sample item records.
2. Error messages.
3. Aggregates of the entries and categories over the part of the file processed.
4. A summarized error report.

Decisions to ORGANIZE and change the adform are implemented with EDL. The PROCESS command can then be used. Decisions to change the data are implemented with the ALTER command after an item record file has been produced and inverted.

5.b.1 Analysis of category records may prompt the user to reconceptualize his data into macro-categories his empirical analyses have shown to be fuitful. He can write an adform which publicly states these macro-categories, that is, an adform which combines entries from different micro-categories into macro-categories. This adform can be applied to the category records and as described in the Organizer-Processor loop discussion yield an item record file. This item record file can be inverted and analyzed along with any other category record from the same source file.

Vertical and Horizontal Processing

5.c.1 The Processor can be used to produce an item record file which is a sample of the source file. The samples can be staggered or random or contiguous. If the sample is contiguous, i.e. taking different chunks of the file, the chunks can be processed separately, and concentrated with the APPEND command; analyzed separately or analyzed together after concatenation. These flexible options are useful in a current data application where items are being originated over time and need be added to a 'master' file.

5.c.2 An adform need not describe all the categories in a file. One can have many adforms, each describing from one to all of the categories in the file. Each adform (selection) can be processed separately but the category records inverted from the item record file can be analyzed together.

Multi-Source Files

5.d.1 One can analyze multi-source files by writing adforms for each file and processing them separately. Category records from many adforms across one or may files can be analyzed in parallel by the Analyzer and reprocessed if desired, as Admins Binary input. The coding across files can be different at the category and/or entry level. The worklist feature and the classified directory of the Analyzer can be used to build up the characteristics of each source file until a classification valid across the files is applicable. Then parallel summarizations can be produced for the different source files.

Multi-Level Files

5.e.1 The Analyzer can be used to construct arbitrary subfiles from a main file. As well Admins can bring already existing subfiles to analysis stage, and relate them to the main file; constructed subfiles; constructed category records from the already existent subfile. Subfiles may contain categories of factual data from the main file, or aggregates of characteristics in the main file. Since all subfiles are in Admins binary form high level adforms can be written which regroup the characteristics of the subfile. The main file and the subfiles (constructed and/or 'inputed') may be multi-source, of course, as discussed in 5.d.1.

5.f.1 The task of describing all that Admins can do is a subset of the task of describing all that a computer can do given that it obeys the instructions add, store, and transfer on minus; that is an impossible task. What we hope we have succeeded in doing is giving a flavor of the power the systemic generality of Admins affords.