AD-754 978

THE CAMBRIDGE PROJECT: COMPUTER METHODS FOR ANALYSIS AND MODELING OF COMPLEX SYSTEMS

D. B. Yntema

Massachusetts Institute of Technology

Prepared for:

Rome Air Development Center
Advanced Research Projects Agency

May 1972

AD754978

RADC-TR-72-169
Technical Report
May 1972

THE CAMBRIDGE PROJECT: COMPUTER METHODS FOR ANAYLSIS AND MODELING OF COMPLEX SYSTEMS

Sponsored by
Defense Advanced Research Projects Agency
Human Resources Research Office
ARPA Order No. 1756

Approved for public release;
distribution unlimited.

Rome Air Development Center
Air Force Systems Command
Griffiss Air Force Base, New York

25

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Massachusetts Institute of Technology The Cambridge Project	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED
	2b. GROUP

3. REPORT TITLE

The Cambridge Project: Computer Methods for Analysis and Modeling of Complex Systems

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Semi-Annual Report July 15, 1971 - Jan. 15, 1972

5. AUTHOR(S) *(First name, middle initial, last name)*

Yntema, D. B., Project Director and Cambridge Project Participants

6. REPORT DATE May 1972	7a. TOTAL NO. OF PAGES 22 pages	7b. NO. OF REFS

8a. CONTRACT OR GRANT NO. F30602-72-C-0001	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO. ARPA Order #1756	
c. ID20	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* RADC-TR-72-169
d.	

10. DISTRIBUTION STATEMENT

Approved for public release;
distribution unlimited.

11. SUPPLEMENTARY NOTES by: S. S. DiCarlo RADC/IRDA Griffiss AFB NY 13440	12. SPONSORING MILITARY ACTIVITY Advanced Research Projects Agency 1400 Wilson Blvd. Arlington VA 22209

13. ABSTRACT

The Cambridge Project is a cooperative effort by a variety of scientists, both at M.I.T. and Harvard to make the digital computer more useful in the behavioral sciences, basic and applied. This Technical Report summarizes progress during the half year beginning in July of 1971 -- i.e., the half year immediately following the period covered by the Project's second Annual Report.

The Project has two main goals: first developing programs and other computing tools that are needed in behavioral science and its applications; second, combining these tools, and others borrowed from other sources, into a Consistent System of programs, models and data that behavioral scientists can use on-line. In both of those areas, the primary effort is devoted to theoretical studies of statistical techniques, equipping a computer-based laboratory for the study of autonomic conditioning, and so on.

The most conspicuous progress during the time covered by this report was in the development of the Consistent System. During this period the foundation for the system went into service within the Multics time-sharing system at M.I.T., and a library of programs began to collect upon that foundation.

DD FORM 1 NOV 65 1473 (PAGE 1)
S/N 0102-014-6600

ia

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Computer programming Behavioral science Statistical analysis Time-series analysis Computer based modeling Systems Analysis						

THE CAMBRIDGE PROJECT: COMPUTER METHODS FOR ANALYSIS AND MODELING OF COMPLEX SYSTEMS

D. B. Yntema

Contractor: Massachusetts Institute of Technology
Contract Number: F30602-72-C-0001
Effective Date of Contract: 15 July 1971
Contract Expiration Date: 15 July 1972
Amount of Contract: $1,696,500.00
Program Code Number: 1D20

Principal Investigator: J. C. R. Licklider
Phone: 617 253-7705

Project Engineer: Samuel S. DiCarlo
Phone: 315 330-2146

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and was monitored by Samuel S. DiCarlo, RADC (IRDA), GAFB, NY 13440 under Contract F30602-72-C-0001.

TITLE: The Cambridge Project: Computer Methods for Analysis and Modeling of Complex Systems

PUBLICATION REVIEW

This technical report has been reviewed and is approved.

[illegible]

RADC Project Engineer

SUMMARY

The Cambridge Project is a cooperative effort by a variety of scientists, both at M.I.T. and at Harvard, to make the digital computer more useful in the behavioral sciences, basic and applied. This Technical Report summarizes progress during the half year beginning in July of 1971 -- i.e., the half year immediately following the period covered by the Project's second Annual Report.

The Project has two main goals: first, developing programs and other computing tools that are needed in behavioral science and its applications; second, combining these tools, and others borrowed from other sources, into a Consistent System of programs, models, and data that behavioral scientists can use on-line. In both of those areas, the primary effort is devoted to writing and documenting programs; however, lesser amounts of effort have been devoted to theoretical studies of statistical techniques, equipping a computer-based laboratory for the study of autonomic conditioning, and so on.

The most conspicuous progress during the time covered by this report was in the development of the Consistent System. During this period the foundation for the system went into service within the Multics time-sharing system at M.I.T., and a library of programs began to collect upon that foundation. These developments are described in Section 6 under five headings: the Substrate, conventions, utility subroutines and basic programs, documentation, and growth of the library.

Sections 2, 3, 4, and 5 are concerned with the first of the two main goals, developing programs and other computing tools. They are divided as follows:

Section 2 is on data-handling,and summarizes progress in: the development of Janus, which is an advanced system for handling data of a type common in the behavioral sciences; a facility that employs a set-theoretic approach to handling behavioral science data; programs that assist an investigator to locate within a collection of surveys questions that are relevant to his problem; and subsystems for handling natural-language text.

Section 3 is on data-analysis, and summarizes progress in: basic statistical methods and algorithms; incorporating into the Consistent System the statistical programs from Toss, which was an experimental version of a consistent system; incorporating a group of interactive statistical

programs developed for a PDP-10; incorporating the Time-Series Processor system; application of spectral analysis to time-series; fitting non-linear growth models to longitudinal data; programs for analyzing networks of relations or transactions; testing the General Inquirer's facilities for distinguishing between homographs in analyzing English text; and construction of two thesauruses to be used in analysis of policy documents.

Section 4 is on modeling, and summarizes progress in: DISCOURSE, an interactive system for constructing models in which geographic locations are important; the Implicator, which helps investigators to transform social theories from words into computer models; interactive facilities for Bayesian logistic regression models; evaluation of a set of algorithms for modeling social networks; modules for probabilistic models, especially decision models; analysis of project-planning networks; validating against data from many geographic sub-areas the simulation of processes that extend over a wider area; and adapting to Multics the CASCON system for modeling information on local conflicts.

Section 5 is on human factors in interactive computing and on computer control of experiments. It summarizes progress in: a computer-based laboratory for the study of autonomic conditioning; a group of programs for manipulating digitized sound waves; the Interactive Graphic Decision Aid; the On-line Decision-Making Laboratory; and a CRT-based program for studying interaction between a computer and users with "intuitive" thinking styles.

Work in both areas -- the development of a Consistent System, and the development of particular tools for behavioral science and its applications -- will continue during the coming half year under the same contract.

I. INTRODUCTION

The Cambridge Project is a cooperative effort to make digital computers more useful in the behavioral sciences, both basic and applied. This semiannual technical report covers the half year that began in July of 1971, where the second Annual Report left off.

The purposes of the Project were set forth in some detail in Part I of the first Annual Report, and were reviewed briefly in the second. As was explained there, the Project has two main purposes: to develop computing tools needed in the behavioral sciences, and to assemble such tools into a Consistent System that the scientist can use interactively. The tools will, for the most part, be programs, but may also include models, or sets of generally useful data. The Consistent System will help ensure that they can be used in unexpected combination by the researcher who may not be a programmer in the traditional sense. We expect that in the Consistent System these tools will therefore be much more useful than they would in isolation.

Computer scientists, statisticians, and a variety of behavioral scientists from both M.I.T. and Harvard are cooperating in the Project. As in previous years, some 60 members of the two faculties participated. They are complemented by numerous students and research personnel, and there is also a staff of full-time employees responsible for designing and constructing the framework of the Consistent System, for providing some of the basic tools that form the collection, and for coordinating the documentation of the rest.

During the first two years the Project was supported by the Advanced Research Projects Agency under Contract DAHC15 69 C 0347.

2. DATA HANDLING

The Project's general plans in the area of data handling were detailed in Section D.I of Part I of the first Annual Report. They range from specific programs for which there is an immediate need to general research on managing data of the kinds that behavioral scientists use.

Principal among the specific programs is the Janus system for interactive data handling and analysis. The overall objectives are to provide a language interface between users and their data, to perform common data-preparation activities, to provide formatted output, to interface with other analysis procedures in the Consistent System, and to permit inspection, manipulation, creation, and cross-referencing of datasets. More details are given in Section 2.I of the second Annual Report.

A prototype of Janus has been working in Toss (an exploratory version of the Consistent System) since November. Recently, it has been modified so as to work within the Consistent System. Extensive timings and testing of the prototype are near completion and a small group of experimental users is being introduced to it.

Feedback from these activities is being used in designing the final version of Janus. Judging from results from the prototype version, the basic design appears well suited to Multics and to the extended capabilities planned for the final version. Design of the final version is well under way, and active implementation will probably begin in late spring.

A set-theoretic approach to data management is also being explored. A prototype implementation is complete and is undergoing performance tests with various types of databases and queries. A direct comparison of this approach with more traditional approaches to data-management is being undertaken with a view to users' anticipated requirements.

Because large-sample surveys can be expensive, multiple use of them is desirable, and collections of surveys have therefore been established (e.g., The Roper Center). However, information pertinent to the investigator's interest may be distributed through many surveys, and he may have difficulty finding it. Some basic software components (the data base generation and retrieval command language) that assist in finding identical or similar questions in different surveys have been made operational with a subset of the Gallup Survey files. The language allows a user to make queries about the number of questions satisfying certain logical conditions and then probe the text of the questions and other data-descriptor information. The system allows a flexible definition of user-environments that accomodate different research styles and interests. The present software is based upon CP/CMS. Current work is on documentation and problems associated with storage of many large files within the Consistent System.

In the area of text-handling, a number of subsystems have been implemented from the components discussed in Section 2.2 of the second Annual Report. Two serious difficulties have been encountered. First, sufficiently accurate tools to monitor CPU usage were not available in Multics and subsystem tuning suffered as a result. Second, attempts to provide subsystems with a full range of powers available in Multics led to unacceptably high overhead. It appears that many of the features that distinguish Multics from other operating systems cannot be used effectively at present by subsystems of this type. However, valuable experience in problem conceptualization was gained, and during the next six months some programs will be transferred to the IBM 370/155 at MIT.

3. DATA ANALYSIS

Section D.2 in Part I of the first Annual Report explained that the Project's general plans about the analysis of data include the development of tools for four kinds of analysis: multivariate techniques, primarily multidimensional scaling; analysis of networks; analysis of longitudinal data, like time-series; and analysis of natural text. In all four cases we are interested not only in the development of new methods, but in the evaluation of those that already exist.

During the period covered here, basic work in regression analysis has centered on an empirical study of alternatives to the ordinary method of least squares. Several have been found to be superior, and a report of the findings will be prepared separately. An attempt to fit logistic models to counted data with random effects is in the preliminary stages, and an interactive system for fitting log-linear models to multidimensional contingency tables has been specified. Also, Portnoy has refined his procedure for determining the dimensionality of non-linear data structures by a method of local principal components and two reports on it are available: "Local Principal Components; A Method for Determining the Dimensionality of Non-Linear Data Structures" and "A Computer Program for Testing and Applying the Method of Local Principal Components". Previous work in all these areas was described in Section 3.1 of the second Annual Report.

Two doctoral dissertations on statistical methods were completed with contract support. One, by R. Heiberger, is a three part thesis on statistical computation. The other, by B. Rosner, is on a semi-Bayesian approach to outlier detection. Copies of each are available.

A number of common statistical procedures, about fifty in all, are being introduced into the library of the Consistent System. Most of them are from the earlier Toss system and are listed in Section 3.6 of the second Annual Report. They include frequency distributions; measures of central tendency, variability, and higher moments; parametric and non-parametric tests; product moment and rank order correlations; and simple regression. Moving these procedures from Toss to the Consistent System was made easier by their design: the computational "kernel" of each of them was written as a closed subroutine, and an interface can be easily supplied to honor the requirements of the Consistent System. Furthermore, the kernels can be used as components in one or another of the statistical "packages" proposed for construction, or could be called directly by suitable data management interfaces.

Special efforts are being directed toward analysis techniques that take advantage of the interactive environment of Multics. A collection of such programs, including many basic statistical programs as well as a newly-designed program for interactive factor analysis, is being readied for the Consistent System. The collection was originally written in FORTRAN on the PDP-10. It is now almost completely debugged and, so far as possible, all system-dependent features are being removed in preparation for adapting it to Multics.

In the area of longitudinal methods, TSP (the Time-Series Processor) has been adapted to Multics and is being introduced into the Consistent System. TSP is a sizeable system of FORTRAN programs aimed primarily at the analysis of economic time-series. It was designed for batch processing, and modifications that will exploit the interactive possibilities of Multics are being studied.

Several applications of spectral analysis to economics have been explored. (As an example, a significant effect of the price of capital on national investment has been found for components around the business cycle frequencies.) A set of spectral programs, based on estimation using the fast Fourier transform algorithm, are being tested prior to implementation in TSP; these will compute (cross-) periodograms and smoothed spectral estimators using "prewhitening" and alignment. A report is in preparation detailing computation of regression estimators using only part of the spectrum -- important in economics, when it is desirable to purge a regression of its seasonal or other frequency components.

Several programs for longitudinal analysis aimed at the validation of non-linear growth models have also been developed and documented. The programs and their descriptions are available.

The program SOCPAC, which analyzes the triad structure of sociometric networks, has been modularized with a view toward bringing the modules onto Multics within the Consistent System, and a set of notes is being prepared on Statistical Modeling of Network Data.

RANULL, a program for analysis of transaction flows, and DICHOT, a program for a hierarchical dichotomization of transaction matrices by the McQuitty-Clark method of iterative intercolumnar correlations, are being transferred to Multics for inclusion in the Consistent System. It is expected that RANULL, which uses the "null model" of Savage and Deutsch, will be integrated with two newly-written programs, DECISION and RWC_POWER. The first of these is a simple programming of the Day-Tinney model; the second calculates Dahl's "power" index and Chadwick's "control" index. A fuller discussion is given in a report by Richard W. Chadwick: "Power, Control, Social Entropy, and the Concept of Causation in Social Science". Previous reports on related work were listed in Section 3.10 of the second Annual Report.

As part of the work on the General Inquirer, a large effort to provide subroutines that disambiguate English homographs is nearing completion. (See Section 3.14 of the second Annual Report.) Test runs indicate 93% success in identifying the correct sense in a sample of 670 difficult English words. Supporting programs, including a retrieval specification format in pseudo-PL/I that is converted into real PL/I via the pre-compiler, are now in final stages.

In a parallel effort in the analysis of text, two thesauruses are being constructed. An expert on the domestic and foreign politics of the USSR has developed concept hierarchies and the beginnings of a thesaurus for Soviet documents. A thesaurus for American documents is being constructed automatically from text taken from "The New York Times". A report entitled "Data Sets for the Development of a Document Encoding and Retrieval System: A Description of Computer Readable Texts and Queries" by Rosemarie Rogers is available. For earlier work, see Section 3.13 of the second Annual Report.

4. MODELING

The general plans for tools for building and evaluating models were described in Section D.3 of Part I of the first Annual Report.

Several new features have been added to DISCOURSE, which was discussed in Section 4.1 of the second Annual Report. It is an interactive computer language for addressing problems in which geographic parameters are important. The language has been modified so the syntax of a DISCOURSE command now accomodates use of non-DISCOURSE functions, general expression evaluation, and recursively imbedded strings of characters. Efficiency of operation has been improved by hashing all names through a single table that includes synonyms and by providing a preprocessed mode that allows preinterpretation of commands.

The basic input and retrieval programs for the General Implicator, which was discussed in Section 4.2 of the second Annual Report, are running reliably. Current work is focused on building out of those basic programs macro-programs for special applications -- for example, a program that will identify causal or inferential chains. Also, the present implementation in PL/I is being compared with possibilities in list processing and other special languages.

The Bayesian logistic regression model described in Section II.D.7 of the first Annual Report is being fitted into an on-line data analysis system that emphasizes Bayesian methodology. This is a step toward introducing it into the Consistent System.

Some evaluation of the social network models described in Section 4.6 of the second Annual Report have been made in an attempt to identify those characteristics of the networks which have predictive and descriptive value. A number of routines have been developed in the APL language to search for such characteristics as cliques, clusters, cycles, semigroup homomorphisms, subgraph isomorphisms, and graph automorphisms. The results have been disquieting, and there is some question about which of the routines should be included in the Consistent System.

A set of tools for probabilistic models, decision models in particular, which were described in Section 4.7 of the second Annual Report are a step nearer completion. The primary effort over the past six months has been on debugging. During this period, however, the system has been used for graduate and undergraduate research projects. Considerable reprogramming and documentation efforts are being done in preparation for introducing these programs into the Consistent System.

In the area of Project Network Analysis, described in Section 4.8 of the second Annual Report, four interactive computer programs have now been developed: a program for creating and editing a network; a program for analysis of a network in terms of physical structure and other parameters; a program which uses some of the outputs of the previous program to predict the result of a particular scheduling algorithm; and a program that uses one of five algorithms to produce a schedule for the network. In addition, batch versions of two of the programs have been developed. A manual describing the programs is in preparation by E. W. Davis.

Simulation techniques for small geographic areas, described in Section 4.11 of the second Annual Report, have made progress on three fronts. A streamlining of data reduction programs into a sequence of well-defined steps has been accomplished. A "zoom" capability has been added in the mapping routine to permit examination of a single area. Finally, a "tuner" is nearing completion. It will take a theoretically sound model that approximates the data reasonably and will adjust the model to its ultimate accuracy.

Finally, support has been given to an effort to adapt to Multics the CASCON system, the Computer-Aided System for Information on Local Conflicts.

5. HUMAN FACTORS AND CONTROL OF EXPERIMENTS

General plans for the development of methods for computer control of experiments were detailed in Section D.4 of Part I of the first Annual Report, while research on human factors in interactive computer usage was discussed in Section E.3.

The computer-based facility for study of autonomic behavior is now in use and was described in Section 5.1 of the second Annual Report. Experiments are being done on assisting people in gaining voluntary control over their autonomic activity -- e.g., heart rate. The presumption is that if people can control their autonomic arousal under stress, their ability to perform perceptual and motor tasks under stress will improve. Specialized hardware and software have been developed to: (1) train people to gain control of heart rate for a short time, (2) train them to become increasingly accurate in assessing their autonomic arousal, and (3) test perceptual performance under stress. The immediate conclusion is that some people can gain control of heart rate, and that they can exercise the control to improve their performance. Future work will be concerned with individualizing the training process so that most people can learn to control heart rate for long periods of time.

A Conversational Audio Signal Manipulator (CASM) is being developed on the PDP-9T in the Computer Based Laboratory of the Harvard Psychology Department. CASM has three parts: an interpretive component, which decodes the user's teletype requests to transform audio signals and move them between storage devices; a special disk directory scheme for referencing stored signals; and a battery of transformation routines.

Development work continued on the Interactive Graphic Decision Aid which is being programmed on a PDP-10 computer with an LDS-1 display, and was described in Section 5.3 of the second Annual Report. The system exploits computer graphics technology in order to apply statistical decision theory as a practical tool for decision makers. Preparations are underway for observing the effect of such an aid on a real decision maker.

The On-Line Decision-Making Laboratory, discussed briefly in Section 5.4 of the second Annual Report, is now operating. The new simulation model with which the players interact is in the final stages of debugging and the information recording programs are operating on a PDP-10. In March the game model will be run with over 700 graduate students participating, and two trial tests of the information gathering system, with 150 students participating in each, will be conducted during this period. At present the information gathering system is passive; it records the time and nature of decisions and the patterns of use of on-line models. The next stage is to develop active cues to allow the decision makers to be questioned about causes of unusual activity.

A version of a CRT-based program to study the software elements that facilitate "intuitive" man-machine interaction has been implemented and revised. It was described in Section 5.6 of the second Annual Report. Data on subject performance is in the process of collection and evaluation.

6. THE CONSISTENT SYSTEM

One of the major goals of the Cambridge Project is to forge into a "Consistent System" tools such as those described in the preceding sections. The System was discussed in detail in Chapter 6 of the second Annual Report, especially in Sections 6.1 and 6.3. It is to be a collection of models, data, and programs for use by the nonprogramming behavioral scientist. If the innovative investigator wants to combine the tools in some novel way to meet his special needs, he should not need a programmer to help him. He should be able to introduce his own data into the system and, if he is an amateur programmer, he should be able to introduce his own programs too; moreover, his programs and data should work gracefully with those already in the collection.

The framework that holds the collection together consists of: a Substrate, which is a package of routines, somewhat like an operating system, on which the programs in the collection depend; a set of conventions that programs and data are expected to obey; and standards for documentation. During the past half year most of the efforts of the Project staff have been devoted to providing that framework and some of the more fundamental programs and subroutines that fit within it.

The projections about progress in building a framework that were made six months ago in the second Annual Report proved remarkably accurate. A Substrate based on the Multics time-sharing system at M.I.T. was completed essentially on schedule, in spite of the fact that there were, inevitably, a few design changes along the way. A small set of utility subroutines that a program may use to handle files, communicate with the Substrate, etc., is ready, and so is a small group of programs that form the nucleus of the collection - the programs needed to test other programs. The conventions they embody are reasonably well established, though they need to be codified and re-stated to make them easier for programmers to understand. Documentation, as predicted, has lagged behind.

The rest of this section will cover these topics in more detail.

<u>The Substrate</u> -- The Consistent System sits upon a software foundation known as the "Substrate". The Substrate serves as an interface between programs in the Consistent System and the Multics operating system -- programs in the Consistent System do not call upon

Multics directly. The Substrate is the operating system from a program's point of view; of course, the implementation of the Substrate calls heavily upon Multics.

The Substrate has three major purposes: it buffers programs against instabilities or changes in the Multics operating system; it is designed to be implemented on hosts other than Multics; and it provides a number of services that are needed by a collection of this kind, but are not available in all the operating systems that might eventually serve as hosts.

The first job, insulating programs against changes in the operating system, is straightforward. The Substrate was so designed that small changes in Multics can be compensated for internally, preserving the external specifications guaranteed to programs in the collection. A program contacts the operating system only indirectly, through the Substrate and through the "run-time support" package of the programming language in which it was written. At the moment, explicit support is provided for the ANSI standard languages PL/I and FORTRAN. Contact with the Multics operating system, then, is limited to their run-time packages and to the Substrate, which attempts to use only those parts of Multics that are most likely to remain stable, principally the "hardcore supervisor" and the I/O system.

To accomplish its second purpose, transfer to other machines, the Substrate does not make use of those features of Multics that are likely to be missing from other operating systems. Indeed a design goal was implementability on systems as diverse as Multics and OS/360. As a consequence, some features of Multics are denied to programs in the collection completely; others are provided by code within the Substrate itself. If the Substrate is implemented on another host, then the majority of programs in the Consistent System can run on that new machine. Naturally they would have to be recompiled; so the new machine would have to support the source language in which the programs are written -- PL/I and FORTRAN for the time being.

The third job, extending the services of the host operating system, is the most challenging. Eight major services are guaranteed to programs in the collection. Some of them are already provided by Multics, but not by other potential hosts.

(1) A file system is guaranteed wherein file names have two parts; one is the name of a directory, and the other is a name to be looked up in that directory.

(2) This file system is extended to cover all file devices, potentially including off-line bulk tape storage, for example.

(3) With each file name there is recorded a Description Scheme Code, which indicates the general type and format of the file.

(4) With each file name there are recorded locators for two separate sections of the file, and (optionally) an entry point to a program associated with the file; usually one section contains the data themselves and the other contains a description of the data and their format (e.g., a "machine-readable codebook").

(5) A "stacked" environment is provided so that programs can call other programs through the Substrate; the called program may call others, and so on, to almost any depth.

(6) A name-translation table is provided so that programs can control the meanings names will have to programs they call.

(7) A facility for trapping console I/O allows a program to divert output messages and input requests from a subordinate program to itself.

(8) A group of related programs may be declared to be a "family"; whenever a member of the family is called, the Substrate puts all files and working spaces back at the addresses where the last member of the family left them.

The implementation of the present Substrate was a major software effort by the Project staff. It is written in PL/I, comprises some eight thousand lines of source code, and includes some 54 external calls and an additional 69 internal routines. Once implementation began, changes to the design were gratifyingly few and the implementation schedule was met almost exactly.

<u>Conventions</u>.-- The conventions that programs and data are expected to follow are the second major piece of the framework. They can be divided into three main groups: conventions about data, conventions about the way programs are called and the effects they may have,

and conventions about transfer to other hosts.

The conventions about data center around the Description Schemes. Each file of data carries with it a Description Scheme Code (which is recorded with the name of the file in the directory). Before using a file, a program must check the Code to see whether the file is of a form it can handle, and when creating a file, a program must supply the correct Code for it.

If a contributor to the collection finds that none of the formats specified by existing schemes meets the needs of the programs he intends to write, he may define a new scheme, and the administrators of the collection will assign an arbitrary Code to it. That Code, and the specification of the new scheme, then become a permanent part of the documentation of the system. In theory there is nothing to keep the number of schemes from proliferating indefinitely; however, it will usually be easier for a programmer to use some scheme that already exists. We hope that that fact will tend to keep contributors from defining new schemes, or that social and administrative pressure will at least persuade them to provide programs that translate data to and from schemes that are already popular.

The conventions about programs are centered around the idea that a program should be able to act as the user's agent in running other programs for him. In general, they insist that programs be called in a uniform manner, and that each program be a neat, tidy module whose behavior an agent can control easily. In particular, they include such provisions as these:

When the Substrate runs a program (at the instigation of a Run Program call from an another program), it assumes the called program will have, paradoxically, the form of a PL/I subroutine with a particular form of calling sequence.

One of the arguments in the calling sequence is a string of characters that are "directions" to the called program. The called program is responsible for decoding these directions itself, and it must return an error-message to the user in some orderly way if they are unintelligible to it.

Normally, a program appeals to the name-translation table to discover the true meanings of the names of any files the directions tell it to inspect or create. This permits an agent to control the meanings that names will have to programs it runs for the user.

If a program sends messages to or asks for inputs from the user's terminal, it must do so through channels an agent program can intercept, so that the agent will be able to conduct the conversation in the user's stead.

Only very special programs are permitted to ask the Substrate to interfere with the program "stack" that was mentioned above as one of the services the Substrate guarantees.

Finally, the conventions that are designed to make the collection easier to transfer to other hosts prohibit the use of those features of PL/I and FORTRAN that are not likely to be available on other hosts. They also provide that only special programs may use certain "irregular" features of the Substrate that probably cannot be duplicated in precisely the same form on all hosts.

The conventions have been shaken down in the process of coding the utility subroutines and basic programs that will be mentioned below. They seem to be in reasonably good shape, although it is now evident that they need to be codified and reworded to make them easier for programmers to understand.

Utility subroutines and basic programs.-- As the actual calls to the Substrate are by design quite primitive, they are supplemented by utility subroutines which allow programming languages like FORTRAN to more easily call upon the Substrate for service. While these are of little immediate concern to the nonprogramming behavioral scientist who would simply use the system, they are of value to those who incidentally have programming skills and a desire to write Consistent System programs. Standardized utility subroutines make programs easier to write and ensure a moderate level of consistency in outward behavior and quality internally. They are also important from the point of view of economy: it is not nece-sary that individual programs contain the code needed to perform the functions these subroutines perform.
To date, a variety of utility subroutines have been provided which, among other things, parse the string of directions with which a program was invoked, resolve a symbolic file name, open several files at once, find a given element in an array, and allow a graceful return in the event of some catastrophe.

A small group of service programs has also been provided to let the user inspect definitions in his directory, inspect the stack of programs, end a session, and perform other essential actions of that sort.

More important, from the view of the nonprogramming behavioral scientist, is the way in which he can exercise a program from his console. There is a command processor or "basic caller" notable for its almost nonexistent syntax and relatively trivial organization - qualities which insure, respectively: freedom to design a command with nearly any syntax deemed appropriate by its author; and the ability to replace the "basic caller" with a more specialized command translator. Perhaps more interesting is a "macro-command" facility that allows a sequence of prespecified commands to be executed as a logical unit; the execution is interpretive, and meta-statements are provided to control the flow of execution and parameterize the inputs to individual commands.

The service programs, the basic caller, and the macro interpreter constitute a nucleus around which the library can grow. They are the programs a programmer needs to test others that he will contribute to collection.

Documentation.-- Because the Consistent System is being devised with long term use as a goal, documentation is of paramount importance. In time, an enormous number of programs will have to be used fluently by a large population of behavioral scientists, and the programs will have to be maintained by generations of programmers. Only self-sustaining documentation will keep these efforts less than Herculean. This past six months has seen a review of the documentation needs of the project and its user community, and planning for their fulfillment. The experience accumulated in documenting the Toss system was helpful in this regard; details about it are to be found in Section 6.2 of the second Annual Report.

Our plans call for two general categories of documents: tutorials for a general understanding, and reference manuals for specific information about a particular routine or file of data. The tutorial manuals will be relatively static once produced, but the reference documents will continually accumulate new sheets for insertion as new contributions come into the system.

The tutorial category can be further subdivided into a general introduction for users of the Consistent System, introductions for particular classes of users or about particular subsystems, and a programmer's introduction to the Consistent System.

The reference documentation will tend to be more piecemeal, more concerned with individual programs, subroutines, and sets of data. It will also be more uniform: there will be standard formats, and authors will be urged to stick to them if at all possible. Reference material will separate itself into two types -- that distributed widely, and that kept in archives at the Project headquarters to be distributed on demand, a copy at a time. The latter, less visible documentation is typically used for maintenance and modification. We assume that routines, data, and the documentation on them will be used much more often than they will be changed, and so it is reference documentation for use that is the more widely distributed.

Every name that appears in a public directory -- whether the name of a program, data-file, or combination thereof -- is represented by one or more sheets that comprise the user's reference manual. The purpose of these sheets is to describe the external appearance of the entity in standardized language understandable to the class of users for which it is intended. For programs, the descriptions must be exact enough to enable a user to use the programs appropriately in a "macro-command". Learning how to write the descriptions that go into this book is one of the most important challenges the Project faces. The description must both be complete and be comprehensible to the users for whom it is intended; when the users are not programmers in the traditional sense, satisfying both demands is difficult.

For those users who are programmers and want to write programs or subroutines for themselves or for the public collection, appropriate reference material, different from that described above, must be provided. It must include information on public subroutines, on Description Schemes that have been officially registered, on the behavior of the Substrate, and on conventions a program must follow if it is to be admitted into the public collection.

For the same audience there is an on-line register of external symbols. This register lists all external symbols used by each program in the public library, as well as all symbols that have been reserved for future use. As presently conceived, this register is interrogative and will provide, on demand, a symbol or symbols guaranteed not to conflict with those already in use or reserved for future use.

This register is, in part, an asset to good documentation practice; it is also the first in a class of control procedures that will ultimately rank with the Substrate, conventions, and documentation as a fourth part of the framework of the Consistent System. Additional procedures must be worked out for program acceptance and installation, quality assurance, and maintenance.

Growth of the library.-- Of course, the real power of the Consistent System for the nonprogramming user is in the spectrum of programs it makes available to him. By summer the collection should be rich enough so that a hardy researcher will be able to make practical use of it in some kinds of work. The programs that are to be included have been mentioned in previous sections and the following is only a brief overview.

Statistical procedures that are expected to be available include all the standard parametric and nonparametric hypothesis tests like t-tests and F-tests. Chi-square, tau, product-moment and rank-order correlations will also be available. A newly developed, interactive factor-analysis program is expected, as are the multiple linear regression and ANOVA (analysis of variance, with covariance added) programs which are now parts of DATA-TEXT.

Additional data analysis tools expected include the TSP (Time Series Processor) package developed for the econometrician, and two transaction flow analysis programs, "RANULL" and "DICHOT". Covering those instances where specifically designed tools are lacking, a set of matrix operators will be provided enabling a behavioral scientist to pursue more individualistic analytic methods. We expect the "macro-command" facility, described above, to make these pursuits easier. The principal tool for data-management and data-editing will be the prototype version of the Janus subsystem.

During the coming half year the library is expected to enter a period of rapid growth as programs from Toss are transferred to it, and as other programs that the participants in the Project have been preparing begin to accumulate within it.