This paper examines the function and design of an accounting program package for a medium to large computer installation. A specific implementation is used to illustrate key points.

Computer installation accounting

by H. M. Gladney, D. L. Johnson, and R. L. Stone

Much has been written on accounting for computer installations, and much remains to be written. Unsettled questions include:

- What must be done to ensure that computational resources contribute to the strategic objectives of the organization?
- What are the economic effects of different pricing policies on the effectiveness, efficiency, and growth of an installation?
- What must be done to comply with government guidelines for procuring computation as part of R & D contracts?
- How can those guidelines be reconciled with internal objectives?
- Is pricing effective in the allocation of computer resources?
- To what extent are complicated accounting methods acceptable to users, and how much do they encourage desirable behavior?
- To what extent is price reproducibility necessary and feasible?

In addition, complex technical problems are involved in measuring the resources delivered in multiprogramming and multiprocessing installations.[1] Few installations can afford to wait for such issues to be resolved, particularly since, for many of them, resolution in the sense of a definitive policy is not to be expected.

Fortunately, it is possible to define an accounting mechanism that does not implicitly assume specific policies, but is helpful in focusing attention on policy questions that must be addressed. This can be done in a way which does not require that individuals or groups responsible for setting policy understand the

more arcane details of measurement and charging, and which gives the individual user enough information to make reasonable decisions in structuring his application to be economical.

Commonly, each computer installation has its own locally developed accounting program, primarily because computer manufacturers do not supply such software. Also, local development sometimes is rationalized by arguments that each installation has its own peculiar financial policies, equipment, and user requirements. We argue that there is a generalized approach by which an accounting system can be designed to satisfy the requirements of most, if not all, medium- to large-scale installations. It must be designed "top-down" to support cost-center, profit-center, or line-funded installations and be independent of the choice between fixed and dynamically adjusted prices. The installation's policy options can be implemented either with parameter changes or with self-contained procedures that make use of specific rate formulas. For illustration, we will use a program developed jointly by two installations between 1969 and 1972, and used for about three years, in various versions, in a very large installation (the System/360 Model 195 at the IBM Research Division laboratory in San Jose, California).

Occasionally, the need for detailed accounting analysis has been questioned as an unnecessary administrative task. It is our thesis that resource allocations will be made, if not by conscious and informed decision, then by default. Even if the final decision is to implement a simplified scheme for measurement, accounting, and allocation, analysis of the type we illustrate should be made in order to get an insight into how the accounting and allocation policies will affect the installation.

Once a commitment is made to collect utilization data in a computing facility, the marginal cost of running a more elaborate system to analyze the data can be small, provided that the accounting data base is suitably structured. We believe that the cost is well worth the investment, especially if a generalized package can be acquired and customized for the particular installation.

In subsequent sections of the paper we refer to previous contributions that address parts of the subject in more detail than we can include; we summarize objectives and problems encountered in designing an accounting program; we give an explicit example of an algorithm for distributing costs as prices for service; we describe a method of estimating the costs of resource pools; and we describe a program that we implemented, with emphasis on minimizing the clerical effort required and ensuring optimal auditability, and with comments on how it might be done better in the future.

Background

There are several complementary views of the role of pricing in a computer installation. Nielson[2] discusses it as a mechanism for decentralizing resource allocation decisions so that such decisions are made in the most appropriate parts of the organization. Extreme decentralization is common in universities, where users' objectives are heterogeneous and the major purpose of the administration is to provide an environment for independent thought and study. Accordingly, Nielson focuses on access by the individual and on establishing priorities for charging. He discussed flexible pricing in an earlier article.[3]

McFarlane et al.[4] discuss pricing as a tool with which management can translate strategic objectives into action, and they are led to examine mechanisms for control. Questions facing top corporate management are: what resource commitment should be made to computing; how should the resource be deployed for maximum effectiveness; and are the resources being used efficiently. The authors indicate that the accounting control system should have formal mechanisms for providing relevant information to responsible managers and that the mechanisms must include schemes for monitoring the use of computer resources and for communicating this information to decision makers and motivating them to take action. According to McFarlane et al., the accounting control system should help bring to management's attention such aspects of computer service as the high ratio of fixed to variable costs, the large economies of scale in hardware and software, the large increments in which capacity is acquired, the fact that demand growth can be very large, and the fact that peak and slack load cycles and flexible priorities are intrinsic.

Sharpe[5] takes an economic viewpoint in which he assumes a perfect market. He focuses on overall pricing strategies for the services provided by an installation, as compared with the distribution of prices to specific service categories. In this model, each part of a parent organization strives to maximize the total value (or benefit) it receives. Sharpe discusses the possibility that optimal strategies for parts of an organization may not be optimal overall. If installation managers take this economic viewpoint, they can focus on measurements that indicate what can be done to increase the average and marginal cost-performance ratio of the system. Such formal theories provide useful background, but their practical value is limited in the daily environment since the value of computation service is difficult to determine precisely.

Singer et al.[6] consider circumstances in which prices are not the dominant mechanism for allocating computer time. They discuss whether prices should be used at all, or whether alternative,

nonpricing methods can be expected to work. They identify the prerequisites of allocation based on prices: the users must have budget constraints that are not forgiven by the administration when overruns occur; there must be a fluid market for services, with alternatives for both customers and vendors; demand must exceed capacity; and prices must be free to fluctuate without reference to costs. In this viewpoint, pricing is not to be regarded as a cost recovery method. Several interesting concepts emerge: that idle time has value in that users can buy it to ensure better response; that profits and losses are simply transfer payments between the corporate body and the installation and can be regarded as a form of line funding; and that priority mechanisms can be regarded as nothing more than another form of pricing.

Overall objectives

Whichever viewpoint is adopted, it is necessary to have information relating the service delivered to its price and/or cost. It is pertinent to note that accounting systems explain costs, not value-to-user.[7] Perhaps a qualitative feature that distinguishes a good accounting mechanism is that it allows management to focus on value, in confidence that cost is under control.

Before getting into the detail involved in designing an accounting program, it is worth summarizing the objectives in broad terms. The analysis and reporting of computer installation utilization data should be an integral part of the cost accounting system of the organization[8] and should meet the same audit requirements as other parts of the accounting function. Clearly reported costs of the various services enable functional managers to make cost/benefit decisions selectively and to economically "tune" their part of the business. In some circumstances, there is a contractual or legal requirement to demonstrate that cost assignments are made within certain guidelines.

Because the computer billing system can conveniently provide system performance information, it can become a component in planning, both within the computer facility and for groups of users.[9] The billing system should demonstrate utilization and cost trends clearly enough to permit projections to be made by customers. It can be used by installation management to plan and justify hardware acquisitions (such as additional direct-access units) or discontinuances, and major changes in software.

Data made available by the accounting mechanism inevitably become an element in management's evaluation of the computer installation. The utilization data enter into dialogs both between upper management and the installation manager, and between

the installation manager and his subordinates, who often are charged with solving problems exposed by the data or with maintaining utilization statistics within specified bounds. The machine operations manager, for instance, commonly is charged with keeping unscheduled maintenance time (down time) under control. McFarlane[4] elaborates by referring to the systems programming function, which often is difficult to manage by formal control mechanisms because it is a highly skilled activity in which obscure errors of approach or execution can have adverse effects far from the areas for which an individual programmer is held responsible. McFarlane suggests that management should give close attention to understanding the results intended and the extent to which they are attained, and that systems programmers should be held accountable for the overall efficiency, reliability, and responsiveness of the system. Measures of efficiency, reliability, and responsiveness can be made available in the accounting system.

From the users' point of view, the accounting system is an important interface to the computer facility. Users should be able to employ it to plan their future costs, to design programs for economy, to ascertain whether they are getting their fair share (or committed share) of the resources, and to reassure themselves that the computer facility is being run efficiently. The accounting system that does not convey useful information in easily decipherable form to the cost-conscious user will quickly become an organizational issue.

Many installations, at one time or another, will contemplate partial cost recovery by selling service outside the parent organization. Establishing a basis for pricing outside service is hardly possible without an internal basis.

Selection of a pricing scheme determines the type of uncertainty to be borne by the user,[5] who must choose between alternatives (e.g., fast turnaround vs. premium prices, time sharing vs. batch) which usually involve expenditure of the user's time in varying degrees. Effective allocation calls for a structured price schedule: there must be a price for each system component that is to be regulated or cost-recovered. In general, each price will be a function of the time, the load, and the cost of the service. The basic issue is between effective allocation and the user's acceptance of accounting complexity.

Planning the accounting program

When the second version of the accounting procedures for the System/360 Model 195 installation at San Jose was started late in 1971, we wrote a programming specification in broad terms.

The specification was designed to set forth general objectives, expose difficult or incomplete aspects of the method adopted, and serve as a reference against which program implementations could be measured. It was intended that the document be available to users or managers to whom detailed information on the accounting method was important or interesting. The programming specification was to be supplemented by separate memoranda documenting the programs and specifying the administrative procedures. It was successful in the sense that little revision seems necessary today. Much of the rest of this paper is extracted from that specification.[10]

Our specific objectives were that:

- Each customer should be charged for the resources his work consumes. The charges should be based on job characteristics that the user can understand and control. For each job or terminal session, his usage measures should be returned to him. In addition, there should be comprehensive summaries of appropriate scope for several levels of management and administration.

- Under certain circumstances, there should be charges for denying resources to other users. The limiting case is a uniprogramming system, for which it is reasonable to charge on the basis of elapsed time. The best known practical example is stand-alone time, for which the user must pay the entire cost of the machine regardless of the use he makes of it.

- Accounting should be supported on a cost-center basis. That is, in specified accounting periods, the rates should be adjusted automatically so that all costs are apportioned among users in proportion to their consumption. The net profit or loss of the cost center is exactly zero. As an alternative, the program should also support fixed rates.

- The charging system should be stable enough to permit prediction of charges, within reasonable limits (±5 percent), so that users can plan their budgets.

- No new data gathering package should be written. For OS/360 and OS/VS installations, the OS System Management Facility (SMF)[11] is the only IBM support available. If other data gathering tools are used, such as installation generated SMF records or on-line storage records, the accounting program must be downward compatible to installations that have not installed such modifications or additions.

- The program should be able to deal with peak and slack periods by applying shift differentials or permitting charges for priority service.[12]

- Each resource pool for which charges are made should be constructed to facilitate the use of pricing as an incentive for balancing resource usage or for signalling the desirability of changes in configuration. For example, if tape usage is heavy, the price per use will drop, indicating that it might be desirable to install more tape units.

- The accounting program must handle arbitrary time periods; that is, the input data sets must be considered files without starts or ends. For example, it should be possible to summarize system usage between 12:35 p.m. on 04/07/74 and 3:04 a.m. on 06/20/74.

- The accounting function should be automated, from measurement of usage to final posting of ledgers. It should be able to handle 1,000 to 10,000 job steps a day, and it should include adequate error handling methods (e.g., rerun credit) and administrative adjustments. Each step requiring human intervention should be reexamined for necessity, and the exposure to human error should be identified.

- In the event that an installation runs several computers, the program should support collation of charges from all machines. It should produce collated summary and project reports.

- The accounting mechanisms must meet stringent auditing requirements, including written program specification and documentation; a file of source code listings and control of new versions of the program; separation of operational, programming, and accounting functions; regular distribution of usage reports to affected organizational functions; and an archival accounting data file of manageable size.

- The implementation should be open-ended in the sense that new services or improved data gathering tools can be incorporated easily.

- There should be fail-safe mechanisms to prevent losses of accounting files, and procedures should be specified to account for tasks that are active at the time of a system crash or abnormally terminated by errors for which the installation is responsible. Since such mechanisms are not provided in the SMF package, the installation must provide them. It is not difficult to implement SMF exit routines[13] that write duplicate copies of SMF data to a file which can be used if the primary SMF files should be damaged or otherwise lost.

- The total system summary must account for exactly 24:00:00 hours a day.

The difficulties and inconsistencies we were aware of included some that were intrinsic and some that were peculiar to the OS/360 environment:

- If the installation is a cost center, charges for the services necessarily fluctuate, depending on the overall load and how effectively the facility's management predicts the load and adjust costs and capacity. The usual technique for minimizing fluctuations is to average usage levels and costs over a sufficient period to minimize extremes and permit publication of rates to customers.[14] A moving average of several months is usually sufficient. This process, however, discourages the partitioning of resources into "subpools" apportioned according to usage levels, since the smaller the resource pool, the greater its chance of exhibiting widely fluctuating usage levels. It is an unfortunate attribute of the cost center that, regardless of the number of pools into which the resources are divided, the economic motivation of its pricing is always counterproductive: as the resource becomes lightly used, it falls into disuse, and the price per unit increases, further discouraging its use. (Fortunately, price is not the only motivation at work in apportioning resources; there are also such factors as turnaround responsiveness, which influences loading.) In practice, for resources whose usage levels may vary widely or be unreasonably priced (such as newly introduced services or resources), we compromise by lumping their costs into the overall cost center, and we make resource subpools of larger items such as CPU time and I/O facilities.

- The complexity of a time-sharing, multiprogramming system seems to preclude a simple relationship among costs incurred by the installation, usage parameters that the customer can understand and control, and load measurements that are available from standard measurement programs. For example, execution of a channel program requires storage space and CPU time as well as I/O paths and devices. This difficulty may be partly compensated for by adjusting the rates of related services, but some arbitrariness will remain in overhead estimation even in a careful approach.

- Frequently the usage measurement is imperfectly related to the actual activity. For instance, a channel program count is not proportional to either the amount of data transferred or the amount of time the channels and control units are busy. It should be recognized that the characterization, measurement, and prediction of the load on a system from a task or set of tasks is a major area for systems research. It seems inappropriate for the personnel of a service installation to do more than stay aware of the current state of the art and ex-

ploit the most significant advances on a conservative schedule. Many of these advances will involve basic changes to system architecture and to the accounting and data-gathering component of the system. One should consider modifying the latter to be a project distinct from adjusting the accounting procedure.

- In a complex environment, a task may generate considerably different usage measurements in repeated executions; for example, in a paging machine, the number of page replacements will depend on paging activity interference from other tasks. Not only is this type of variation philosophically objectionable; very often it implicitly creates a price differential between peak load and off-shift periods. Sharpe[5] illustrates this with the case of a time-sharing system in which CPU time and terminal contact time each have fixed-rate prices: since response is slower during peak-load periods, prime shift services not only will be less desirable, they will be considerably more expensive than having the same work done at otherwise unattractive hours.

- We have inadequate insight to relate prices for the same job run on different machines. It is often argued that prices should follow machine costs, permitting classical market forces to act, but this can lead to difficulties in a multiple-CPU installation where the user has some freedom to select among the various machines. Work tends to migrate to the machines with the best performance, leading to overloads on those machines and under-utilization of others. If each machine is a separate cost center, there is economic motivation for some machines to be used more than others, making the loading even less stable.

- Collecting accounting data and generating reports can become too expensive if too much detail is involved. For example, SMF can provide device usage for each unit employed by a job step, but to do so may require several million records a month for an installation processing 50,000 jobs a month; we are aware of one case in which data gathering alone consumed about two percent of the installation's capacity. Sometimes we can see only in retrospect that the data collected is too much or too little to provide the utilization and cost information needed. Fortunately, SMF allows widely varying levels of data collection.

- The cost of some resources, such as CPU time, can reasonably be divided among users proportionally to their usage, since the cost is incurred whether the resource is used or not. For other resources, such as tape mounting, it can be argued

that a fixed rate per event is appropriate. A hybrid of fixed and varying rates is feasible, but not elegant.

- It is common practice to base charges on measurements of machine usage, and to include, as overhead, staff services (e.g., consulting, tape librarianship) which are seldom proportional to machine usage. This inequity can be dealt with in part by charging separately for the more expensive overhead services, with the disadvantage that this method requires manual data-entry steps. Another mechanism is to put a premium on machine services, such as volume mounting, which require human intervention.

- Sometimes the data collection method omits information necessary for the implementation of consistent policy. For example, in OS/SMF, the usage measure for I/O activity is executed channel programs (EXCPs). Unfortunately, an EXCP may move widely varying amounts of data; a better measure would be channel bandwidth used, integrated over time. Similarly, the CPU time measured by SMF may vary widely due to paging activity and interruptions caused by the servicing of other tasks. (Curiously, the *interrupted* program is charged with some CPU time in OS.) A better measure would be instructions executed.

An algorithm for distributing costs

In this section and the next, we describe a method for setting prices if an installation is run as a cost center. The following definitions apply: A *resource* is an identifiable service or machine for which an installation incurs costs. Some resources are not introduced explicitly and thereby become overhead. A *usage measure* (u) is a count (in specified units) made available by SMF for each service delivered.

For brevity, we will refer to any unit of service as a *job step* (j). This will include time-sharing sessions, unit-record work, stand-alone time, or any other unit of service the installation wishes to define. A *charge* (c) is an apportionment, for a single job step, of the price of a resource. Each charge is calculated as a function of one or more usage measurements. Charges may be made at either variable or fixed rates. Variable charges are calculated as the user's equitable share of a pooled resource. Fixed charges are calculated at a rate independent of the load. Associated with each job step is a *class identification* (C), which can be used to implement differential charging policies. A class can identify either a service priority or a type of service, such as an APL terminal session.

The objective is to apportion resource costs among job steps according to their usage measures in an equitable, readily understandable way, and to accumulate charges for each project. This can be done in four stages: determination of the fraction of each resource used by each job step, determination of the fraction of the installation's cost associated with each resource, accumulation of each customer's fractional use of the cost center, and, finally, conversion of these fractions into dollar charges based on the expenses of the cost center. For resources that are to be charged for at a fixed rate, a modified method can be used: fixed charges can be accumulated separately and deducted from the cost center expense before the rate for variable charges is calculated.

resources used

The resources used in our installation were: problem program CPU time; problem program channel program count (EXCPS); problem program occupancy of main storage; direct access storage space; tape mounts; unit record output operation;[9] unit record input operation;[9] teleprocessing port connect time; job step initiation; graphic console time; stand-alone system time; off-line pack rental and other miscellaneous services; and spindle occupations for direct access devices. Of these, we chose to handle direct access storage, tape mounts, the graphic console, teleprocessing port usage, and unit record operations as fixed-cost resources. For direct access storage, we decided that the most precious resource was long-term space usage. Consequently, rather than trying to monitor this resource on a job step basis, we decided to sample the on-line data base periodically to obtain a profile of the space used, and to charge users according to this sampling. Several other resources were considered but not used in our implementation; they included plotting, data set allocation, and private direct access device usage.

It must be emphasized that, given the input data we discuss, it is a minor program change to adopt formulas other than those we outline below. For expository reasons, we have simplified some of the formulas that were actually used.

The charge for resources that are billed at variable rates is measured in arbitrary units called machine units. For the jth job step, the charge, c_j, is the sum of subcharges for each of the resources used by that job step, as expressed by

$$c_j = \sum_i s_{ij}$$

In this formula, s_{ij} is the subcharge, in machine units, for job step j's use of the ith resource.

Each subcharge s_{ij} is calculated as follows:

$$s_{ij} = TK_i \frac{f_i\,(u_{ij}, p, C, t_j)}{\sum_k f_i\,(u_{ik}, p, C, t_k)}$$

Here, T is the total number of machine units (seconds of availability of the entire system) in the accounting period, K_i is the fraction of the system recovered by the usage of the ith resource, and f_i is a function of the usage measure, u, the class, C, the time of job step initiation, t, and externally supplied parameters, p. The denominator represents the sum of the usage of all job steps in the accounting period. For resources charged at fixed rates, the normalizing denominator is omitted in the formula above, and the coefficient TK_i is replaced by a dollar rate per unit of service. For simplicity in exposition, these formulas and those below are written for an installation with a single machine. The result of extending the concept to multiple machines is obvious: subscripts proliferate.

usage measure functions

Our usage measure functions follow. In each case, d_t is a shift differential function whose value depends on the time of day when the job step is started; it is piecewise constant, but not continuous. A mechanism is provided for charging weekend and holiday usage at third-shift rates. D_c is a premium factor for certain job classes. It allows resources to be priced differently for different usage classes. (For example, it may be desirable to induce users to make use of resources in a cost-effective way, as by establishing a low price for low-priority background work and a high price for resources used in the time sharing environment.)

- CPU: $f_1 = d_t \cdot D_c \cdot$ (problem program CPU time).
- Channel programs: $f_3 = d_t \cdot D_c \cdot$ (EXCP count).
- Direct access storage: $f_4 = \int$ (space occupied) $\cdot\, dt$. The direct access occupancy integral is measured by rectangular integration based on periodic sampling at unequal intervals (about once a week).
- Tape mounts: $f_5 =$ (number of tape mounts).
- Unit record input/output: $f_6 =$ number of cards punched, local or remote + number of lines printed, local or remote. We used HASP to provide the unit record usage statistics by writing installation defined records onto the SMF file from user exits provided by HASP. The unit record input load was omitted, since it is negligible compared with output. (It should be noted that in OS/VS, JES records are in SMF.)

- Job step initiation and allocation: CPU, main storage, and I/O resources are all required whenever a job step is initiated. Most of the system operating cost is in accessing system data sets, allocating data sets, and the main storage occupied by the operating system during job step initiation. For the sake of simplicity, we chose to charge initiation as an I/O surcharge and a main-storage-occupation surcharge. Since the cost of any initiation increases with the number of data definition statements, n, an increment of the form $(a + bn)$ would be more precise.

- TP port usage: f_7 = (length of time connected). SMF records the elapsed time of job steps from initiation to termination; for TP ports, this time approximates the actual connect time (for normal user behavior).

- Graphic console time: f_8 = (length of time graphic device is allocated) $\cdot\ d_t = t_{elp} \cdot d_t$, where t_{elp} represents the actual elapsed time of a job step.

- Main storage occupation: Since the actual elapsed time varies for a job step executed at different times in a multiprogramming system, it is desirable, in the interest of consistency, to estimate the pseudo-elapsed time—that is, the length of time a job step would occupy main storage if it were the only program in the system. In a spooled, disk oriented system, a reasonable approximation is the sum of the CPU time and the product of the number of direct access I/O requests and the average delay on each request. The average delay is different for job steps in which the application is multitasked to overlap its own CPU and I/O time. For batch job steps involving interactive displays, the time in main storage is determined not by job characteristics, but by user response times, so that such job steps are best charged for according to total elapsed time rather than pseudo-elapsed time. For the region size, it may be desirable to have varying rates so that, as an installation option, extremely large jobs can be penalized for reducing access to the CPU by other jobs; we avoid this extra complexity in the discussion.

  The pseudo-elapsed time, t_{elp}', is

  $$t_{elp}' = (\text{CPU time}) + (\text{average time for an EXCP}) \cdot (\text{EXCP count})$$

  For any job, then, the elapsed time, g, is given by

  $$g = t_{elp}' + b\delta(t_{elp} - t_{elp}')$$

where $\delta = 1$ for a batch graphics job step and $\delta = 0$ otherwise, and where b is an installation parameter that could be used to reflect graphic device rental and overhead as well as main storage occupation costs.

Then the main storage usage measurement function can be taken to be

$f_2 = g \cdot d_t \cdot D_c \cdot$ (storage allocated in kilobytes)

For interactive systems, the main storage formula can be modified to account for resident portions of the operating system dedicated primarily to time sharing support:

$f_2 = d_t \cdot t_c \cdot$ (TS region size + TS monitor region size)

The foregoing largely presupposes that the computing environment includes a fixed-capacity main storage, whose space is precious. In a virtual storage environment, the occupancy (in the sense of total virtual storage used) is less important. A measure more closely related to the utilization of a precious resource would be real storage occupancy and paging activity.

- Finally, for each project, the total charge, c, in machine units is the accumulation over all job steps for that project:

$$c = \sum_j c_j$$

For each project, there is also an accumulation of fixed prices, denoted by P. In addition to items already referred to, fixed prices include rental for terminals allocated to specific projects but managed by the computer facility, stand-alone machine time, set-up time, and special services. For each fixed-price item, the rate generally includes administrative overhead as well as hardware rental costs.

edit facility

The accounting program includes an input facility for charging for services not measured automatically and for crediting facility errors. It provides for credit and debit entries, charges in terms of machine units or dollars, and charges on either a prorated or an absolute basis. Input data can be repeated in appropriate output reports as an auditability feature.

rerun credit

The program credits rerun costs automatically for jobs in execution at the time of a system failure and for jobs where the SMF entry is incomplete because of an SMF failure. In addition, when there is a known system problem that causes abnormal termination of a user program, the program can charge the affected runs to the rerun account automatically, and it can accu-

mulate relevant statistics in a special output report. This facility reduces administrative overhead and obviates a remarkable number of minor complaints on the part of computer facility users.

extension to virtual systems

In OS/VS2 Release 2, SMF provides additional information regarding resource utilization: specifically, the above algorithm might reasonably replace the usage measure functions for CPU time, channel programs, and main storage occupancy with functions for service units[13,15] and paging activity, and the class (C) with the performance group number. Of these functions, paging presents something of a problem. Generally, good accounting practice strives to minimize fluctuations in costs over which the user has no control. Paging activity represents such a cost since it depends on the configuration of active tasks in the system. On the other hand, the paging activity of a task may be vastly influenced by programming techniques, and it seems desirable to implement charging schemes designed to encourage good programming. Perhaps there is a trade-off to be made, or perhaps there is a technical solution that permits paging activity to be measured independently of neighboring tasks. This is a subject that calls for further research.

Estimation of resource costs

In this section is described a method for estimating K_i, the fraction of the system to be recovered in proportion to the ith usage measure. Note that it is not possible to ascribe system rental costs unambiguously to measured resources. For instance, one might inquire whether the system console should be regarded as part of the CPU, part of main storage, or part of the I/O configuration, or apportioned somehow among those resources. For main storage and the CPU, parameters are given below for apportioning such overhead. In the individual costs below, it is possible to include apportionments that represent salaries, materials, space rental, etc. Alternatively, they may appear as a burden expense proportional to the direct machine costs.

In view of the arbitrariness of some of the measures presented, the purpose of such a mathematical analysis may be questioned. Not all the measures are arbitrary, and those that *are* have limited ranges of reasonableness. The main point is that the type of analysis illustrated is an orderly exposition of which measures are arbitrary, and it makes for clear recognition of the relationship of prices to costs and to policy decisions. We have also found it effective as a background for discussion of prices with users and with upper management, and for internal decisions in the computer center.

CPU

SMF estimates the time used by the CPU for each job step while the program is in the problem state. The CPU also accomplishes overhead functions such as job, task, and I/O management. It may be desirable to recover part of this overhead in proportion to I/O activity rather than in proportion to CPU time. In the other direction, part of main storage is occupied permanently by system control programs; therefore a desirable installation option is control of the proportions of main storage overhead recovered by assessment against CPU usage and I/O load. Accordingly, the adjusted cost of the CPU, r_c', can be expressed by the formula

$$r_c' = r_c(1 - x_c) + r_M x_M$$

where r_c is the cost per month of the CPU, x_c is the fraction of the CPU cost to be recovered as I/O overhead, r_M is the cost per month of main storage, and x_M is the fraction of main storage cost to be recovered as CPU overhead. The second term reflects the usage of the CPU for I/O overhead functions, and the third term reflects the portion of main storage assessed as part of the CPU.

main storage

Typically, part of main storage contains modules required for source and sink I/O support (e.g., HASP). If this part is assessed against I/O usage, then the adjusted cost of main storage, r_M', can be expressed by

$$r_M' = r_M(1 - x_M - x_{IOSUP})$$

Here, x_{IOSUP} is the fraction of main storage that supports external I/O devices. The second term compensates for the adjustments in the formula for r_c', above, and the third term reflects I/O support.

tape mounts

The cost of tape drives can be recovered partly by a mount charge and partly by I/O counts. The part of the tape cost to be recovered by tape mount charges, r_T', can be represented as

$$r_T' = x_T r_T$$

where x_T is the fraction of tape rental to be recovered by mount charges, and r_T is the cost per month of tape drives and tape control units.

unit record I/O

Where unit record I/O is measured, it is reasonable to include the cost of printers, card read-punches, and associated control units, and the cost of part of the multiplexer channel, of the BSC portion of a transmission control unit (TCU) used for RJE and associated telephone costs, and of direct access devices used for spooling. In addition, it is reasonable to include the main storage cost of the spooling programs. Thus, the adjusted unit record I/O cost, r_{pp}', can be expressed as

$$r_{pp}' = (r_{pp} + r_{BSC} + r_{SPOOL} + r_M x_{IOSUP}) \cdot d_u$$

where r_{pp} is the cost per month of printers and card read-punches, r_{BSC} is the cost per month of telephone equipment and the BSC portion of the TCU, r_{SPOOL} is the cost per month of spooling devices, and d_u is the fraction of unit record machine rental recovered from I/O count assessment. Paper costs should be included in estimating the fixed charge for print and punch I/O.

connect time

For terminal support, it is reasonable to include the cost of the terminals' portion of transmission control units, associated telephone equipment, and swapping or paging hardware. The adjusted connect-time cost, r_{con}', then, is

$$r_{con}' = (r_{ss} + r_{SWAP}) \cdot d_c$$

where r_{ss} is the cost per month of telephone equipment and the start-stop portion of the TCU, r_{SWAP} is the cost per month for swapping and paging hardware, and d_c is the fraction of transmission control rental recovered from connect time. Note that d_u and d_c can be calculated from summation of direct charges after an overhead ratio is included.

on-line storage

Long-term on-line storage is charged to the user at a fixed rate in a separate calculation. To avoid including the same resource twice, a deduction should be made from the cost of the peripheral equipment to reflect the portion of on-line storage assigned to user libraries. Thus r_{ST}, the long-term storage cost per month, is

$$r_{ST} = \left(\sum_{\substack{\text{device} \\ \text{type}}} c_{dev} m_{dev} \right) x_{ST}$$

where c_{dev} is the cost per magabyte-month for storage on a given device, m_{dev} is long-term storage on a given device in megabyte-months, and x_{ST} is the fraction of storage device cost recovered by long-term occupation.

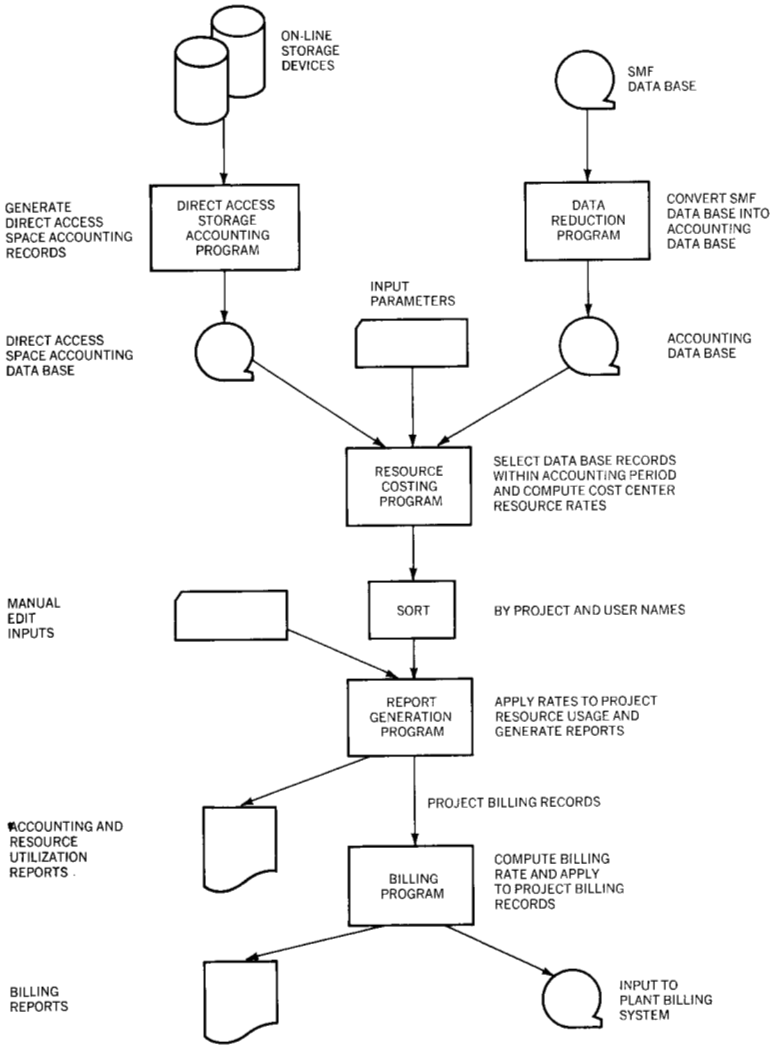
remaining I/O equipment

All remaining I/O equipment charges can be recovered by measurement of I/O activity. If the installation chooses not to charge directly for some of the resources addressed above, their cost will be included as part of the residual I/O costs (see formula for K_{EXCP} below). The total cost per month for the peripheral I/O configuration, represented by r_{IO}, is

$$r_{IO} = r_T + r_{BSC} + r_{ss} + r_{pp} + r_{SWAP} + r_{CH} + r_{SPOOL} + r_{DISK} + r_{MISC}$$

Here, r_{CH} is the cost per month of channels, drums, and control units not otherwise included, r_{DISK} is the cost per month of direct access storage excluding long-term storage, and r_{MISC} represents a catchall term for rental costs not otherwise included.

Figure 1 Overall structure of the accounting program



Adjusted for main storage and CPU overhead terms, the I/O rental cost is

$$r_{IO}' = r_{IO} + r_M x_{IOSUP} + r_c x_c$$

Then r_p, the cost of the I/O equipment not covered by fixed-rate charges, is

$$r_p = r_{IO}' - r_{ST} - r_T - r_{con} - r_{pp}$$

and the estimated rental cost of the entire system, r, is

$$\begin{aligned} r &= r_{IO}' + r_M' + r_c' \\ &= r_{IO} + r_M + r_c \\ &= r_p + r_{ST} + r_T + r_{con} + r_{pp} + r_M + r_c \end{aligned}$$

The formulas can be combined to give the fraction K_k of the system recovered against the charge measure k:

CPU $$K_{\text{CPU}} = \frac{r_c'}{r} = \frac{1}{r}(r_c - r_c x_c + r_M x_M)$$

Main storage $$K_{\text{MSTOR}} = \frac{r_M'}{r} = \frac{1}{r}(r_M - r_M x_M - r_M x_{\text{IOSUP}})$$

Unit record $$K_{\text{UREC}} = \frac{r_{pp}'}{r} = \frac{d_u}{r}(r_{pp} + r_{\text{BSC}} + r_{\text{SPOOL}} + r_M x_{\text{IOSUP}})$$

Tape mount $$K_{\text{TPMNT}} = \frac{r_T'}{r} = \frac{1}{r}(x_T r_T)$$

Connect $$K_{\text{CON}} = \frac{r_{\text{CON}}'}{r} = \frac{d_c}{r}(r_{\text{SS}} + r_{\text{SWAP}})$$

EXCP $$K_{\text{EXCP}} = \frac{r_p'}{r} = \frac{1}{r}(r_{\text{IO}} + r_M x_{\text{IOSUP}} + r_c x_c - r_{\text{ST}} - r_T - r_{\text{CON}} - r_{pp})$$

Note: $\sum_k K_k = 1$

At installation option, any parameters x or d can be set equal to zero.

Reduction of machine charges to dollars

For converting machine charges to dollars, the usage figures can be combined with costs as determined by the financial department. Also, adjustments can be made to exclude machine use by the computer facility for maintenance, since this is often considered an overhead item.

The dollar charge per job step, $\$_j$, can be calculated as follows:

$$\$_j = P_j + c_j \frac{E - \sum_j P_j}{\sum c_j}$$

Here, E is the total expense for the accounting period, including machine rental, labor, materials, and miscellaneous overhead expenses, c_j is the machine unit charge of the jth customer, and P_j is the dollar charge for resources billed at a fixed rate to the jth project. The set $\{c_j, P_j\}$ is not to include computer facility overhead projects.

Figure 2 Key accounting data base records

STEP RECORD

REC TYPE	MOD NO	SYS ID	JOB NAME	JOB READ TIME	JOB READ DATE	STEP START TIME	STEP START DATE	STEP ELP TIME	STEP CPU TIME	STEP COMPL CODE	STEP EXCPS	STEP CORE	TAPE MOUNTS	STEP NO
3	2	2	8	8	6	8	6	8	11	3	7	4	1	2

HASP RECORD

REC TYPE	MOD NO	SYS ID	JOB NAME	JOB READ TIME	JOB READ DATE	PAGE COUNT	LINE COUNT	PUNCH COUNT	USER NAME	PROJ NO	JOB CLASS
3	2	2	8	8	6	6	7	5	16	8	1

JOB RECORD

REC TYPE	MOD NO	SYS ID	JOB NAME	JOB READ TIME	JOB READ DATE	JOB START TIME	JOB START DATE	JOB ELP TIME	JOB CPU TIME	USER NAME	PROJ NO	JOB COMPL CODE	JOB CLASS
3	2	2	8	8	6	8	6	8	11	16	8	3	1

DIRECT ACCESS SPACE ACCOUNTING RECORD

USERID	RUN DATE	RUN TIME	DEVICE TYPE	NO OF DATASETS	NO OF TRACKS
8	6	8	4	2	7

Implementation

program structure and operation

The overall structure of the accounting program is depicted in Figure 1.

The data reduction program combines the SMF records, which may include installation defined records in addition to those provided by OS-SMF, and it drops data not needed by the accounting method, makes simple validity checks on the data and prints an exception list, and collects the job and step records of a single job. In addition, the program can combine SMF files from several systems. Each SMF record type is processed independently. Figure 2 summarizes the key output records, with a few of the less important details omitted.

We use installation defined SMF records generated by HASP to account for and describe all non-OS time. The data reduction achieved is approximately 50 percent of the original SMF records. The output accounting data-base tape (actually a multireel volume of indefinite length) contains *all* the transaction records and cannot itself be edited by the accounting programs. After a grace period to permit examination of the exception listings, the SMF input data are destroyed so that the corresponding section of the accounting tape cannot be reproduced. The accounting data-base tape must be stored as long as audit and tax regulations specify.

The direct access storage accounting program samples the online data base periodically, determining ownership of data sets from the first qualifier of the name (a convention which must be enforced by data-base maintenance procedures and programs). This program produces a data base that is similar to the accounting data base. These two data bases, along with the cost parameters, are the input to the resource costing program.

The resource costing program is structured (as is the data reduction program) with a separate procedure for each type of input record, so adaptation to new records and formats is facilitated. Input parameters include formula constants, accounting period dates, factors for service levels, and rates for fixed-rate resources. This program computes the denominators and prefactors in the formula for calculating the value of s_{ij}, on page 325. If all rates are fixed, this step can be omitted.

Following a sort by project and user names, the report is generated. We have done this on a weekly and monthly schedule. Figures 3 and 4 indicate the output format. Although not shown, all manually entered records and input parameters should be printed on the output. These records include adjustments for rerun credit (job aborted because of computer facility error) and administrative adjustments such as one-time charges for new hook-ups and bills for dedicated equipment. To provide timely information for users, the report necessarily estimates the costs. Later, when financial information for the accounting period is available, the billing program generates precise charges based on the cost of operation. At this time, input of suitable format for the corporate accounting system is generated. (Note: numbers shown in the figures are for illustration only.)

reports and their use

Figure 3 shows the part of the report that summarizes system activity and gives a dollar breakdown. Although most of the information is self explanatory (as it most emphatically should be), a comment is in order: system performance information is useful not only to persons in a position to do something about it, but to the user community at large. We found that a brief summary of machine up time and utilization, published for the user community on a regular basis, makes for good "customer relations." Thus, when system performance and reliability are good, everyone can see that it is; and when it is not good, the pressure (even embarrassment) of public knowledge provides a powerful incentive to make things better.

The breakdown of usage by departmental groups is useful in at least two ways: it can show user groups that they are getting their rightful share of the resource (as properly determined by upper management), and it allows computer facility overhead to be compared with the quantity of resource delivered to more directly productive work.

Figure 3 Example of a summary report

METER TIME IS FROM MAY 15, 1973 TO JUNE 22, 1973 SJRL SMF V3 2/1/72

ACCOUNT PERIOD 09:00:00 73:135 TO 08:59:59 73.173

ELAPSED TIME	911:59:59			
METER TIME	0671.44.40	= 73.6% OF E.T.	= 2418280	MACHINE UNITS
MACHINE DOWN TIME	7:19:45	= 1.3% OF E.T.		
PM TIME	0016.00.00	= 1.7% OF E.T.		
TOTAL ACTIVE	545:53:33	CPU UTILIZATION 81.2%		

PROJECT STATISTICS:

GROUP NAME	JOBS	CPU TIME HH:MM:SS	EXCPS	CORE MBS	% OF TOT MU	FIXED CHARGES	TOTAL CHARGES
PROJECT A	2840	134:40:44	082246	600158	16.5%	$11055.55	$127368.45
PROJECT B	761	1:03:47	904678	23384	0.5%	$3336.80	$6642.25
PROJECT C	3241	11:14:42	12338120	157029	4.7%	$21291.93	$54576.93
PROJECT D	117	0:22:54	137889	3418	0.1%	$675.20	$1248.65
PROJECT E	840	1:17:13	317549	17788	0.3%	$3945.33	$6066.68
PROJECT F	592	2:04:21	661334	12390	0.4%	$3012.17	$5592.82
PROJECT G	253	0:53:15	223822	13318	0.2%	$1545.99	$3087.09
OTHER	3873	106:28:38	12049072	368156	11.6%	$15258.09	$96822.44

RATES:

RESOURCE	RATES	UNITS
CPU	0.34911	MU PER CPU SEC
CORE	0.22426	MU PER MBYTE SEC
EXCP	0.00525	MU PER EXCP
CONNECT	$8.00	DOLLARS PER HOUR
UNIT RECORD I/O	$0.0015	DOLLARS PER LINE
TAPE MOUNTS	$3.00	DOLLARS PER MOUNT
DIRECT ACCESS STORAGE		
2314	$63.00	DOLLARS PER MEGABYTE MONTH
3330	$63.00	DOLLARS PER MEGABYTE MONTH
2250 TIME	$0.00	DOLLARS PER HOUR

CLASS	TOTAL E.T. HH:MM:SS	ACTIVE HH:MM:SS	EXCPS	CORE MBS	JOBS	SHIFT 1ST	2ND	3RD	% OF TOT MU	PAGES	LINES	CARDS
A	213:05:41	8:35:11	7139294	69212	6852	4384	1387	1081	2.6%	266813	12575740	104919
B	76:18:07	11:00:28	2538017	30366	1118	624	319	175	1.4%	38389	1689553	1330
C	61:21:53	8:59:34	1133541	15680	548	294	141	113	0.9%	19417	853264	8602
D	169:36:08	18:47:40	12709810	107325	3708	2267	1075	366	4.7%	143008	5985327	26041
E	163:34:16	18:24:18	10876956	77314	4263	2571	1199	493	4.0%	112467	4883418	109734
F	31:13:48	2:25:20	944417	17505	415	266	97	52	0.5%	9686	442125	9
G	270:20:23	23:08:18	9532834	148583	4413	2966	874	573	4.6%	220543	9401991	41899
7	152:04:36	31:50:07	8685182	236086	77	1	12	64	5.7%	3818	205403	1853
8	168:18:23	45:09:12	5724499	229893	50	0	4	46	5.7%	3399	163127	1169
9	93:17:23	21:58:06	1201579	111372	34	0	8	26	2.4%	17109	988290	2632
RRUN	20:48:46	2:50:28	751869	22876	184	77	23	84	0.5%	999	39766	350
TOTALS			126349633	3058450	35728	22326	8530	4872		1362059	59602165	563240

SHIFTS	JOBS	% OF TOTAL JOBS	CPU HH:MM:SS	% OF TOTAL CPU	EXCPS	CORE MBS	TAPE MOUNTS
1ST	22326	62.4%	125:25:14	22.9%	40528972	998272	3421
2ND	8530	23.8%	182:58:14	33.5%	37341582	843580	3895
3RD	4872	13.6%	237:30:05	43.5%	48479079	1216599	2622
TOTAL	35728	99.8%	545:53:33	99.9%	126349633	3058450	9938

DIRECT ACCESS SUMMARY

DEVICE	TRKDAYS	BILLED MBYTE MONTHS	BILLABLE MBYTE MONTHS	% BILLED	RECOVERY
2314	319678	377.68	526.93	71.7%	$23632.31
3330	1735277	753.11	2062.13	36.5%	$47445.31

RUN DATES

73.135
73.143
73.172

Figure 4 Example of an individual user and project report

PROJECT NO. AAA-1234 | PROJECT NAME INFO. SYS. ARCH | PROJECT MGR. E. MCFURD | SJRL SMF V3 2/1/72

PROGRAMMER

METER TIME IS FROM MAY 15, 1973 TO JUNE 22, 1973
ACCOUNT PERIOD IS FROM 09:00:00 73.135 TO 08:59:59 73.173

A. SMITH	CLASS	1ST	2ND SHIFT	3RD	ELP TIME HH:MM:SS	CPU TIME HH:MM:SS	EXCPS	CORE MBS	UNIT RECORD I/O PAGES	UNIT RECORD I/O LINES	CARDS	TAPE MOUNTS
	D	33	16	0	1:59:06	0:08:36	74466	815.37	1382	60351	0	0
	E	15	6	0	0:46:51	0:01:48	14698	205.90	2264	89637	0	0
	A	10	7	0	0:09:50	0:00:03	2803	69.42	172	5737	0	0
	H	3	0	0	0:17:05	0:03:31	4737	144.76	83	3564	0	0
	U	13	0	0	0:19:06	0:00:13	2 46	61.88	13	777	0	0
	TSO	33	12	0	58:09:43	0:23:12	3350	8660.78	0	0	0	0
	TOTAL	123	54	0	65:19:45	0:47:36	212340	11600.40	5566	253090	0	0

ABEND	NUMBER	PCT OF TOTAL CPU TIME
B0A	4	0.21%
03B	2	0.01%
913	1	0.01%
622	2	1.01%
TOTAL	53	2.11%

DEVICE	DATASETS	TRK DAYS
2314	5	16720
3330	26	8774

P. SMYTHE	CLASS	1ST	2ND SHIFT	3RD	ELP TIME HH:MM:SS	CPU TIME HH:MM:SS	EXCPS	CORE MBS	UNIT RECORD I/O PAGES	UNIT RECORD I/O LINES	CARDS	TAPE MOUNTS
	TSO	2	2	0	1:28:25	0:00:30	260	178.73	0	0	0	0
	TOTAL	2	2	0	1:28:25	0:00:30	260	178.73	0	0	0	0

DEVICE	DATASETS	TRK DAYS
3330	2	389

133

PROJECT TOTAL	SHIFT	JOBS	ABENDS	CPU TIME HH:MM:SS	EXCPS	CORE MBS	MACHINE UNITS
	1ST	125	33	0:36:11	161055	9073	3638
	2nd	56	20	0:11:55	51545	2707	1127
	3rd	0	0	0:00:00	0	0	
	TOTAL	181	53	0:48:06	212600	11779	4765

COST CENTER RATES

RESOURCE	RATE	UNITS
CPU TIME	0.34911	MACHINE UNITS/CPU SEC.
CORE	0.22400	MACHINE UNITS/MEGABYTE SEC.
EXCPS	0.00525	MACHINE UNITS/EXCP

FIXED RATE CHARGES
RESOURCE

CONNECT TIME 59.636 HRS AT $8.00 PER HR = $476.27
UNIT RECORD I/O 253090 LINES at $0.0015 PER LINE = $378.54
DIRECT ACCESS SPACE
16720 TRK DAYS ON 2314 = 4.0629 MBYTE MONTHS AT $63.00 PER MBYTE MONTH = $255.32
9181 TRK DAYS ON 3330 = 3.9845 MBYTE MONTHS AT $63.00 PER MBYTE MONTH = $249.67

TOTAL ESTIMATED CHARGES

TOTAL FIXED RATE CHARGES $1359.80
COST CENTER CHARGES 4765 MACH UNITS = 1.324 HRS AT $1050.00 PER HOUR = $3902.40
TOTAL ESTIMATED CHARGE TO PROJECT AAA-1234 =$15262.20 ± 5%
SUBJECT TO COST CENTER FLUCTUATIONS

The breakdown by job class provides a 24-hour-a-day picture of system utilization. Trends in amount and types of usage can be identified, so job class distinctions can be refined, hardware can be ordered or released, and so forth.

The report illustrated in Figure 4 is sent to each project manager, enabling him to see at a glance which persons in his project are doing what kind of work. The cost center rates are printed here as a convenience, to facilitate future planning as well as to help in figuring the values of various usage trade-offs.

future plans

In retrospect, several aspects of the implementation could bear improvement: The output from the direct access accounting program should be merged with the output from the data reduction program, thereby reducing the physical size of the accounting data base, the amount of clerical support needed, and the opportunity for error. For better security, it would be preferable that the manually entered records and the input parameters be entered into the accounting data base, thus consolidating the accounting information in a single untamperable data set. Also, our implementation did not take into account that a user assigned to one project might legitimately charge runs to another project.

The reports generated would be more helpful if year-to-date totals for selected information were provided. To go a little further, budget tracking could be nicely provided for with a plotted curve of cumulative dollars spent versus a planning line for the various projects.

Although the publishing of cost center rates in the user report is helpful, it could be carried a step further: a brief summary of the charge algorithm could be provided so that the user need not refer to separate documentation when verifying or analyzing his charges.

Our future plans call for a substantial revision of this accounting system. The minor revisions discussed above can be incorporated, but major changes in our computing environment dictate a major rewrite. The computing facility has undergone a massive consolidation, integrating some dozens of systems and several major cost centers, with a flexible netting and load sharing capability among many of the machines. Not all the accounting ramifications of these changes are yet understood, and we are looking forward to a period of challenge and new enlightenment.

performance

No attempt has been made to optimize the program's performance. To process a monthly input generated by 30,000 jobs, the program typically requires four minutes of CPU time on the

Model 195 and 40,000 EXCPs in a 300K-byte region. A new version could be much more economical.

Concluding remarks

Administering the allocation of a computer resource can be segmented into several responsibilities, with fairly well defined subjects of common concern: the policy of the parent organization and strategies to realize the overall objectives by allocating computer resources and assessing results; tactics of users to maximize value per unit price; control functions to demonstrate compliance with externally imposed regulations; planning and assessment within the computer facility; and technology of load measurement. It is possible to design an installation accounting package that does much to decouple these varied functions. We have described an implementation that is quite successful in this regard.

It is possible and desirable to design and implement a single accounting package that would be acceptable for the majority of medium to large installations, and that would have a well defined set of options and alternatives and clear identification of areas in which installation-written procedures could be added to implement unusual policies. The program should be structured to take advantage of new measurement methods and to support new services. We believe that the implementation presented here is valuable as a guide for an entirely fresh design.

ACKNOWLEDGMENTS

For constructive criticism over several years, our users. For the installation-defined SMF records, K. G. Field. For a preliminary plan and implementation, W. H. Kwok. For constructive criticism of the manuscript, G. M. Giddings, W. Doherty, D. N. Streeter, and F. P. McAllister. For manuscript preparation, J. M. Valdillez.

REFERENCES AND FOOTNOTES

1. S. J. Golin, "Resource pricing in a multiprogramming environment," *Installation Management Review* **3,** 1, 3–11 (January 1974).
2. N. R. Nielsen, "The allocation of computer resources—is pricing the answer?" *Communications of the ACM* **13,** 467–474 (1970).
3. N. R. Nielsen, "Flexible pricing: an approach to the allocation of computer resources," *AFIPS Conference Proceedings* **33,** 521–531 (1968).
4. F. W. McFarlan, R. L. Nolan, and D. P. Norton, Chapter 11, *Information Systems Administration,* Holt, Rinehart and Winston, New York, New York (1973).
5. W. F. Sharpe, Chapter 11, *The Economics of Computers,* Columbia University Press, New York, New York (1969).
6. N. M. Singer, H. Kanter, and A. Moore, "Prices and the allocation of computer time," *AFIPS Conference Proceedings* **33,** 493–498 (1968).

7. For discussion of value-to-user, see D. N. Streeter, *The Scientific Process and the Computer*, John Wiley and Sons, Inc., New York, New York (1974).
8. R. C. Rettus and R. A. Smith, "Accounting control of data processing," *IBM Systems Journal* **11**, 1, 74–92 (1972).
9. See, for example, R. A. Watson, *Computer Performance Analysis: Applications of Accounting Data*, RAND Report R-573-NASA/PR, RAND Corp., Santa Monica, California (1971).
10. A copy of the specification is available from the authors.
11. *OS SMF*, IBM Systems Library, order number GC28-6712, IBM Data Processing Division, White Plains, New York.
12. A controversial economic, accounting, and legal question concerns the extent to which prime-time price differentials and premium prices for priority service are permissible in installations with some government contract work or whose prices become a significant cost component in the pricing calculation of a corporate product. There is an interesting and little known ruling in Defense Contract Agency Regulation 7640-9, as reported by H. Kanter, A. Moore, and N. Singer in *The Journal of Business*, July 1968, pages 383–388: "Where real cost differentials . . . exist . . . separate rates for such cost differentials may be used. . . . Where rental or lease costs are based on prime shift usage, second and third shift usage may, with appropriate approval, be charged at reduced rates. . . . Such differentials would permit . . . priority . . . runs at premium rates and/or nonpriority or nonprime-time or large-volume runs at reduced rates. . . . Where differing rates are used, they should be applied . . . on a nondiscriminatory basis."
13. *OS/VS System Management Facilities (SMF)*, IBM Systems Library, order number GC35-0004, IBM Data Processing Division, White Plains, New York.
14. Occasionally the assumption is made that the capacity of a computer installation is a known, inflexible quantity, making it possible to determine a unit cost for each resource (see, for example, reference 8). In our experience, such an assumption is misleading.
15. H. W. Lynch and J. B. Page, "The OS/VS2 Release 2 System Resources Manager," *IBM Systems Journal* **13**, 4, 274–291 (1974).

Reprint Order No. G321-5019