THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 62, No. 7, September 1983
Printed in U.S.A.

Total Network Data System:

Central Office Equipment Reports for Stored Program Control Systems

By R. F. GRANTGES,* V. L. FAHRMANN,* T. A. GIBSON,* and L. M. BROWN*

(Manuscript received February 16, 1983)

Stored Program Control System Central Office Equipment Reports (SPCS COER) is one of the earliest of the family of Operations Systems. Deployed on a centralized time-shared system, it produces engineering and administrative reports for Stored Program Central Office Equipment. This paper describes the user needs that motivated the design of SPCS COER. It traces the development of the system from a joint experiment of Bell Laboratories and the New York Telephone Company's Manhattan Engineering Department to the present system serving over 2800 electronic central offices from three centralized Amdahl V6 computers. We explain the functional system design from the user's point of view and outline the structure of the software, emphasizing the "tools" approach that is central to the development. The project management featured a research and development style, which we call the "whole-job" approach and discuss in some detail. Finally, we explore possible future directions for the project.

I. INTRODUCTION AND HISTORICAL PERSPECTIVE

When the first 1 *ESS*† electronic central office cut over into service on May 30, 1965, at Succasunna, New Jersey, the practice of traffic

* Bell Laboratories.
† Trademark of Western Electric.

data collection took a large step forward. Peg count, overflow, and usage measurements were made by the stored program control of the switching machine and kept in memory until their scheduled printout on a remotely located traffic teletypewriter.[1] No longer was there any need for the old electromechanical traffic registers, for clerks to read them, or for cameras to photograph them. Since the traffic teletypewriter could easily perforate a machine-readable paper tape with the measurement data, the tapes could be collected and read into a computer whenever convenient. It seemed clear to AT&T and Bell Laboratories planners that telephone company written computer programs would soon be processing the measurements into statistical information that would be unprecedented in accuracy, reliability, and timeliness.

1.1 Traffic data needs

Central office traffic measurements tell a Bell Operating Company (BOC) what is going on in the switch and the network. The company needs to know about call load, customer service, and equipment utilization. Information needs vary from short to long term. Network administration people are responsible for ensuring that valid, complete, and timely data are properly collected and reported for administration, business, engineering, maintenance, and other purposes. If something is wrong with the data, or the switching office, the administrators need to know quickly so they can fix it quickly, thus minimizing the effect of the problem on data and service quality.

Network administrators use the data to trend and balance loads and determine exhaust dates for central office equipment components (i.e., dates when the expected call loads will exceed the traffic capacity of the installed equipment). Network design personnel have more long-term needs, using the data to determine equipment capacities and the size and timing of future office additions. Marketing people help determine how well customers' needs are met by existing equipment arrangements.

Because telephone use varies widely (even wildly) with time, most data needs are best served by statistical information (e.g., averages) rather than by individual measurement readings. This means arithmetic operations (data processing) must be done on the measurements collected. It also means that—unlike, for example, the banking industry—a few missing measurements may not be serious. Some data may need to be excluded from the averages to keep the results representative. A good example of this was the day it snowed in Miami in 1977. Or the day a telephone building caught fire in New York City. Or the time when an equipment trouble caused all calls to certain areas to be set up so that no one could talk. Thus, data cannot be accepted

uncritically but must pass the test, "Is this measurement both accurate and representative?"

To meet these needs in the years before electronic switching had been difficult and labor intensive. Data systems were semiautomatic at best. For example, measurement registers (mechanical counters that resemble automobile odometers) could be grouped together in large arrays and photographed periodically by automatic cameras. But, after development, the film had to be manually read and keypunched before it could be used by data processing machines. It was particularly difficult to obtain timely information in quantity. Thus, the electronic switching systems' promise of an economical, completely mechanized collection and data processing capability was truly an important and exciting step forward. Increased mechanization was going to permit improved information quality and reduced processing time and save a lot of manual effort as well.

The expected increase in mechanization was not intended to permit force reduction but to prevent a large force increase. Electronic switching systems were going to produce a lot more traffic measurements than previous electromechanical systems. This was because of the increased complexity of the switch (requiring new measurements), the ease and low cost of making the measurements, and the plans of the Bell System to offer new and special services using the capabilities of the stored program control. Indeed, within 10 years a typical 1 *ESS* central office was estimated to be producing perhaps 18,000 register readings daily or some 4.5 million per year. As the Queen said in Lewis Carroll's *Through the Looking Glass,* it was going to take "... all the running you can do, to keep in the same place."

1.2 An experiment begins

By 1969, little of the expected telephone company development had taken place. A few companies were running programs on time-shared computers that rearranged the traffic measurements into neatly labeled columns. Many companies were using transparent overlays to locate a few key measurements in the teletype printout, and were then recording and processing those manually. Some 50 *ESS* central offices were in service and new offices were being installed at an increasing rate that would soon reach one a week.

In mid-1969, work to design an experimental traffic data processing system for 1 *ESS* offices was begun as a joint project of Bell Laboratories Traffic Studies Center and the Manhattan Engineering Department of the New York Telephone Company under the auspices of the Traffic Division of AT&T. The system was experimental because, from the outset, the intention was to use technology radically new for the time in an attempt to build a practical and economical data system

uniquely responsive to the needs of the telephone company users. Those needs, of course, were incompletely understood at the time, a further reason for considering the system experimental.

After two years of initial development, overlapped with about a year of trial use in several companies, AT&T in October 1971 announced a new time-shared computer program to process and manage traffic measurement data in 1 *ESS* offices. The new system, "PATROL" (Program for Administrative Traffic Reports On Line), provided an uncommon combination of features of significant help to the network administrator and network design engineer in the management of data. The announced features included the following:

1. On-line access to large quantities of traffic data.
2. Daily, weekly, monthly, high day, and busy season reports on demand in nearly real time.*
3. Automatic selection of high-day data.
4. Built-in data validation and exception flagging capabilities.
5. Single end-user control of traffic data entry, data retention, report generation, and data management in general.
6. A forgiving, interactive user dialogue that makes it easy to use with minimum computer knowledge and little or no formal training.

(Later experience proved two more features important:)

7. Reliable operation with rapid, low-cost system maintenance and feature development.
8. No hardware investment required of the user community for the experimental system that used only widely available teletypewriters with paper tape readers.

PATROL was initially implemented in a combination of FORTRAN and EXEC language on an IBM 360/67 computer operating under a modified Cambridge monitor. The vendor, Computer Software Systems, Inc. (CSSI) of Stamford, Connecticut, was chosen for the unique capability of its service at the time. There were no plans to make PATROL portable.

1.3 The first system

After initializing the system with a detailed description of a particular switching system, a network administrator would dial up the CSSI machine over the regular telephone network and transmit hourly ("Block H") traffic register counts from the paper tape punched by the traffic teletypewriter. PATROL would immediately make data validation tests, and items exceeding predetermined tolerance limits would be called (flagged) to the administrator's attention, along with

* In this context, "near-real time" refers to reports obtainable with little delay (perhaps minutes) after the end of the period being measured.

a brief summary of office performance data. The measurements would be accumulated in history files of data on an hourly basis. As many hours of daily data as desired could be processed, with costs increasing in proportion. Because of cost, most offices processed only one or two hours of data a day.

The history files were arranged for the generation of reports on a daily, weekly, and monthly basis, or for any desired span of dates at user request for each individual switching office. The reports were usually printed off-line and mailed to the administrator, but when time value justified higher cost, the reports could be printed on-line at the administrator's (or network design engineer's) terminal.

Probably little in PATROL was startling or new from a computer science point of view (a term just then beginning to come into vogue). What was new was the application of the technology to assure maximum capability and ease of use for end users with little training. A single time-shared machine serving thousands of offices spread over a continent was a concept still considered a bit adventurous (perhaps even foolhardy) in the early 1970s, although there were several precedents in the 1960s.[2,3] An on-line, interactive development facility was still a wonderful novelty for program designers and, of course, proved efficient and cost effective.

One design approach that was not recognized as new at the time was to provide each switching office with a database distinct from every other office and to conduct almost all processing as though only the one office and the one administrator existed. The (perhaps concurrent) processing of other offices for other administrators was generally hidden in the design, it being left to the operating system to sort out. This single-office approach greatly simplified development and maintenance by reducing the complexity of the software compared to other Total Network Data Systems (TNDSs) [e.g., Traffic Data Administration System (TDAS) and No. 5 Crossbar Central Office Equipment Reports (5XB COER) described elsewhere in this issue] whose designers had to plan for the sequential processing of perhaps hundreds of different offices in the same batch run.

1.4 Growth and evolution

In the years that followed the initial introduction of PATROL, most of the original features were preserved, while many features and systems for other types of electronic switching systems were added. At the end of 1971, 27 offices in 9 companies were using PATROL.

During 1972, the participation of the New York Telephone Company was gradually phased out and Bell Laboratories continued the experiment, working closely with AT&T and representatives of the BOC

network administrators.* Many report types were added, an on-line documentation feature, Lessons, was produced, and processing costs were reduced. By March 1973, there were more than 165 offices on the system.[4] In subsequent years, versions of PATROL were developed for 2 *ESS*, 3 *ESS*, Remote Switching Systems (RSS), Voice Storage System (VSS), and 5 *ESS* machines.[5]

When an in-house (AT&T) computer service offering nearly the same software features [Virtual Machine/Conversational Monitor System (VM/CMS)] as our original vendor became available in 1974, PATROL was the first large project to move to that facility. It has remained there since.

In 1975, when the Engineering and Administration Data Acquisition System/Traffic Data Administration System (EADAS/TDAS)† data collection portion of TNDS was available in most BOCs, PATROL developed a major new feature called Batch Data Entry (BDE). Data could be collected by EADAS, sent by magnetic tape to TDAS, and relayed from TDAS to the PATROL computer over the high-speed corporate data network (see Fig. 1). Compared to paper tape data entry, BDE significantly reduced computer expense at the sacrifice of two initial PATROL features. A single end user could no longer control the whole system alone, as coordination with EADAS, TDAS, and the corporate data network was required. And, because of the delays inherent in the new batch process, results from PATROL could no longer be obtained in "near-real time." However, because of the data processing and reporting capability of EADAS, the near-real-time feature of PATROL was no longer needed where BDE could be used.

In 1977, a major rewrite of PATROL changed its basic orientation from processing clock hours for all office components to processing selected busy and side hours for each office component. This component busy hour feature was followed in 1979 by a Busy Hour Determination System. By the end of 1978, there were more than 1600 offices using PATROL.

1.5 The experiment ends

During its early years PATROL was enthusiastically received by network administrators and network design engineers. However, upper BOC management, often advised by computer experts that time shar-

---

* As has often been our experience, the collaboration with a BOC, here New York Telephone, had been richly rewarding. The "real world" expertise and "get it done now" attitude of the New York people really helped move the project along. By the end of 1972, however, the on-line system was providing feedback from the whole country.

† EADAS is the subject of the article "Data Acquisition and Near-Real-Time Surveillance," this issue.

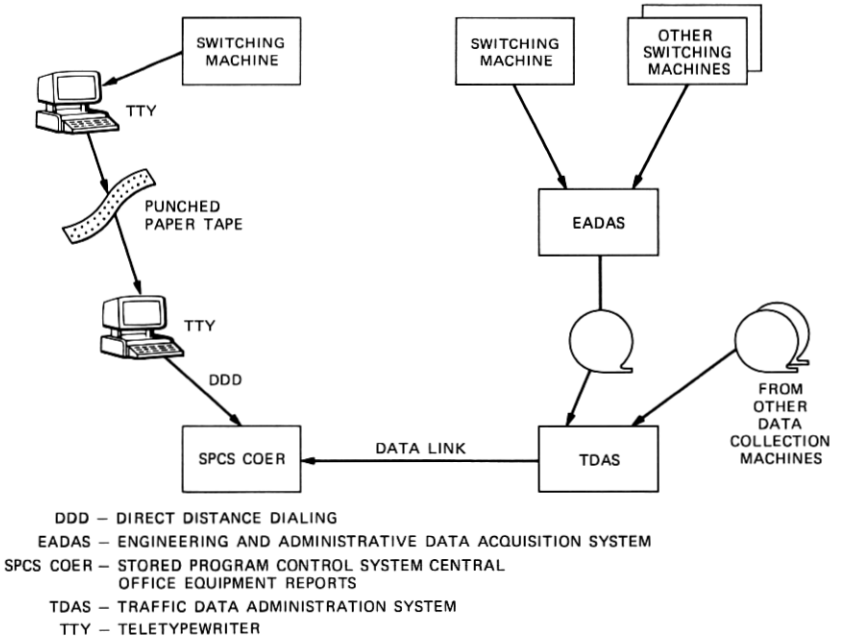


Fig. 1—On-line and batch entry of data in SPCS COER.

ing was always more expensive than batch, feared that perhaps the system was too expensive. In 1976, these fears were laid to rest by two events. A major survey of computing costs showed that batch operating costs were much higher than was widely believed—sometimes 10 times higher. Finally, an extensive multilevel, multidepartment survey of every BOC, including three that at that time were building or operating their own traffic data systems, was conducted. It found that the PATROL system, of all other candidates, had the best match of features to the future data needs of the BOCs. With these and other studies concluded, it was finally clear that the PATROL experiment could be considered at an end. Continuing standard development under the Business Information Systems (BIS) funding agreement was begun in January 1978.

1.6 Continuing development

The BIS Advisory Committee asked that the name of the system be changed from PATROL to *ESS* COER to better agree with other TNDS component system names. This was later amended to SPCS COER when we realized that the system would be used for Stored Program Control System Reports not involving switching. However, because the term PATROL had become extensively embedded in Bell

System literature and had been used by the BOCs for many years, and because two syllables are much easier to say than six,* the term PATROL is still frequently used by the field forces.

The change from experimental to standard status made it possible to emphasize system design considerations in SPCS COER over simple feature availability because long-term planning and investment of resources were now warranted. The original version of PATROL, with substantial amounts of interpretive code, had been replaced by a much more efficient FORTRAN version by the end of 1972. As an experimental system, the software emphasis was on achieving features quickly, while steering a prudent course between run-time inefficiency and high performance. With the prospect of an assured future, the development effort began to emphasize new approaches to the design in 1977. A modular, generic-tools approach would achieve both high performance and reduce development and maintenance effort in the long term. This new approach is described in Section III.

In 1979, a new experimental feature was designed for AT&T. The Data Profiles feature consolidates selected information from all the individual 1/1A *ESS* office databases for the whole Bell System. Elaborate retrieval, computation, and reporting facilities are provided in an English-like language for the use of personnel engaged in studies of various types.

During the 1980–81 busy season (January to March), SPCS COER began to strain the resources of its host (Amdahl V6) computer, with the traffic data load from nearly 2400 *ESS* machines. Therefore, in September 1981, after about a decade of continuous operation on a single computer, the user community was divided up among three computers (at the same location).

In December 1982 a new feature was introduced that allows printing of reports in the user's local company data center. In addition to existing options that allow reports to be either printed on-line or off-line and mailed, the user may now have the reports sent to the nearest BOC data center over the corporate data network or via magnetic tape. This new option was provided to allow users willing to incur additional delay in the distribution of reports the opportunity to reduce centrally billed expense for SPCS COER by as much as one third.

During its first decade of operation, SPCS COER compiled an excellent record of availability. While few detailed records were kept, the longest recorded nonscheduled total system outage (no access to reports) was about six hours. Some individual users or BOCs experienced longer outages while software errors were being repaired.

At the end of 1982, there were more than 2800 offices using SPCS

* The preferred pronunciation of SPCS COER is S-P-C-S KO-AIR.

COER. The program had grown to 225,000 lines of FORTRAN code with an additional 105,000 lines of comment. There were about 1000 pages of on-line user lessons for all the component systems. Users were logging on to the system at a rate of about 25,000 per month and issuing about 1,000,000 commands a year. About 3,000,000 hours of data were being handled annually and 1.2 million pages of reports produced per month. One third of the report output is produced on fiche, the rest on paper.

II. FUNCTIONAL OVERVIEW

This section presents a functional overview of SPCS COER from the user's point of view.

SPCS COER is really a family of on-line data management systems that process traffic data from the class of switching systems called sorted program control systems. This includes 1/1A *ESS*, 2/2B *ESS*, 3 *ESS*, 5 *ESS* and RSS switching entities.

Each switching entity, or office, served by SPCS COER has its own distinct, independently managed database. This implies separate data collection, reporting, and data management for individual offices. This single-office view is incorporated in the software design and the user interface. It reflects the usual approach taken by network administrators and network designers in examining central office traffic data. A consequence of this design is that most of the SPCS COER systems cannot summarize data across multiple entities. There is, however, one SPCS COER system, described in Section 2.6, that provides data summaries across offices.

2.1 User interface and documentation

A paramount goal in designing the SPCS COER systems is that they be easy to use by network administration clerks as well as managers. An interactive English-like dialogue, on-line documentation, and a generally forgiving design help support this goal. Since SPCS COER is an on-line system, it gives the user direct control of the system functions. When something unexpected happens, for example, an invalid request is submitted, the user is notified and given an opportunity to take immediate corrective action.

The user interface takes the form of a question and answer type of dialogue. The user types a command in response to a REQUEST = prompter. Any parameters required before the command can be executed are then prompted for by the system. The user need type only a carriage return to ask for help in responding to a particular prompter. The system will then print a list of valid responses or response formats. This list is also printed if the user types an invalid reply. The user can type CLEAR in response to any prompter to abort the command and return to REQUEST =.

A complete set of on-line user documentation is available with each SPCS COER system. The user may access the documentation on request and have one or more sections of it typed at the terminal or printed off-line and sent by mail. This allows changes in the documentation to be made available to the users quickly and easily by the developers. It also assures the user of an easily accessible source of the latest information. The combination of readily changeable on-line documentation with a hot line open 24 hours a day (by answering machine at night) has proven effective in keeping the documentation error free and the users informed, confident, and satisfied. One finding of many studies is that software system problems are often due to the unavailability of up-to-date documentation at the user sites. SPCS COER totally avoids this problem by placing the most current information in the hands of the user whenever the user asks.

News messages may be printed on request as well. News items are put on the system by the developers to announce new features, revised documentation, or the existence of (or, preferably, the solutions to) newly discovered common problems. Users are alerted to news items by one-line flashes whenever they log on the system. Two levels of flashes and news items are used depending on the intended audience for the item—all users or only users of a particular component system.

Since costs of any operations support system need to be monitored, SPCS COER provides estimated expenditures to the user for all on- and off-line requests. This feedback has helped users manage their use of the system effectively.

An important part of the SPCS COER user interface is a network of "SPCS COER coordinators," one or more in each BOC. These coordinators directly assist the BOC users in operating and managing the SPCS COER system. Usually it is the company coordinator who contacts the SPCS COER staff over the hot line about problems. The coordinators help the users identify and rectify problems. Over the years, the coordinators have asked for and been given the tools to identify and correct system problems that initially only the PATROL central staff could fix.* These tools have been grouped into a separate interactive system, modeled on SPCS COER, called COMP for Coordinators' Management Package. COMP even has its own set of on-line lessons.

2.2 Office description file management

Before any SPCS COER system can process traffic data for an office, an Office Description File (ODF) must be established for that

* That is, when the generally forgiving design had stopped being forgiving to some user.

office. The ODF contains information describing the office and the traffic measurements that are to be processed. This information includes:

1. The size of the office
2. The equipment or service components of the office
3. Characteristics of each component
4. Measurements to be processed and components to which they correspond
5. Hours of traffic data to be processed for each component.

An SPCS COER user establishes, or activates, an ODF through an on-line request. The user must supply to COER all the information needed for the ODF. This information is used by the system to interpret, validate, and make calculations on the traffic input data and to store and retrieve processed data. Therefore, it is essential that the ODF be current and correct.

A validate function ensures that the ODF information is internally consistent and that the user-specified values fall within appropriate bounds (e.g., the information represents a valid *ESS* office configuration). Validation is automatically done when an ODF is first activated and whenever the information in it is updated. If errors are found in the ODF, the user is alerted and COER will not process any new traffic data until the errors are corrected.

The on-line update command allows the user to examine the ODF and change it. All changes are validated, and any problems immediately reported to the user on-line. The user can then fix any errors by issuing update commands. Commands are also available to print the contents of an ODF on-line or off-line, and to deactivate an ODF (remove the ODF and its associated database from the system).

2.3 Data entry

To enter data into the database for an office, an SPCS COER system must first read the raw data values put out by the SPCS on what is known as a traffic schedule. A traffic schedule contains measurement data for a particular office, the collection date, collection interval, and end-of-interval time. The system uses ODF information to associate measurement values with their corresponding components. Format checks are made on the values, and traffic statistics are calculated for each component based on the raw data values and ODF information. These processed component values are then stored in the database for the office.

As part of the data entry process, SPCS COER makes various data reliability tests. These tests are intended to pinpoint instances of traffic measurements that are invalid, for whatever reason. Invalid measurements are caused by equipment or software malfunctions in

the switching machine or data collection system, highly unusual and unrepresentative traffic conditions, or (most of the time) errors in the ODF information. If data for a component fail one or more reliability tests, they are flagged, or marked as questionable, in the database. Flagged data are processed specially during data retrieval. These data are marked on all reports on which they appear and are excluded from report averages. The user has a recourse if data are inappropriately flagged, as described in Section 2.5. The results of the data reliability tests made during data entry are saved and made available to the user as exception reports.

Traffic data may be entered into SPCS COER in one of two modes, on-line or batch, as shown in Fig. 1. On-line data entry, the original PATROL data entry mode, involves using punched paper tape generated at the traffic teletypewriter connected to the switch. These tapes, containing the required traffic measurements, are transmitted to SPCS COER on-line over the switched telephone network via a terminal with a paper tape reading facility. The user must monitor the process while the tapes are being read. For large volumes of data, this is a tedious and time-consuming process that also involves substantial computer connect time charges.

The newer batch data entry facility reduces the cost and clerical effort associated with on-line data entry. It is now used for most of the offices served by SPCS COER. In designing BDE, the idea was that a user could set things up once, and then data would flow from the switch into SPCS COER almost unattended.

Batch data entry may be used by offices supported by EADAS and TDAS,[6] or their equivalents. Under this method of data entry, data for many offices and for several days are transmitted from a telephone company TDAS computer site to the SPCS COER computer site over the corporate data network. The SPCS COER system automatically places this data into a common disk storage area as pending files. An individual pending file is created for each block of data that corresponds to a particular office, date, and collection time. These pending files are organized on a per-user basis.

At night, an automatically scheduled job is run for each user who has pending files. This job processes the data from the pending files into the databases for the user's individual offices. It also generates messages for the user that indicate the status of the batch data entry job. These messages may be printed on request when the user next logs on to the system. In this way, a user knows exactly what data were entered into the database, or why any data were not entered.

Pending files that are not successfully processed are retained on the common disk area for two weeks. They are also available for 45 days from a backup tape. The user can obtain a listing of all pending files

for a given office. Users are provided with on-line commands for managing these pending files. Pending files can be erased from the common area or restored from backup tape. Certain errors in the header information for pending files or in the traffic data values can be corrected interactively to enable the system to process those data.

2.4 Reports

To retrieve data from an office database, a user requests traffic reports from the system on-line. The user has the option of having them run on-line and typed directly at the terminal, or run in a batch mode, printed off-line, and mailed to the user.

There are two basic types of traffic reports produced by the SPCS COER component systems: daily reports, which produce individual daily data and data averaged over a given span of days; and Machine Load and Service Summary (MLSS) reports, which make available year-to-date summaries of monthly averages and the highest traffic days.

Daily reports may be requested in weekly, monthly, or intermediate form. Weekly and monthly reports provide data for all central office equipment components and all hours. Intermediate reports may cover any span of days and any selected components and hours. Each section of a daily report is devoted to one component in the office. It shows the daily statistics as well as the averages of the values over the report period. Flagged data are excluded from the averages. Monthly reports are useful to the network administrator for analyzing the overall service in the office, for trending office loads, and for data validation (as described in the next section).

The MLSS reports are used by the network designer in capacity determination and in planning for future central office equipment needs. An MLSS report contains, for each component and hour requested, data for the 15 high load days of the year to date, together with the average of the 10 highest unflagged days. Also included are the monthly averages for the year to date, together with the average of the three highest months. At the end of the data collection year, this last average represents the Average Busy Season (ABS) value for the component.

The formats of the traffic reports produced by the SPCS COER component systems follow those produced by the 5XB COER system wherever possible. The 5XB COER reports are covered in some detail in the article "Equipment Systems," in this issue.

The Busy Hour Determination (BHD) systems (1 *ESS* BHD and 2 *ESS* BHD) of SPCS COER produce a different set of reports from the ones just described. The BHD systems help the network administrator determine, for each component in an office, which hour has,

on average, the highest traffic load. This is called the time-consistent busy hour for the component. A BHD study for an office is usually done once or twice a year, for two to four weeks. During that time, data are collected and processed by the BHD systems for 12 to 24 hours per day. The output of this study helps the network administrator determine which hours of data should be processed for each office component in 1 *ESS* COER or 2 *ESS* COER. There are five sets of reports produced by the BHD systems. They range from detailed hourly reports to a high-level summary report listing the suggested COER collection hours. Because RSS and 5 *ESS* switches are Extreme Value Engineered (EVE),[7] instead of time-consistent busy hour (or average busy season busy hour) engineered, they do not require busy hour determination studies.

2.5 Database management

SPCS COER makes available to the user several data management capabilities. There are on-line requests that tell the user what data are available in the database and allow the user to flag and unflag data, remove unwanted data, and initiate (or terminate) a data collection year.

The on-line flag and unflag requests allow the user to override the data reliability flagging decisions that were automatically made by the system during data entry. A user can selectively flag or unflag data for any component, hour, and date. This is usually done before requesting a monthly report.

After a monthly report is run, the monthly averages are stored with the MLSS data, and users have received their paper or microfiche reports, there is usually no need for SPCS COER to keep the detailed daily data for that month. The SPCS COER has a remove command that lets the user remove daily data for any span of days from the database. Data can also be removed selectively from the MLSS high day lists. A cycle function lets the user remove all MLSS data from the database at the end of the year.

The database management features, along with ODF validation, are key parts of the success SPCS COER has achieved in producing high-quality results. However, the degree to which user actions are required to manage the database might seem inefficient. Why not further mechanize these data management processes to require less user intervention? Let us look at why this user control is so important.

The user, usually a network administration clerk, is responsible for producing accurate and timely information. But it appears to be a fact of life (based on some 20 years of data processing experience) that bad data often does not become apparent until after it is used to produce reports. And every new type of report produced seems to uncover new instances of bad, or at least suspicious, data.

When this happens in SPCS COER, the user, after suitable investigation to learn the facts, simply flags or unflags data as appropriate and reruns the report. Only after the reports have been thoroughly examined is it generally safe to remove the data from the system. Data processing systems without these features often leave their users with no usable results and/or the task of reprocessing the data by hand.

It is also a fact of life that users make mistakes. While this may be regrettable, it is no excuse for producing poor results. Thus, almost every action that a user can take in data management can be reversed if the action proves to have been a mistake.

User control of the data management has a cost to the user in data processing and storage charges, of course, But if the user feels that exercising the recovery features is not worth the cost, then the feature is not used and the cost avoided.

There is an overhead cost in software for having some of these options available, even if they are not exercised by individual users. However, the system design has assumed that these are far offset by several types of savings that their presence makes possible. First, manual data processing by clerks is eliminated for this purpose. Second, results are sometimes achievable that could not possibly be obtained in other designs (for example, when a serious error is discovered some months after the data has been discarded and a large monetary decision hangs in the balance). Third, formal training expense is avoided by virtue of the new system design (e.g., users can learn by doing with minimal risk of data loss through user error).

Since data retention affects storage requirements (and costs), the system monitors storage usage and alerts the user when the amount of storage consumed exceeds a predetermined threshold. The system also provides the user with a storage command that reports the current storage consumption.

2.6 Data profiles

The functions and capabilities described so far in this section apply to most of the family of SPCS COER systems. However, one system, called Data Profiles, serves a special function. Data Profiles, unlike the other systems, is not restricted to a single-office view of data. It provides a facility for summarizing traffic data across many offices (and also across more than one service year).

This system collects preselected MLSS data from each SPCS COER office database and stores this information in a single hierarchical database. In a hierarchical database, data are divided, subdivided, and sub-subdivided in a treelike fashion as many times as needed to permit convenient access to its parts. Data Profiles divides and subdivides the MLSS data by BOC, type of *ESS* switch, particular office, and

collection period (current year, latest complete year, and archived older years).

The data for an office are collected by Data Profiles when the SPCS COER user requests a monthly report for that office. The database currently contains data for all 1/1A *ESS* offices. Data exist for each year, starting with 1978. Users can access any or all these data on an on-line demand basis.

Data Profiles uses a generalized database system/tool called the Off the Shelf System (OTSS),[8] which provides data management routines and a query language. This query language allows Data Profiles users to retrieve any items or combination of items stored in the database. The query language statements are simple. They consist of a few verb and modifier expressions that specify the part of the database to be traversed and what action should be taken.

By typing a single OTSS sentence, a user can ask, for example, "What was the average busy hour CCS per main station for *Touch-Tone** dialing digit receivers in New York Telephone offices during 1980?" Users can set screens on data so that the system will retrieve only data that satisfies certain conditions. For example, including the phrase, "when average busy season percent capacity is greater than 90," in the above query would cause the system to include in the average only data for receiver groups that were operating at more than 90 percent of capacity. The query language has commands that distribute, plot, print, alter, and make statistical computations on the data. Because queries may become complex and may be useful to many users on many occasions, a query library is maintained by the system.

Data Profiles provides a powerful tool for studies since it contains data for an entire area and so can be used by telephone engineers for many different planning purposes.

III. SOFTWARE STRUCTURE

The first SPCS COER system was hard coded with data structures and algorithms specifically designed for the 1 *ESS* central office application. The resulting system, although remarkably popular, was also inflexible and hard to enhance. What might seem a small conceptual change (the component busy hour feature) required a major rewrite of the system. Even the seemingly trivial change to compress the batch reports seven spaces to allow for hole punching required an embarrassing amount of reprogramming. In this section, we describe a software design method that led to SPCS COER modules that are easier to develop and maintain.

---

* Registered service mark of AT&T.

3.1 The view from the back room, or a hacker's view of history

One motivation for this design philosophy was the rapid proliferation of SPCS COER subsystems (serving different switching systems) during the mid-1970s. The 2 *ESS* COER was the first offspring of the initial 1 *ESS* system. This system was produced by the venerable red pencil technique. Engineers poured over the ponderous listings of 1 *ESS* COER (60K lines) deleting and changing sections of code that were not quite right for the new application. The resulting system worked but became difficult to enhance. Then 3 *ESS* COER was produced from 2 *ESS* COER in much the same way. The prospect of making parallel enhancements in these systems was awesome in much the same way that the Okefenokee Swamp is awesome. The laws of genetics suggested that this inbreeding would have to stop or the species would become extinct.

The crux came in the summer of 1977 when the SPCS COER team was faced with the task of developing four new subsystems (Busy Hour Determination, Voice Storage System, Remote Switching System, and Advanced Communication System). It was clear that the red pencil technique was no longer tenable. It was not so clear how these ambitious systems could be produced without a large development effort and substantial duplication of code. Fortunately, this crisis coincided with the emergence of the software-tools approach to system development. This method advocates developing larger systems out of a kit of small general-purpose tools rather than building monolithic special-purpose systems. This philosophy was an enormous help in managing the large development task. Rather than rushing off in four separate development efforts, time was taken to define a set of tools that could be useful in all four systems. The result was a marked reduction in development time and lines of code. In addition, the resulting systems are simpler and easier to maintain.

3.2 Tools plus

As valuable as the tools approach was, it was not the whole answer to the problem. Each SPCS COER system contains a fair amount of domain-specific knowledge both about traffic engineering and about a particular switching system. Without further philosophical underpinnings, it would be difficult to capture this knowledge in simple, generically useful tools. What was needed was some way to factor this specialist's knowledge out of the tools. The goal was to make the tools simple enough to be flexible and yet sophisticated enough to be knit into a useful system. The solution was to capture the knowledge of traffic engineering in a data model that can be used by the tools and to capture the switch specific knowledge in a set of tables that drive

the tools. As a result, the systems developer's job changed. Previously, it was to write or change large amounts of code to meet the requirements of the system. With the new method, it is to first select the proper tools for the system, then define the tables specifying the new system to the tools, and finally to write the controlling code which welds the individual tools into an effective system. The advantage is that the developer spends less time in low-level development and is free to devote more time to the design of an integrated system meeting the customer's need. In effect, with intelligent tools we have raised the developer's perspective from that of programmer to that of system designer.

The heart of the new approach is the traffic data model. The model consists of both a view of the traffic data and the routines that manipulate the data in accordance with this view. The traffic data that SPCS COER deals with come in two varieties—daily and MLSS.

3.3 Daily data manager

The SPCS COER Daily Data Manager (DDM) captures the notion of daily data in an abstract model. To DDM, an individual traffic measurement is represented by a tuple of five elements (day, hour, equipment tag, measurement tag, value). The quadruple (day, hour, equipment tag, measurement tag) is an index to the data values. Any measurement in an office's database can be accessed by specifying the identifying index.

The DDM provides standard database operations. The user of DDM can create and destroy databases, and add, delete, retrieve, and update items in an existing database. But DDM is a powerful tool because it understands the particular needs of an SPCS COER system. For one thing, it understands traffic data. It understands that the data are collected as a block of measurements for all the equipment in the office for a given interval. The data structures and algorithms of DDM are carefully tailored to efficiently organize and store data collected in this way. The DDM also understands that these measurements are subject to error and provides routines to flag questionable data items.

The DDM is particularly useful to SPCS COER systems because it understands how the systems are likely to use the data it manages. Report generation provides a convenient example. An SPCS COER report is a formattted output of the traffic statistics for a specific piece of equipment at a particular hour over a span of days. The DDM provides two special interface routines to aid in report generation. The first takes an hour, an equipment tag, and a range of days as an argument to prepare DDM to retrieve data for the report page in a systematic way. Each time the second routine is called, it returns a

new line of data for the report which includes the day on which the data were collected.

3.4 MLSS data manager

The MLSS database contains the same measurements as the daily database, but instead of keeping a day-by-day account, the MLSS database contains rank-ordered lists of the busiest days and the monthly averages on the equipment in the office. The MLSS Data Manager (MLSM) is an SPCS COER tool for managing MLSS databases. It views an MLSS data item as a five element tuple (hour, rank, equipment tag, measurement tag, value). The MLSM provides SPCS COER systems with the same sorts of services for managing MLSS databases as DDM provides for daily databases. In particular, MLSM maintains the rank-ordered lists.

DDM and MLSM free the individual system developer from worrying about the complexity of data management. They take care of storing and retrieving data on disk and worry about such issues as concurrency control and recovery. But more importantly, they enforce a standard data model across all the SPCS COER subsystems. This has two important ramifications. First, since all the traffic data are stored in a common manner, they can be analyzed by common software. Second, since the data are stored by a common data manager, higher-level tools can be built that use DDM and MLSM to fetch and store the data.

3.5 RSL

The SPCS COER Report Specification Language (RSL) is a good example of a high-level tool that uses DDM and MLSM. It also illustrates how a system designer can customize a generic tool to fit specific subsystem needs. The RSL is a simple but powerful language which allows the user to specify the format of a wide class of SPCS COER reports. Figure 2 shows a sample RSL program and Fig. 3 shows a report that was generated from this specification.

A brief study of Fig. 2 illustrates how RSL works. The first section of the report specification is the heading. This is the information to be printed at the top of the report. Information in quotes is printed here as it is written on the report. The key words offnam, datnam, schnam, and stunam specify translations to be made when a specific report is printed. The key word offnam will print the name of the office for which the report is being generated. Key word schnam will print the hour for which the data were taken, stunam will print the name of the equipment, and datnam will print the Gregorian representation of the first or last day in the report.

The supertitles section presents titles that span more than one

```
daily report 2 for (s52, s53, s54, s55, s56);
        flag line on position 9;
        heading
            offnam;
            '';
            'NO. 1 ESS SERVICE CIRCUITS AND SPECIAL TRUNKS';
            'DETAILED DATA FROM', datnam (fstday), 'TO', datnam (1stday);
            'ALL COMPONENT HOURS';
            '';
            '';
            'ITEM NO.', %d stutag, ':', stunam, '          ', schnam;
            '';
        end;
        supertitles
            'PER STATION' 14-24;
            'USAGE' 29-33;
        end;
        columns
            datnam (curdat) 1-8%s'' no average;
            i1  12-18%6.4f* 'CCS' exclude on '*%&';
            i2  20-25%5.3f* 'CALLS';
            i3  26-31%5d* 'TRFFC';
            i4  32-35%3d* 'MB';
            i5  36-41%5d* 'PEG COUNT';
            i6  43-48%5.2f* 'HT SEC.';
            i7  50-54%4.2f* 'OVFL %';
            i8  55-58%3d* 'CAP %';
            i9  59-63%4d* 'NCI';
            i10  64-69%5d* 'CAP CCS';
            100* (i3+i4) / 36*i9  70-72%3d  'OCC %';
        end;
end report;
```

Fig. 2—Sample RSL program.

0000013H MLDNMAELCGO SP CTX-8 3.5 COMP. ID: 001
NO. 1 ESS SERVICE CIRCUITS AND SPECIAL TRUNKS
DETAILED DATA FROM 07/01/79 TO 07/09/79
ALL COMPONENT HOURS

ITEM NO. 54: T-T MEMRY P 10:00

	PER STATION		USAGE		PEG	HT	OVFL	CAP		CAP	OCC
	CCS	CALLS	TRFFC	MB	COUNT	SEC.	%	%	NCI	CCS	%
07/01/79	0.0029	0.058	40		792	5.04	0.00	3	55	1256	2
07/02/79	0.0041	0.047	55		634	8.72	0.00	4	55	1256	3
07/03/79	0.0493+	0.047	66		643	10.26	0.00	5	55	1256	3
07/04/79	0.0077	0.052	105		705	14.89	0.00	8	55	1256	5
07/05/79	0.3333*	0.039	62		531	11.73	0.00	5	21*	1256	3
07/06/79	0.0070	0.066	2*		897	10.64	0.00	8	55	1256	5
07/07/79	0.0050	0.001*	68		547	12.36	0.00	5	55	1256	3
07/08/79	0.0054	0.051	73		693	10.59	0.00	6	55	1256	4
07/09/79	0.0060	0.043	81		581	13.96	0.00	6	55	1256	4
AVERAGE	0.0109	0.050	69		669	10.91	0.00	6	55	1256	4

Fig. 3—Sample report generated from an RSL program.

column in the report. The quoted string gives the title, and the numbers give the character positions over which the title should be centered.

The columns section specifies the main body of the report; each line

in this section specifies a column in the report. The first entry is the name of the data item to be printed in this column and the numbers show the character position boundaries for the column. Information after the percent sign tells how the value should be printed (s = string, d = integer, f = floating point, n.n = how many total digits in the number and how many digits after the decimal point. Compare with the C language function, printf). The asterisk shows that the data item can be flagged, and the string in quotes is the column heading. The key words no average indicate that there should be no average taken over this column. The key words exclude on followed by a list of flags indicate the data items flagged with any of the listed flags should be excluded from the column averages.

The system developer writes the report specification when designing the system. The RSL compiler converts the report specification into intermediate code to be interpreted at report generation time—that is, when the SPCS COER user requests a report of this type. At report generation time, the SPCS COER subsystem calls the RSL run-time interpreter with a short description of the desired report. This description includes the report type, the office, the equipment tag, the hour, and the range of days. The RSL interpreter then prints the report according to the specification and data. First, RSL prints the heading, supertitles, and column headings in an attractive fashion. It then calls DDM or MLSM to get the data for each line of the report, which it prints according to the column specifications. Finally, it prints the appropriate averages for each column.

RSL takes the drudgery out of report generation. Instead of writing code to fetch, format, and print reports, the system designer writes a few simple specifications that describe the required reports. RSL is able to do the rest because it understands what is generic in SPCS COER reports.

This section has described only a few of the tools used in the construction of new SPCS COER subsystems. The present arsenal contains many other tools both of the type providing some generic service, such as DDM, and of the table-driven type, such as RSL. Moreover, each new component SPCS COER system that is built suggests new tools so that the arsenal continues to grow. The net result is that the system designer can concentrate more and more on the problems of each system, with less and less bogging down in the details of coding.

The savings can be considerable. Studies on the RSS COER system indicated that tools have resulted in a 65-percent reduction in the code required. Tools have also made SPCS COER an easier project to manage. Since a large portion of the code is generic, it is well established and tested. This substantially reduces the maintenance burden.

In addition, better service is given to system users since new features and component systems can be developed more quickly.

IV. THE MANAGEMENT APPROACH TO SPCS COER SPECIFICATION AND DEVELOPMENT

Because of its origin as an experimental system, the SPCS COER project was managed with a research and development style,[9] instead of the classical software design style. Although many software projects seem to use this approach, it does not appear to be discussed anywhere in the software management literature. Weinberger hints at it in various places in his excellent book.[10] On the other hand, there is vigorous literature on research and development management methods.[9,11–14] We find this literature applicable to software projects and have coined the term "whole-job software design method" to describe it.

4.1 Whole-job team

The "whole-job" software design method puts all responsibility for a project, including requirements and development, on a small design team. The principal advantage of this approach is that the developers themselves have a close relation with the clients. This results in other derivative advantages: high motivation, flexibility, and low overhead. The team is responsible for:

1. Studying and understanding previous work.
2. Meeting with clients and understanding their views.
3. Proposing a solution. This includes selling the proposal to the BOC's, to AT&T, and to Bell Laboratories affected organizations and management.
4. Designing the system.
5. Implementing and testing the system.
6. Writing user documentation.
7. Conducting a field trial.
8. Managing the general release.
9. Following up on user problems.

Steps 1 to 3 are systems engineering functions, and 4 to 9 are development. In the whole-job approach, they are all done by the same team.

4.2 Structure and responsibility in the whole-job team

A team consists of one to a few members of technical staff. A team member may belong to more than one team. Typically, we have used people with Master's degrees in computer science. The teams are flexible; they reform from time to time as people come and go, but maintaining continuity of people assigned to a given project is a

primary goal of whole-job management. This has not been difficult to achieve.

For multiperson teams, the supervisor sometimes appoints a "team leader." Other times, team members are free to organize themselves as they see fit. They devise and tell the supervisor their plans and a method for reporting progress. The supervisor must choose between imposed and free organization. This choice is based on a variety of factors: the needs and expertise of the team members being the principal ones. In any case, the team plans its own work.

The team does not make commitments on its own to AT&T or BOC clients; management does that. With experience, this hard line can be relaxed somewhat. Team members learn management's views and make minor commitments. This smooths client relations because clients get quick answers.

The whole-job team tells management what it believes it can deliver and when. Management may adjust these estimates a little, sometimes a lot. But even if the dates are adjusted, the team members are usually confident they own them. This has led to much hard work and overtime, not because it was demanded by management, but because the team members assumed a responsibility to deliver to *their* clients what *they* had promised.

4.3 High team motivation

The whole-job approach places virtually all project responsibility on the shoulders of a few people. It is not divided among several organizations (e.g., a requirements group, development group, test group, maintenance group, etc.). The team members can immediately learn about and satisfy customer needs. This is a motivator.

High motivation arises from all the advantages of this approach: low overhead, closeness to clients, career flexibility, and broadening of team members. Small teams are easier to motivate than large teams. Self-organization is a motivator, a required motivator among professionals.

Team members develop a sense of pride in their project that we call "parental motivation." This carries into parts of the project that are less enjoyable, for example, documentation and maintenance. In SPCS COER, there have been few complaints about documentation and maintenance; they just get done. User documentation is delivered to end users at least a month before general availability of the software.

4.4 Low overhead

Low overhead arises for two reasons:

1. There are no "handoff ceremonies" from requirements to development, and from there to other follow-on organizations.

2. There is only one chain of command within Bell Laboratories that is responsible for the technical project management.

The whole-job approach allows the team to begin some design work even before requirements work is completed. There is a tight feedback loop from design to requirements because few people are involved. When a plan is generally accepted by clients, and details of formats, formulas, thresholds, etc., are all that remain, the planner's mind can gradually shift to design. Coding of stable or generic parts may even begin. Thus, a smooth and natural transition occurs.

4.5 Better understanding of the client's problem by developers

Requirements writers are usually close to the client. As they work their part of the problem, they can understand the client's point of view. They can see possibilities for trade-offs that will not compromise real objectives.

If the requirements people are the developers, then it follows that the developers understand the client's problems, have empathy, etc.

If there is a formalization of requirements and a handoff to developers, then the developers are once removed from the client. Many developers minimize this distance by attending some requirements meetings and getting to know the customers. Nonetheless, there is a distance between developers and clients when developers are once removed from the clients by project structure.

In general, the whole-job approach avoids the problem of distance between developers and clients. There are only two groups of people who must thrash things out, see each other's points of view, and agree: the whole job team and the client. One is trying to provide something that the other needs. Relations are simple. Add a third organization, requirements, and you must choose between the two communications networks shown in Fig. 4. The left plan has distance between client and developer, and slow feedback loops; is inflexible; and has difficulty in exploiting opportunities. The right has overhead, tri-team committee meetings, and is a little more flexible and capable of exploiting opportunities. The whole-job diagram is shown in Fig. 5. It has direct feedback and is maximally suited to exploiting opportunities.

Some developers see distance between development and client as helpful. It gives the developers more freedom to pursue design without reference to the customers' opinions. It protects developers from the well-known human trait of clients asking for more and more features on the original development schedule. Experience on the SPCS COER project indicates that neither of these conditions is a problem if the whole-job team works closely enough with the client.

The whole-job approach does have at least two problems. Close ties to the client tend to emphasize short-term over long-term goals and

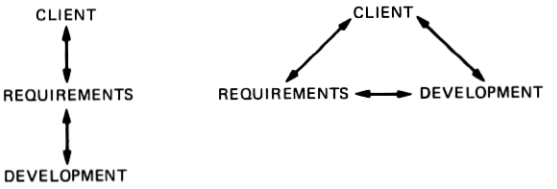


Fig. 4—Communications in a multiorganization software project.

Fig. 5—Communications in a whole-job software project.

activities. Overcoming this calls for very close ties and a willingness to invest time in explanations and discussions. When a client is obdurate, perhaps it really is better to emphasize the short-term goal. A second potential problem is that projects staffed by one or two people would seem extremely vulnerable to sickness and accidents. SPCS COER experience over 10 years shows that this is not a serious problem. Because of the multiply overlapping team design and because people develop friendships with their peers and like to talk, several members of other teams always seemed to know what a missing team member had been doing so that gaps could be quickly closed.

4.6 Career flexibility

System developers often view systems engineers as plodding writers of dull requirements. Systems engineers in turn frequently perceive development folk as narrow crank turners, plagued by self-imposed inflexible deadlines. While there may be something to be said for both points of view, both are distorted views and do not encourage communication and personal growth.

The whole-job approach ensures that individuals are exposed to the realities of both systems engineering and development—both the good and bad of each. This is a broadening experience. Entering the project with a development orientation, as most fresh college graduates do, many whole-job people develop a strong desire to move into full-time systems engineering. In our experience, this rarely happens in conventionally structured developments. Graduates of the whole-job experience have gone on to do well in both systems engineering and development areas. We view this as being greatly to the benefit of both the individual and the organization.

4.7 *Extending the scope of whole-job possibilities*

Whole-job projects are better for the customer than large multiorganization projects. Customers' needs are met more quickly and at lower cost. Long planning/development cycles are not needed. Customers' needs often change rapidly, and the whole-job approach is well suited to a rejuggling of priorities.

This conclusion seems evident. It is common knowledge that small, highly motivated teams can hit quicker and harder and more accurately than large organizations. In the military, commandos are considered elite. But commandos cannot win wars alone. And probably the whole-job approach cannot work on all software projects.

For small projects of two or three persons, most would agree it can be used. But size alone is not an adequate criterion. The method has worked on one medium-sized (10 to 20 person) project, SPCS COER, but may not work on others of this size. To find a criterion for where the whole-job approach can be applied, let us examine the structure of the project.

SPCS COER is organized horizontally instead of vertically, as shown in Fig. 6. SPCS COER is designed as a set of largely independent facilities. Of nine current and one future facility, their interdependencies graph is shown in Fig. 7. The asterisks represent customers. Most of the facilities depend only on BDE, which is a source of data from the world outside of SPCS COER. The 1 *ESS* subsystem supplies data to the world outside. Remove these two interfaces, and the graph reduces to Fig. 8.

What is remarkable here is the strong parallel between the customer's view of the project and its division into loosely coupled subsystems. Let's examine what is meant by the customer's view. Consider one of these subsystems, VSS COER. One person developed this subsystem. He consulted with the AT&T Network Design and Network Admin-

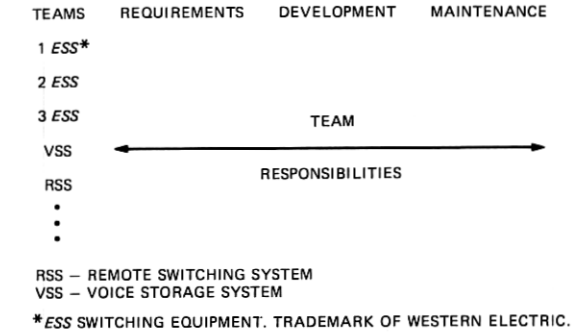
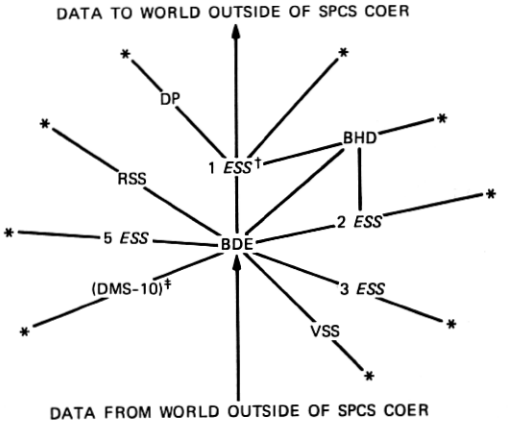


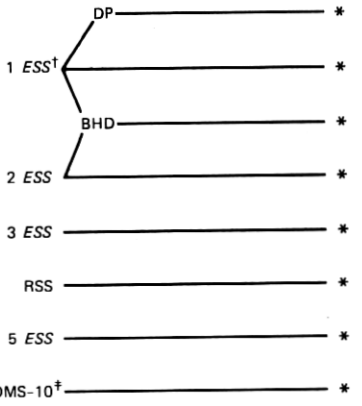
Fig. 6—SPCS COER team organization.



* – CUSTOMERS OF SPCS COER
BDE – BATCH DATA ENTRY
BHD – BUSY HOUR DETERMINATION
DMS – DIGITAL MESSAGE SWITCH
DP – DATA PROFILES
RSS – REMOTE SWITCHING SYSTEM
VSS – VOICE STORAGE SYSTEM

†*ESS* SWITCHING EQUIPMENT. TRADEMARK OF WESTERN ELECTRIC.
‡TRADEMARK OF NORTHERN TELECOM.

Fig. 7—SPCS COER subsystem interdependencies.



* – CUSTOMERS OF SPCS COER
BHD – BUSY HOUR DETERMINATION
DMS – DIGITAL MESSAGE SWITCH
DP – DATA PROFILES
RSS – REMOTE SWITCHING SYSTEM

†*ESS* SWITCHING EQUIPMENT. TRADEMARK OF WESTERN ELECTRIC.
‡TRADEMARK OF NORTHERN TELECOM.

Fig. 8—Simplified SPCS COER subsystem interdependencies.

istration departments in a small meeting. There was no representation from 1 *ESS*, 2 *ESS*, or any other parts of the COER project. The independence of the VSS module from other SPCS COER modules was clear from the customer's point of view. And this is reflected in the structure of this meeting. There are no customer requirements that the VSS database or processes interact closely with any other SPCS COER subsystem.

Consider 1 *ESS*, 2 *ESS*, and 3 *ESS* central offices, and the role of a real network administrator in a Bell Operating Company. One administrator may be responsible for all three types of offices. But this person usually works on offices one at a time. A network administrator may use 1 *ESS* COER for an hour or so examining a problem in one particular office, then switch to 2 *ESS* COER to examine the validity of some new data, and request a report. Although one person may use several modules of SPCS COER, the independence among these modules is still there from the customer's point of view. The independencies derive from the method of use.

So from both requirements and usage points of view, the customer's view of the project is a set of almost independent facilities. And each of these facilities is small enough to be planned and developed by a small team.

Underlying all SPCS COER, however, are common system design decisions: the on-line nature of the system, method of data entry, dialogue style, style of reports, etc. These can be viewed as system design decisions that affect all subsystems; but they were all made some years ago and evolve at a slow pace. It is these underlying design criteria that give SPCS COER a cohesiveness from the customer's point of view, i.e., make it one system.

So our hypothesis is this: It is not the size of a project, per se, that makes the whole-job approach applicable or not. The whole-job approach is applicable whenever:

1. A system can be divided into parts that are independent, from a customer's viewpoint.
2. The resulting parts are loosely coupled.
3. Each part can be planned and developed by a small, whole-job team.

Extending the whole-job concept to large projects requires that they be structured according to these criteria. Large projects that cannot be structured this way can probably only be done by the multiorganization approach.

4.8 Whole-job approach summarized

The whole-job approach to software specification and development is a project method. It is described in the research and development

literature, but apparently not in the software literature. It is a proven technique and has been in use on software projects for at least 10 years. Its central advantage is that developers and customers build a close relation. This leads to realistic design by the designers, and high customer acceptance of the software products. Structure of a project, not size, determines whether this method can be used.

V. FUTURE DIRECTIONS

SPCS COER, well into its second decade of service, is still growing vigorously in features, component systems, and entities served in response to the expressed needs of its users. But this is only to be expected of a successful product serving the needs of a growing market. We close this paper briefly citing four new and challenging areas toward which SPCS COER thinking is being directed. It is not possible to tell now which of these areas, if any, will bear fruit.

As mentioned earlier, an SPCS COER system was developed for VSS. While VSS COER is not now in operation for reasons having nothing to do with the present paper, this development was an interesting first application of SPCS COER technology outside the realm of switching systems. There appear to be many other potential applications beyond switching that require the capabilities of SPCS COER. Work is presently proceeding to expand our applications in this arena.

The reports produced by SPCS COER are laboriously worked out with user representatives to meet the reporting needs of the BOCs. But surveys have shown that fixed-format reports seldom meet all the needs of individual users. This leads to wasteful manual posting from different mechanized reports to the desired report format. Planning is well advanced, under the title of "Flexible Reporting," for new features that will allow users to alter the format and content of existing reports and create new reports from scratch.

While the development of generic tools, described in Section III, has greatly simplified and speeded the development of new SPCS COER systems, developing a system for each new application is still a significant and time-consuming effort for highly skilled and experienced program designers. Under the name of "Generic COER," also referred to as "User Programmability," planning is under way to allow users to (in effect) create their own component systems for new switching applications. Unlike flexible reporting, the user in "User Programmability" need not be the ordinary SPCS COER user. Most hands-on users of SPCS COER are clerks with little formal training. The user in "User Programmability" could well be a highly trained computer program designer employed by an individual BOC as long as such a resource were readily available to the operating users in the BOC.

Finally, in the 12 years SPCS COER has been operational, there has been a continuing advance in the computing facilities available to the BOCs in their data centers and elsewhere. It now seems possible and economically attractive to move the SPCS COER operation from its single central location for all BOCs into one or more installations in each BOC. Planning work is in progress on this future direction.

REFERENCES

1. H. J. Dougherty, H. Ragg, P. G. Ridinger, and A. A. Stockert, "No. 1 ESS Master Control Center," B.S.T.J., *43*, No. 5, Part 2 (September 1964), p. 2286.
2. E. P. Gould and J. W. Mosior, "TELPORT - Time Shared Information Systems," Bell Lab. Rec., *46*, No. 6 (June 1968), pp. 197–202.
3. N. R. Sinowitz, "DATAPLUS - A Computer Language for English Speaking People," Bell Lab. Rec., *46*, No. 11 (December 1968), pp. 362–9.
4. M. S. Hall and G. P. Patti, "PATROL Surveys Traffic for No. 1 ESS Offices," Bell Lab. Rec., *51*, No. 3 (March, 1973), p. 72.
5. V. L. Fahrmann, "Reporting Central Office Traffic Information," Bell Lab. Rec., *57*, No. 4 (April 1979), pp. 99–104.
6. M. S. Hall, J. A. Kohut, G. W. Riesz, and J. W. Steifle, "System Plan," B.S.T.J., this issue.
7. K. A. Friedman, "Precutover Extreme Value Engineering of a Local Digital Switch," Proc. of the Tenth Int. Teletraffic Cong., Montreal, June 8–15, 1983.
8. L. E. Heindel and J. T. Roberto, "The Off-The-Shelf System—A Packaged Information Management System," B.S.T.J., *52*, No. 10 (December 1973), pp. 1743–63.
9. E. B. Roberts, "Stimulating Technological Innovation—Organizational Approaches," Res. Manage., *12*, No. 6 (November 1979), pp. 26–30.
10. G. M. Weinberger, *The Psychology of Computer Programming*, New York: Van Nostrand Reinhold, 1971.
11. T. J. Allen, D. M. S. Lee, and M. L. Tushman, "R&D Performance as a Function of Internal Communication, Project Management and the Nature of the Work," IEEE Trans. Eng. Manage., *27*, No. 1 (1980), pp. 2–11.
12. J. D. Goldhar, L. K. Bragaw, and J. J. Schwartz, "Information flows, management styles and technological innovation," IEEE Trans. Eng. Manage., *23*, No. 1 (1976), pp. 51–62.
13. T. J. Allen and S. I. Cohen, "Information flow in research and development laboratories," Admin. Sci. Quart., *14* (1969), pp. 12–19.
14. T. J. Allen and A. R. Fusfeld, "Design for Communication in the Research and Development Lab," Technol. Rev. *78*, No. 6 (May 1976), pp. 65–71.

AUTHORS

Richard F. Grantges, B.S., B.E.E., 1953, University of Minnesota; Bell Laboratories, 1953—. Mr. Grantges was first assigned to long range engineering studies of transmission systems where, in 1954, he demonstrated the economic feasibility of Pulse Code Modulation for exchange carrier. After military service he was transferred to *ESS*® Systems Engineering in 1958 and promoted to Supervisor in 1961, responsible for switching network engineering and the mechanization of central office engineering practices. He was promoted to Head in 1968, responsible for appraisal studies in the Traffic Studies Center. In 1969, he went to AT&T as Traffic Operations Manager—Electronic Switching, responsible for the traffic engineering and administration of local and toll electronic switching machines. Returning to BTL in 1971 as Head of the Operator Services and Traffic Engineering Department, he was active in planning the mechanization of operator services and traffic data collection and processing, particularly the SPCS COER, Small Office Network Data System and EADAS systems. In June 1981, he became responsible for the

systems engineering of the Total Network Data System (TNDS) except for Network Management and Trunking. Member, Tau Beta Pi, Eta Kappa Nu, IEEE.

Virginia L. Fahrmann, B.A. (Mathematics), 1963, St. Mary-of-the-Woods College; M.S. (Mathematics), 1965, Purdue University; Bell Laboratories, 1965—. Ms. Fahrmann initially worked on traffic load simulations of 1 *ESS* and No. 1 Traffic Service Position System (TSPS). In 1971 she joined the AT&T Switching Engineering Department, with responsibility for providing technical support in long-range planning studies for local wire centers to Operating Companies. She returned to Bell Laboratories in 1973, and was involved in systems engineering and development for SPCS COER. In 1980 she became Supervisor of one of the two SPCS COER groups. Currently, Ms. Fahrmann supervises a group that works on requirements and development for the Plug-in Inventory Control System/Detailed Continuing Property Record (PICS/DCPR) system, which provides inventory control and maintains investment records for central office equipment.

Thomas A. Gibson, B.S.E.E., 1960, Kansas State University; Bell Laboratories, 1960–. At Bell Laboratories, Mr. Gibson did systems engineering studies on Electronic Switching Systems and developed a variety of software systems including the Master Links database management system and SPCS COER. He is currently Head of the Plug-in Inventory Control System/Detailed Continuing Property Record Development and Support Operations Department. Mr. Gibson has chaired a subcommittee of the American National Standards Institute. Member, ACM, Sigma Xi, Eta Kappa Nu.

Laurence M. Brown, B.A. (Classics), B.S. (Fundamental Sciences-Computer Science), 1975, Lehigh University; B.A. (Literae Humaniores), 1977, Balliol College, Oxford University, Oxford, U.K.; M.A. (Electrical Engineering), 1978, Stanford University; Bell Laboratories, 1977—. Mr. Brown first worked on the software tools of ESS COER. He is presently Supervisor of the Software Systems Design and Support Group.