AT&T Technical Journal
Vol. 64, No. 9, November 1985
Printed in U.S.A.

ANALYTICOL—An Analytical Computing Environment

By C. CHILDS* and C. R. MEACHAM†

(Manuscript received April 9, 1984)

A good workman is known by his tools.—Proverb

This paper is an overview of this special issue of the *AT&T Technical Journal* on ANALYTICOL, an analytical computing environment developed by AT&T and based on the *UNIX*™ operating system. ANALYTICOL was developed to provide specific capabilities of value to business analysts. The environment has the potential to provide substantial productivity gains and quality improvements for analysts working on ad hoc studies, and analysts and programmers developing applications. The other papers in this issue describe some of the individual software tools that make up the ANALYTICOL environment. This paper describes a set of generic tasks that need to be performed in solving business analysis problems, and how these tools can work together to perform the tasks and build an application system. An example of a typical business problem in the telecommunications industry is provided to illustrate the use of the tools in a business environment.

I. INTRODUCTION

In the early 1980s, the Bell System underwent a great deal of change, some directly under its control, some caused by external forces. The ability to respond rapidly to new business problems was of paramount importance. In response to this need, research was begun at AT&T Bell Laboratories on using the computer more effectively as a tool for business analysts performing modeling and financial studies. The

* AT&T Bell Laboratories. † AT&T Bell Laboratories; present affiliation Bell Communications Research Inc.

articles in this issue describe the individual tools that, when combined with the *UNIX* system, provide a highly flexible and evolving environment for such analytical studies.

In this overview we describe the nature of business analysis and its implications for analytical computing techniques. The scale of problems that are typically addressed by business analysts ranges from moderate to large, often with multimillion-byte data sets and equations with many variables. An example of a business problem, the modeling process that led to its solution, and the application program resulting from the work are provided to demonstrate the use of ANALYTICOL.

Analytical computing shares with many other computer applications a number of generic functions including interactive, form-oriented data entry and user-interaction management. As a result, many of the tools described here are in general use in a variety of applications. We describe how these tools are applied specifically to business analysis and the pros and cons of tightly integrated systems versus loosely integrated tools. Finally we describe the productivity gains that these tools provide when they are used as reusable modules for software application within the larger *UNIX* system development community.

II. A CHANGE IN APPROACH TO BUSINESS ANALYSIS AND SOFTWARE

AT&T Bell Laboratories and Bell Communications Research often address business problems that cannot easily be addressed by conventional approaches and therefore require development of new methods and techniques. New solutions to these problems are usually demonstrated through a prototype computer software implementation to determine their effectiveness. If the particular method is generic in nature, the successful prototypes are then given to other business analysts for experimentation. The prototype is then tailored to meet an individual company's needs, polished into business application software, and added to its repertoire of business tools. If the problem and solution are ad hoc and not expected to be readdressed, the results are reported and the understanding of capabilities needed to support such analysis is increased.

2.1 The old way

Many business analysts have limited computer experience. The conventional approach has been for an analyst to rely on programmers to obtain potentially useful data, manipulate it under the analyst's direction, and eventually code an analytic model. Or, the analyst programs in a traditional language, such as Fortran or COBOL, to develop models directly. Frequently model changes or "what-if" modeling studies are difficult and time-consuming to implement in the typical "waterfall" model of software development. (For example, it has taken over six months for some tax changes in depreciation to be included in one capital asset analysis program.) Even if the time frame is acceptable in a business sense, these resources can be justified only for software that has major effect on or repetitive use within a corporation. Exploratory interactive analysis or ad hoc studies cannot be done effectively in this type of environment.

2.2 The new way

New analytical application systems can be built far more productively by rapid prototyping with appropriate tools. If analysts are given an environment where they can directly use a computer and high-level languages—frequently referred to as fourth-generation languages—then many more exploratory and ad hoc studies can be undertaken, and understanding of the decision-making process can be increased. But how is it possible to bring about fundamental change in the way business analysts approach their tasks? First, it is necessary to create an alternative. We propose ANALYTICOL as an alternative.

The work of ANALYTICOL began when an expiring mainframe lease created the opportunity to develop a new analytical computing environment within an AT&T organization containing both business analysis and computer science research activities. Development began with a fundamental decision to use the *UNIX* operating system to support the environment. Another critical decision was to provide sufficient hardware, including high-quality graphics printers and terminals, high-speed communications, and necessary computing power in the form of super minicomputers and work stations networked together.

In the beginning stage of the research, the proximity of the computer scientists to a willing test population of business analysts provided early identification of system requirements and rapid information feedback. This contributed substantially to the selection and character of the tools and the quality and usability of the software.

III. CURRENT STATUS

Use of the ANALYTICOL system within AT&T is growing both through demonstration of the productivity gains by analysts and application developers and by word-of-mouth reports by enthusiastic observers. Each of the ANALYTICOL tools exists either as a working prototype or as a fully supported tool, and each has been effectively applied to several problems. The tools are implemented in the portable *UNIX* operating system environment and have been executed on a variety of hardware including work stations, minicomputers, and mainframes under *UNIX* System V, Releases 1 and 2, and the University of California Berkeley Releases 4.1 and 4.2 BSD. To use all of the tools effectively, a typical configuration has at least 2 megabytes of main memory and 20 megabytes of disc memory plus high-speed tape drives. Because of the full-screen nature of some of the terminal interfaces, 1200 baud or higher terminal access is recommended.

In addition to the basic ANALYTICOL environment that we describe here, many specialized analytical environments also exist that include ANALYTICOL tools, commercially available database managers and modeling systems, and data filters translating data formats between tools. Section V describes a specialized tool that was created with ANALYTICOL tools to analyze pricing options for telephone services in a marketing analytical environment.

IV. THE ANALYTICOL APPROACH

The computing model selected for this environment was a set of independent, high-level, efficient, and functionally specific tools addressing generic analytical tasks that could be loosely integrated into more specialized application packages. This model enhances the user's tool development many times over by creating reusable software. In the *UNIX* system environment, capabilities such as redirection, pipes, and the shell language facilitate this loose integration. Other *UNIX* system commands extend the environment, especially when ASCII flat files form the data interfaces. Furthermore, the *UNIX* operating system provides users with flexibility in the selection of terminals (Termcap/Terminfo) and computers. This is important because, as tool developers, we cannot foresee who our eventual users will be or what their hardware environments will be.

Determining functionally generic tasks common across a variety of ad hoc studies took the combined and concerted efforts of a group of computer scientists and a group of business analysts. It was the close interaction of these two groups that enabled the identification of a set of generic analytic tasks that could be addressed by a small set of tools which, in turn, effectively could be used to solve a large set of problems. The five generic tasks that were identified, and the tools that address them, are described in the rest of this section. Figure 1 shows the data and work flow among the tasks.

4.1 Data acquisition

The first step in exploratory data analysis is typically the acquisition and preparation of data. Usually data from many sources—often not computerized—must be integrated to provide the foundation for analysis. Examples of sources of data are previously compiled operations data about customer calling patterns, cost data for providing a service using new and existing technologies, demographic data from the cen-

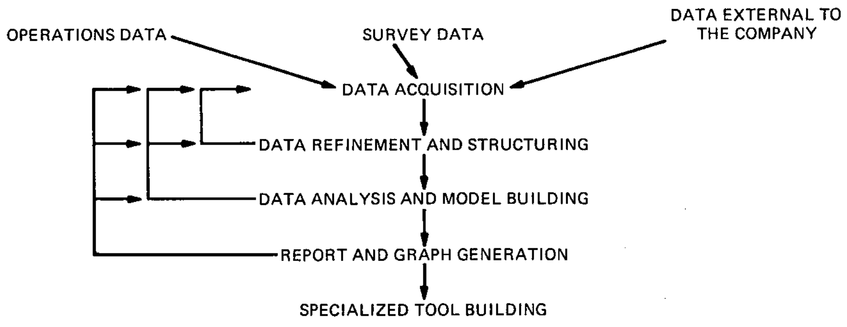
OPERATIONS DATA
SURVEY DATA
DATA EXTERNAL TO THE COMPANY
DATA ACQUISITION
DATA REFINEMENT AND STRUCTURING
DATA ANALYSIS AND MODEL BUILDING
REPORT AND GRAPH GENERATION
SPECIALIZED TOOL BUILDING

Fig. 1—Flow of an analytical project.

sus, and customer survey data on types of service selected. Two papers in this issue describe tools that provide this data-acquisition function. The paper by Belanger and Kintala describes two tools for data acquisition based on new program-generation techniques. These tools, TTU (Tape to *UNIX* system) and IMX (IMS to *UNIX* system), address the problem of transferring data residing in large mainframe databases to *UNIX* systems. TTU takes a description of an IBM-formatted tape and creates an efficient C language program that reads the selected records and files from that tape onto a *UNIX* system file. IMX takes a description of an IMS database and creates a COBOL program. This program can be transmitted by remote job entry or other methods (e.g., the *UNIX* system command `co`) to the system running IMS and then executed, with output data returned to the *UNIX* system environment by a route appropriate to its size.

The paper by Prichard explores the use of editors for forms entry and management. With FE (Form Editor) an analyst can paint a form and describe attributes of data to be collected, including intraform data validation and computation. For those more proficient in programming, fields can be filled by *UNIX* system commands (e.g., `date`) and user-supplied C language functions. When selecting FE to do data management, a file cabinet model is provided. Forms, as objects, can be manipulated by the user with a command language similar to that of the `ed` file editor for actions such as insert, append, move and delete. When a form is displayed, the `vi` full screen editor is the model for cursor positioning and data entry and update. Both features were designed to build on *UNIX* system editor training. A less powerful full screen editor is also available for users who do not know `vi`. For applications that need to interface with another data manager and need only the interactive form display and editing features, a C language library exists.

4.2 Data refinement and structuring

The *UNIX* operating system file system is adequate for much of the data storage and retrieval needs. All of the data-acquisition processes mentioned above provide flat *UNIX* system files (with the capability of building implicit hierarchies). White space and new lines are enough to meet minimal structuring needs, but some problems require hierarchical or relational structures and query capabilities. Two tools described by Swartwout and Yanofchick have been created within ANALYTICOL as experiments in database management aimed at analysis.

The paper by Swartwout describes Datastream, a simple language with expression handling, control structures, and built-in I/O. It makes it reasonable to query large (multiple megabyte) flat files interactively for data selection, refinement, and computation.

The paper by Yanofchick describes a hierarchical data manager, T. One of its most interesting capabilities is that a node can consist of a shell script. Access to such nodes can cause programs to be executed whose outputs appear as virtual data in the hierarchy.

4.3 Data analysis and model building

Data analysis and model building are the essence of the analyst's work. Such work requires a diverse and flexible set of functions and may include the interactive application of numerical or statistical procedures and exploratory graphical techniques. It may also include iterative development and solution of a series of equations—some of which may be simultaneous sets—that model financial or corporate problems. Often the analysis has multivariable equations. Two tools are available to meet these problems, S and HEQS.

The paper by Becker and Chambers describes S, a language and system for interactive data analysis. S provides a comprehensive set of statistical and graphical functions for interactive use, and mechanisms for users to extend its functional capabilities and language. Because its data structures are rich, yet easy to manipulate, it can also serve the data-refinement and structuring needs of many applications.

The paper by Derman and Sheppard describes HEQS (Hierarchical Equation Solver). This tool provides algebraic and logical expertise normally implicit in human model solving. Users can define, debug, change, and interactively solve nonprocedural models described as sets of linear and nonlinear multivariable equations. It provides for what-if, goal-seeking, impact analysis, and sensitivity analyses.

4.4 Report and graph generation

Reports and graphs are important not only to summarize and document the results of a project, but also for information updating during the analysis stage. A report describing the project and its conclusions must be produced efficiently and with high quality. Often it is a mixture of text, graphics, filled-in forms, and tables. Many tools support this need, including FE, S, D, and *UNIX* system commands (e.g., `mm` and `nroff`). D (Display system for reports and graphs) is part of ANALYTICOL and is built on the S graphics capabilities. It provides a nonprocedural language for describing tabular reports and business graphs (pies, bars, and two-dimensional graphs). Although its capabilities are similar to those of many Database Management Systems (DBMS) that create reports and graphs, its principal strength is that it is an independent tool and offers output flexibility advantages to applications and studies not otherwise using a DBMS. Since it represents commonly available technology, no further mention is included in this series.

4.5 Specialized tool building

If the data analysis and model do not address an ad hoc need (or the solution can address a large set of problems and therefore is no longer ad hoc), then the analyst has reached the point where the analysis and model solution are known and ready to be "packaged" into a specialized tool for use by others in the future. In other words, the requirements are now understood and could be documented, and application software should be created.

However, by using ANALYTICOL in the prototype stages, major pieces of the application already exist. They need to be integrated smoothly. A user interface needs to be created that hides the individual tools and intermediate steps and places the user in control of his/her work environment. The last paper, by Vo, addresses this problem.

The Interpretive Frame System (IFS) provides a block-structured, high-level programming language for specifying both the structural information of a system and the interactions between its various subprocesses, coprocesses, and human users. The application builder can create polished user interfaces (menus, question/answer, forms, help, and other dialogue) for assortments of tools from scripts and, thus, separate program function from user interface in a consistent way. To end users, the application appears as a single program allowing them to analyze their problem within an interactive structured environment. Some users have described it as an application-specific shell.

V. AN EXAMPLE

In this section, we give an example of using ANALYTICOL to solve a problem. To put things in perspective, we briefly describe what the analysts are trying to do, and then we discuss the ANALYTICOL tools that help in developing the model. Finally, we show how the applications developers can use the ANALYTICOL tools to build a system that contains this model for others to use.

One practical business problem is to estimate the profitability of services under alternative pricing scenarios. Analysts want to understand what will happen if alternative pricing plans are available to customers for certain types of telephone usage. In our example, the analysts investigate optional calling plans that feature discounts on intrastate or interstate direct-distance-dialed calls placed in the night or weekend period. A customer who elects an optional calling plan "buys" into the plan each month and pays for toll calls according to a discounted rate schedule.

To assess the profitability of the plan, the analysts model the demand response to the plan, as well as any potential changes in the calling behavior of all customers to whom the plan is initially offered. To do this, the analysts obtain pre- and post-usage measurements on the customers. Since this approach involves many usage measurements and large customer populations, the method of data management is an important consideration.

The analysts use two ANALYTICOL tools that are helpful in manipulating large amounts of data: TTU and Datastream. The TTU tape-reading facility can read at high speed the 35- to 80-megabyte data files that contain the monthly customer usage data collected on mainframes. TTU reduces the read time of this data to one-half hour from the roughly seven hours that other facilities would take. And it also provides checking, extracting, and summarizing of the raw data while it is being read from tape. The second tool the analysts choose is Datastream, which they use for more complex extraction, computation, and general clean-up for data residing in the very large flat files created by TTU. Datastream also is used to provide a link between individual customer-level information located on various data files. A master file is created, with one record per customer, of pointers to that customer's information in each of several data files.

S is used to generate graphs for graphical analysis to identify and analyze subscriber probabilities for the different calling plans. It helps determine empirical curves and to fit the curves with different functional forms, using both linear and nonlinear methods. This becomes the working model of customer choices.

The analysts then tailor the model for their use in predicting customer reactions to other alternative pricing options. For this they create an FE form on which to enter a pricing option. They add financial equations with the equations of the customer choice model to do revenue and profitability studies. HEQS is applied to the equations of the total model and the pricing options being studied in order to solve the profitability model. They use the what-if and goal-seeking capabilities of HEQS to explore other scenarios, and they use the sensitivity analysis capability of HEQS to study how customers might react to different pricing schemes and thus help predict the profitability of the different options. Finally, they document their results in reports and graphs using D.

As a result of their work, the analysts now have a customer-calling pre/post database, a customer-preference model, a form to enter pricing options, a model of the financial equations, and a model of the reports and graphic output displays. And, in the process, the analysts have developed a predictive model that can be tailored by an individual telecommunications company to meet its specific needs. What is needed is an application system to be used by other market analysts in studying future pricing options.

A key point is that the only additional programming needed to build this application system is the end-user interface and on-line assistance (help) and the reformatting of the data between FE, HEQS, D, and the database. IFS integrates the tools and data interfaces and provides the end-user menus and help messages; the data formatters are programmed in shell or C language. The only interactions an end user must make with the resulting application system are entering the pricing plan on a form and executing the tailored system by choosing appropriate paths through the system via IFS menus to meet his or her needs. The new optional pricing tool is now added to the tool set making up the marketing analytical environment.

VI. INTEGRATION, ARCHITECTURE, AND FUTURE DIRECTIONS

The computing model for ANALYTICOL is a set of independent, functionally specific tools that can be used as building blocks, with the *UNIX* system and IFS providing the glue to bring them together. This model requires that the analyst or application builder be able to convert data from one data format (output of x) to another (input of y), as described in the example in Section V. Taken to the extreme, if there are n tools, this model may require up to $n(n-1)$ data filters. Furthermore, the application builder must determine an architecture and sometimes create software for some of the manual steps the analysts may have taken. A tightly integrated ANALYTICOL system could avoid this.

To experiment with tight integration, S, HEQS, FE, and D were combined using IFS as a menu/help interface and the utilities in S for making HEQS, FE, IFS, and D appear as S functions. The tools themselves were not modified. IFS and S became communicating coprocesses and the S data structure became the common data structure by which all tools could communicate (i.e., since the tools were not modified, the model for *n* tools would require at most 2*n* filters). Obviously there was overhead, but the performance remained in the acceptable range for interactive studies. For only occasional S users, there was the added advantage of a menu interface to the more than 300 functions, plus the capability to learn and switch into the S command mode whenever desired. A small amount of FE functionality was lost since some form layouts cannot be described in S data structures.

Although some analysts like this integrated tool and have used it in studies and applications, it was not as popular as expected. The S data model is not a natural one for thinking about forms or equation modeling. Unless all the functions are required for a problem, many analysts prefer loose integration, using the tools independently and specializing their data filters to meet their singular needs. Therefore, the experiment has met only limited success. A new experiment in tight integration is under way to define and build the tools around a new common data structure that fits this set of the generic tasks in a more natural way.

Furthermore, while greater integration may be preferred for some analytical problems, experience shows that many users use only one or two tools for any specific problem. Generic filters that bridge tools commonly-used together in the extended *UNIX* system environment have been developed by some users, and are beginning to be shared and added into the ANALYTICOL environment. This is at least a short-term solution to redundant filters and to further reduce the need for programming. It has the advantage of keeping the environment open to new tools developed by anyone, anywhere. It is therefore more natural to the *UNIX* system.

The architecture of a project can have a major effect on its ability to use tools as components. One architecture that works particularly well is that of independent processes whose executions are controlled by a common process. Communication passes dynamically between each process and the control process and through shared data files. The order of execution of the application's functions is controlled directly by the end user responding to either results of previous functions or external needs. This is the architecture of the example in Section V, and the one for which IFS was designed.

The ANALYTICOL environment will continue to evolve. In addition to exploring questions about integration and architecture, work is needed to standardize common elements among tools, such as error handling, style of error messages, common commands usage, and common editing techniques. Other future work will extend the environment to include new tools and emerging technology such as database machines, distributive computing, advanced work stations, bitmap graphics, and user interfaces, through pointing devices such as mice, fingers, and light pens.

VII. EFFECT ON PRODUCTIVITY AND QUALITY

We have described a process of analytical computing, and an environment of software tools that were created to provide for more effective use of the computer and to increase the productivity of analysts involved in business and financial studies. But, how do you measure increases in productivity?

Some statistical measures exist that can be applied to software, such as lines of code per programmer months. But since the languages are very high level (more power per line) and applications are done in less time, this measure is not very useful. The measurement of errors per lines of code can show increased quality for an application when the tools' source code is included in the denominator, since the tools have been used extensively and introduce fewer errors in the numerator. But statistical measures do not seem to capture the benefits effectively. Since there have been no controlled studies, we can only offer the following observations.

For two analytical application systems, resources were estimated for the old way and compared to resources consumed in the new way. Details of the applications resources leading to these conclusions are provided in Table I. In both cases the productivity gain—about a factor of 5—led both projects to conclude the benefits were substantial. (The appearance of the value 5 in both applications should be considered only coincidence. The similarity of lines of code per staff-month for either approach within each application is indicative of the application complexity.)

In a recent survey of applications developers using FE, IFS, and D—these three tools offer more generic functions and have had

Table I—Experiences of two analytical applications

	Old Way (After Req. Known)	New Way (Evolving Req.)	Productivity Factor for Staff Months
Application 1. Supporting Rate Planning			
Linear months	7	1.5	
Staff months	21	4.5	>5
Lines of code*	50,000	10,000	
Lines/staff month	2,333	2,222	
Application 2. Financial Analysis of Business Plans			
Linear months	8	2	
Staff months	32	6	>5
Lines of code*	10,000	2,600	
Lines/staff month	313	433	

* These figures only include code written specifically for each application.

significant use outside analytical computing—85 percent showed major productivity gains and 40 percent considered these tools critical to the success of their projects.

Benefits not directly associated with the code development process are even more difficult to quantify. Fringe benefits, such as real-time validation of data entry that would not otherwise have been provided, have contributed to improved user interfaces and decreased user error. Such features were not a project requirement and did not enter into the benefit analysis leading to the selection of the tools. However, both types of benefits have improved the quality of the application and have led to increased user satisfaction.

A tariff for a new service was based on a study of five alternatives. To what extent did the study, which would not have been done without the tools, contribute to a better decision? This type of benefit is very difficult to quantify!

We feel that the environment of ANALYTICOL has been successful in increasing productivity of business analysts and application developers. We base this conclusion primarily on their acceptance and use of the tools. We invite you to read the next seven articles in this journal, which describe the generic tasks and features of each tool in more detail, and reflect on our conclusion.

VIII. ACKNOWLEDGMENTS

While we are grateful to many people who have contributed to ANALYTICOL and this paper, we especially want to thank J. P. Downs for taking the risks and providing the leadership and resources, and Lynn Grala for sharing her experiences in the rate studies example.

AUTHORS

Carolyn Childs, B.S. 1965, M.S. 1966, (Mathematics), University of Massachusetts; M.S. (Computer Science), 1974, University of Wisconsin-Madison; Instructor of Mathematics, 1966–1975; Assistant Professor, 1976; University of Wisconsin Center System, Waukesha; AT&T Bell Laboratories, 1976—. Ms. Childs is currently Supervisor, Common Software Tools Group. At AT&T Bell Laboratories she has contributed to methods and systems supporting analysis for project evaluation and economic impact of new technology. Her involvement in the creation and support of the analytical computer environment reflects her research interests in reusable software, tools and development environments. Member Phi Beta Kappa, Phi Kappa Phi, IEEE Computer Society.

C. Rebecca Meacham, B.S., (Mathematics), 1969, Millsaps College, Jackson, Mississippi; M.S. (Mathematics), 1973, The University of Mississippi; M.S. (Computer Science), 1979, The University of Tennessee; Mathematics Instructor 1969–1979; Member of Technical Staff at AT&T Bell Laboratories, 1979–

1983; Bell Communications Research, 1983—. Ms. Meacham is currently a Member of Technical Staff in the Analytical Computing Systems District at Bell Communications Research. Upon joining AT&T Bell Laboratories in 1979, she helped create a software development environment in which analysts can create and experiment with new modeling techniques, participated in the experiment that fully integrated the ANALYTICOL tools, and developed several prototype application systems using ANALYTICOL tools.