Current bank modeling systems are generally based on sets of equations that place limitations on the flexibility of the applications and the predictive ability of the model.

The experimental system discussed in this paper uses a threefold approach to the simulation of actual banking activities. A generalized bank modeling system is presented from which a subset is selected for use. The user interacts with the modeling system via an interactive simulation language. The composition of the desired model is determined interactively via a terminal and an interactive model generator.

Research results indicate that representative models can be generated by using these techniques.

Interactive simulation for banking

by J. F. Brown and D. W. Low

During the past decade, the financial industries—and banks in particular—have been major users of management science and operations research techniques. Bank managers continually face some form of the following imperative: Invest the bank's resources, always assuming some degree of risk, so as to maximize shareholder wealth while complying with federal and state regulations for protecting depositors. This assignment is a technically difficult one because of the large number of variables and the structure of the constraints involved.

Many useful results have been achieved through management science and operations research. Still bank researchers face the problem of new project directions for their institutions to pursue. Banking models have generally tended to guide the production of returns on investments that have perhaps been less than anticipated. Some banking researchers have rejected proposals for more complex model developments because of the belief that marginal gains expected of a more comprehensive proposed model would not justify the cost of producing it. Similarly, proposals have been excluded from the budget on grounds that they are not representative of the real world or that the technique is applicable only to problems of limited dimension.

We have been studying this problem at the IBM Los Angeles Scientific Center with the hope of improving bank modeling. Our approach has been to devise a general-purpose bank modeling system that uses a programming-by-questionnaire technique

by which we have simulated banks in a research environment. The programs themselves, which are experimental and research oriented are, therefore, not discussed and are not available for distribution.

The bank modeling system discussed in this paper simulates such major elements as reserves, investments, loans, borrowing, deposits, and capital accounts. The system with which we have been experimenting is designed so that the user, whether he is a managerial or technical support person, participates in the bank model building in an interactive question-and-answer mode. In this way, he constructs a digital simulation of the bank's operation.

Our method extends the most useful features of such bank modeling systems as that discussed in Reference 1, by incorporating simulation to produce a more realistic modeling technique. The overall intention is ease of use by bank personnel through such capabilities as the following:

- No user programming.
- User-system communication via banking terminology.
- Ease of learning.
- Ease of producing and modifying banking models.

Financial modeling of banks in perspective

current systems

We now briefly describe the overall structure of typical bank modeling systems of the current generation, and the structure of the system we have been studying. The chief assumption implicit in current systems that potentially affects their validity is that the model consists of a set of equations in which the user essentially provides data for the constants. In many bank modeling systems, these data are key financial and planning data. The user enters such categories of asset (or liability) management to be studied, as, for example, the balance sheet categories of Treasury Bills, secured loans, and Federal Reserve Board borrowing. Dollar amounts currently allocated to these accounts are obtained from a current balance sheet. Prime rate, Treasury Bill discounts, available maturities, and other applicable factors in the banking environment are entered. The user also provides the length of a typical planning period and the number of periods in the planning horizon.

A typical system then asks the user for the operating decisions to be made during the next period. One such decision might be the proportion of assets to be placed in Treasury Bills and in commercial loans. The system then calculates net gains and/or losses in each category for the next period and then projects a new balance sheet. Of course, other reports may also be project-

ed and computed on the basis of a given set of transition equations. At this stage, the user may change the set of decisions to be projected or he may proceed to the next period.

The approach just outlined implies certain assumptions about the external environment and the internal bank structure. One such assumption is that the banking environment is predictable: the discount rate is constant; no loans default; and no unplanned demand deposit activity occurs during the planning period. New types of bank structures appear not to be readily simulated by current systems. Also, there seems to be no easy provision for the introduction of such new services as Ready-Reserve, Balance Plus, and Master Charge. Costs associated with the various banking activities and services are given by a set of equations for which the user supplies values of coefficients. Both the internal and external environments remain constant during a planning period, and changes are introduced at the beginning of a period.

experimental systems

This summarizes many bank modeling systems currently used, from the point of view of the user. We now summarize the system we have been studying. The primary difference between the two approaches stems from a methodology by which most current systems use transition equations to move from one balance sheet to the next, whereas, we have been investigating the discrete simulation of modeled banking activities. Banking operations are described in our system in terms of discrete events of deposits, withdrawals, granting of loans, investing in Treasury Bills, and outside borrowing. The user particularizes these activities for his bank, and the system keeps track of the resulting cash flow. A simulated balance sheet is computed by examining each of the various asset and liability accounts. These discrete banking events are simulated asynchronously in our system, just as they may in a real bank. Thus, in the system we have been studying a bank is modeled as a set of banking activities and renders unnecessary the need for the assumptions required by equations.

Our banking studies can include planned and/or unplanned elements. For example, the user may wish to see how his policies handle the unexpected loss of a large depositor. Also, a standard bank structure has been embodied (as in current systems), which may be enlarged, simplified, or replaced. This means, for example, that we can easily adapt our system for modeling foreign banks where different laws and customs normally cause a reprogramming effort. We can compute, from predefined formulas (as in current systems) or from user-defined formulas, costs associated with such banking activities as the granting of a large commercial loan or accepting a new depositor. We can also accommodate banking activities and changes in the environment

that can occur periodically, or at randomly spaced intervals, or as the result of another activity or change. For example, an unexpected drop in the prime interest rate may require a management review that results in a change of the interest rates on savings accounts. This, in turn, may change the average rate at which customers deposit and withdraw money in these accounts.

There are three principal elements in the approach we have studied. First, we have designed our system around a generalized bank model, only a subset of which may be required to model any given situation. (For other models, see References 1, 2, and 3.) Also included in our studies is a simple, interactive simulation language with which a user, if he wishes, can expand and enrich the model generated by the system. The third principal element of our system is an interactive model generator that analyzes user responses obtained during question-and-answer sessions. From this analysis and from the general model, the system selects specific relevant components for a subset model. It should be noted that the basic concepts of programming by questionnaire are not new. See References 4 and 5. Our use of these techniques in an interactive environment is believed to be new. In the following sections, we describe each of these system components.

Generalized bank model

In our modeling studies, a bank is viewed as being represented by its balance sheet or statement of condition, and by basic elements or entities in the banking model, which are the accounts. In our general use of the term, *accounts* are defined as holding points for collections of a bank's assets and liabilities. These collections may be described as *portfolios* of individual instruments, or as single sets of attributes of aggregated combinations of items. Each account can have its position reviewed, and, as a results, the system user can take direct actions such as buying or selling asset items for or from its portfolio or it may request that other accounts review their positions and make desired adjustments. Accounts can have target positions or budget goals that are the basis for management review actions as previously mentioned. These goals may be set simultaneously across several accounts in order that the objectives of the bank as a whole may be achieved. Finally, each account is considered to be a profit center of the bank, so that income, expense, gains, and losses may be collected separately for each account.

general accounts

The general account concept is classified into the following categories or types: Cash, Investments, Loans, Deposits, Borrowings, Capital Accounts, and Other Accounts. Whereas, in some

applications there may be a one-for-one correspondence between a real bank's general ledger and the accounts of a model, this need not be the case. That is to say, the classifications just given represent the types of actions and events that occur at an account or reflect the nature of the items held in an account portfolio, rather than representing a particular accounting convention. For example, it is up to the modeler and the nature of his problem into which asset category he places such a specific account portfolio as Securities Purchased Under Agreements to Resell. If one were developing a large-scale planning model, this portfolio would probably be considered as Investments. However, in a detailed model of a bank's activity in the money markets, where the individual placement of funds with correspondent banks is important, that portfolio might be modeled as Loans. Similarly, certain types of Certificates of Deposit might be categorized as Borrowings rather than deposits in many applications. Reports can be defined so that ambiguities are removed from the model and its output displays. The following brief descriptions characterize the general account types.

Cash accounts hold the primary reserve balances of a bank and are distinguished primarily by the way in which they are managed. That is, a cash account does not buy or sell other types of assets but may, as a result of a review of its position, request some other asset account to do so.

Investments are the basic portfolio-type accounts that buy, sell, and hold quantities of individual investment issues in a simulated marketplace. The system allows the accumulation of information on the purchase price and date of each item in the portfolio. Individual transaction control information such as maturity, risk, capital gains (or losses), and profit are also included. Price deviation from expectation are accumulated so that possible arbitrage and speculation can be modeled.

Loans are designated as assets because of the nature of their marketplace. That is, each loan account interacts with a loan application queue that holds the current demand for loans. Items in a loan queue have an option life (or maturity) after which, if not acquired by the appropriate account, they expire. If loans held by the bank are sold, they may not be repurchased.

Deposit accounts are the mosaic pieces of the Deposits picture of a bank. Deposits are not directly controlled by the bank, but their balance levels and/or portfolio attributes may be accessed by other accounts to aid in decision processes.

Borrowings accounts are essentially general-purpose controlled (or controllable) liabilities accounts. For example, they may be used to model money borrowed from the Federal Reserve Bank

and federal funds, or money borrowed because the bank's projections have gone wrong. They interface with a Borrowings market and hold a portfolio of future cash outflow requirements of the bank.

Capital Accounts are used primarily as capital stock references at the start of a simulation and as records and accumulation points of income, expense, gains, losses, and various reserve changes in the other accounts. Capital Accounts are defined and used primarily for formatting and displaying simulation output.

Other Accounts provide bookkeeping completeness to meet special needs, and our interactive simulation language is capable of defining such accounts. Our modeling program experiments make a provision for fixed assets in this category that involve depreciation and have user-defined expenses.

environmental events

Through the specification of accounts, instruments, and markets, a series of environmental event structures are implicitly defined. Typical events are the arrival of deposits, withdrawal of funds, generation or issue and maturation of investment instruments, and changes in interest rates and prices in the various markets. These and other events are automatically modeled so as to maintain the basic balance sheet structure. Thus when deposits arrive at the bank, the balance (or portfolio) of the appropriate Deposits account is increased. At the same time (depending on model specification) the appropriate set of cash accounts is also increased in balance or in funds of deferred availability. Similarly, when an investment matures, the asset accounts are searched to determine which, if any, hold the instrument in their portfolios. The instrument is removed, destroyed, and the appropriate cash accounts are debited. We now briefly discuss the major types of environmental events that can be included in a model.

Deposit events characterize the ways in which monies are deposited and withdrawn from a bank. The banking system user may identify several different deposit structures in which each structure has special deposit time series, withdrawal time series, deposit (withdrawal) amount characterization, and deposit make-up in terms of, say, coin and currency, on-us checks, and out-of-town checks. Time policy options include periodic deposits and withdrawals—generated by a probability function—and/or user-defined deposits and withdrawals. Internally, a deposit event is handled by crediting the appropriate deposit account and by debiting one or more asset accounts. In a simple structure, a single account called CASH might be debited. In a more complex extension of perhaps the same model, a portion of the deposit amount might flow into COIN AND CURRENCY, another into an ON-US RECONCILIATION account and still a third portion into CASH ITEMS IN COLLECTION. At later times in a

simulation defined by the user, the ON-US RECONCILIATION might be credited against other withdrawals, and the CASH ITEMS IN COLLECTION might turn up in correspondent and Federal Reserve accounts.

Loan demand may be simulated for as many types as the user requires in his environment. Loan demands may be as simple as the generation of a single amount to represent an entire month's consumer installment loans or as complex as one event that corresponds to a single application for a large commercial loan. For each type of loan demand generation, the user may specify a time policy, amount policy, maturity, pay-back schedule, and risk model. The system places these applications in the loan demand queue of the appropriate loan account. Loan applications from the queue are accepted or rejected by the bank in accordance with policies determined by the bank management.

Investments are characterized in much the same way as loans. Principal differences occur in the way the holding accounts are managed and in the interpretation of the market structure. (Some loans cannot be sold, for example.) Our structure, in the investment case, assumes that particular instruments may be bought and sold from time of issue to their maturity. Loans, however, must be accepted or rejected shortly after their generation, and, once sold, are not repurchased. A key feature of the investment characterization is its flexibility in investment pricing. The user is given a menu of yield curves to select from, plus the option of adding a stochastic pricing error to each. Both bid and ask price structures may be modeled, and the user may define events that change the yield curve parameters so as to model an environment wherein the underlying economy is changing. If the yield curve structure does not satisfy the user's need, the user language enables him to specify his own price structure.

Economic considerations may be added for deposit supply, loan demand, and the various other investment markets. Such a capability permits the sophisticated user to develop models wherein many individual structures are related. In this way, one may describe a relationship between, say, a deposit supply generator and a loan demand model, or between all the interest rates and the supply of money to the bank. These types of considerations are possible through the specification of similar sources of deposit supply and loan demand interacting with conditional-event time policies and common-amount, time, and review policies.

management policy simulation

A key feature that we believe is important is a provision for the simulation of management policy. We have accomplished this through the incorporation of the following four types of review events: account review, audit, rebudget review, and control parameter review and update.

Account review events are types of management events that simulate the day-to-day decision-making process at an accounting center. Of concern in its specification are considerations of how often the account should be reviewed. Also of concern is the size of adjustment, if any, and whether it is to be made to its balance or to its portfolios, and how the adjustment is to be implemented. Slightly different event structures are provided for each of the following general accounts: Cash, Investments, Loans, and Borrowings. Account review structure may vary among accounts of a similar class. For example, DUE FROM FEDERAL RESERVE and VAULT CASH are both considered to be cash type accounts, but they may have different account review formats. There may be several account reviews for each account.

Account review events are specified by the user for Cash, Investment, Loans, and Borrowings accounts as required. Basically the account review checks an account portfolio against certain control parameters to determine what action, if any, is required to keep the account in control. For example, a loan account might be reviewed to see if it has written a prescribed quantity of loans since its last review. If not, obtaining as many new loans of the type it deals in as required or available may correct the situation. Similarly, a cash account balance may be checked against a precalculated target budget (plus or minus a reserve). If the cash account is out of control, a corrective response may be to access a source or application account list to place or generate required funds. In this light, the system allows for two modes of account review, free and directed. In the free mode, the account review computes the amount of balance adjustment required to bring the budget back under control or to meet budget goals. In the directed mode, the review is called with a quantity parameter, and this quantity is generated or used insofar as it does not put the reviewed account out of control. Thus a cash account review might call an investment account review to generate a needed amount of cash. An investments review results in the selling of investments to raise the required cash until such sales violate some investment control constraint or the investment portfolio is empty. The value of requested funds remaining to be raised is entered into the calling account for appropriate action.

Audits augment account reviews. Whereas, account reviews deal primarily with individual accounts, it is recognized that there is a great deal of account, market, and instrument interdependence associated with proper bank management. To meet these needs an *audit* is specified by the user to check simultaneously several management constraints over several accounts. For example, liquidity, capital-adequacy, and minimum-marketable-portfolio criteria may be simultaneously and periodically checked for a

given group of accounts. The result of such an audit might confirm that the bank is in control, or that a new budget is required. The sale of risky investments may be indicated or the simulation may stop, display the audit results, and await instructions from the user.

Rebudget event reviews are often called as the result of an audit to establish new budgets or goals for specified accounting centers. Typically, these are changes in certain portions of investment or loan portfolios. A rebudget review should also have options for reestablishing goals. Included in the current set of options is a rebudgeting technique based on a single-period, asset-allocation linear program, which—if selected—produces a logically best set of goals for the accounts under consideration.

Control parameter updates and review are events used for specifying a management structure. In addition to account balances and other attributes, the user is allowed to simulate management decisions based on forecasted and/or historical data. By selecting options from the control parameter review menu, the user specifies that he wants the test of an account review or audit to be based on seasonally adjusted, exponentially smoothed forecasts of loan demand. The user may specify that his yield curve estimate is a logrithmic function of days to maturity. Additionally in defining the event, he specifies how often he wants forecasts updated or constraint parameters replaced.

Interactive simulation language

We have been experimenting with an interactive simulation language that contains elements similar to other procedural languages plus elements oriented toward the particular problems of interactive financial modeling. The basic unit of simulation is an *event*, which is a subroutine scheduled to be executed at a particular simulation time. Events may be scheduled or canceled as with existing discrete-event simulation languages. The passage of simulated time is quantized between events, and a simulation time is the time at which each event is activated. Events are defined by the user and consist of both procedural statements and statements that define local and parametric variables for the event. Global variables may be defined and variables may be aggregated into data organizations called *trees* in a manner similar to PL/1 or COBOL structures. Our interactive simulation language has few types of statements, but it addresses the following general problem areas of interactive simulation: graphic output, time series, and checkpoint and restart.

Graphic output becomes increasingly important as model complexity increases and as potential output from models becomes

voluminous. Our interactive simulation language includes a PLOT statement, and that further assumes the availability of a graphic display device.

Time series are often desirable in a simulation environment to record the value of a variable at points in time and then to manipulate or to inspect the series of recorded points. For some variables, it is also desirable to predict or forecast a series of values of the variable into the future. Therefore, at a given simulation time, any variable may consist of the following two parts: (1) the current value of the variable, and (2) the series of data points that represent past values of the variable that have been recorded, and/or the series of points that represent forecasted values of the variable.

The user, through assignment statements, may set values for any current, past or future values of any variable. The PLOT statement is used to produce graphs of time series. Arithmetic operations may be performed on single points or all points within a time series.

Checkpoint and restart eases the game-playing capability of our bank modeling system. Many model studies are useful, not so much for obtaining absolute values in digital output form as for gaining insight into the complex interactions between various components of the model. Typical of this type of modeling is the situation where the user performs a "what-if" interaction with a model. If that simulation path does not prove fruitful, the user tries other possibilities starting at that same point. Our experience suggests a language that permits the user to checkpoint and restart the model using SAVE and LOAD statements that are similar to corresponding functions in APL. Note that SAVE and LOAD statements may be contained within events or may be executed interactively. The SAVE statement saves the model as a file in an internally formatted workspace.

example

We present now a small section of a bank model. The user first represents the balance sheet structure as follows:

```
1 ASSETS
  2 CASH
  2 INVESTMENTS
    3 LOANS
    3 SECURITIES
  2 EXPENSES
    3 INTEREST SAVINGS
    3 INTEREST BORROWINGS
    3 OVERHEAD
```

```
2 LIABILITIES
  2 DEPOSITS
    3 CHECKING
    3 SAVINGS
  2 BORROWINGS
  2 INCOME
    3 INTEREST LOANS
    3 INTEREST SECURITIES
  2 RETAINED EARNINGS
```

He similarly represents liquidity requirements as follows:

```
1 CONTROLS
  2 MINIMUM CASH ON HAND
  2 LIQUIDITY
    3 OVERLOANED AMT
    3 LOANS TO DEPOSITS RATIO
    3 LOANS TO DEPOSITS RATIO LIMIT
```

At the end of each banking day the user determines whether the LOANS TO DEPOSITS RATIO is within preestablished bounds by entering the following statements into the simulator.

```
LOANS_TO_DEPOSITS_RATIO ← LOANS
  ÷ (CHECKING & SAVINGS)
IF LOANS_TO_DEPOSITS_RATIO < LOANS_TO_DEPOSITS_
  RATIO_LIMIT THEN GO TO DONE
OVERLOANED_AMT ← LOANS − (LOANS_TO_DEPOSITS_
  RATIO_LIMIT × (CHECKING & SAVINGS))
PRINT 'LOANS_TO_DEPOSITS_RATIO EXCEEDS UPPER LIMIT'
SCHEDULE MANAGER
DONE:
```

The event named MANAGER may contain many different options including manual interaction at the terminal. The user can attempt to correct a condition by scheduling an event such as FORECLOSE. As the user gains experience from manual intervention, he can determine—as a matter of policy—alternative actions to take. He then replaces the SCHEDULE MANAGER statement with SCHEDULE LOANS TO DEPOSIT OUT OF BOUNDS, where the new event contains simulation language statements required to implement automatically the desired policy. It is expected that a gradual evolution from manual interaction toward automatic policy implementation continues as the user gains insight into the probable consequences of policies.

Interactive applications generator

Our research indicates that an interactive simulation language can facilitate the execution, testing, and modification of a bank

model. We have also studied a technique that can aid the user greatly in the original construction of his model. We refer to an interactive applications generator with which we have been experimenting and which virtually eliminates the need for programming on the part of the user.

When modeling studies are undertaken, one approach is to code highly specialized programs that are designed to represent in depth a fairly narrow portion of a business. This often results in a very limited degree of transferability to other business areas that vary from the primary model in one or more significant details. On the other hand, when general-purpose application packages are constructed for use in modeling many different structures, oftentimes the result is a set of programs that consume significant quantities of storage space and execution time by repeatedly determining the model structure from input parameters.

Our approach to modeling is a natural extension of the programming by questionnaire method discussed in References 4, 5, and 6. Our interactive applications generator is a decision table processor that guides the financial model user through a complex logic network by asking structured questions in banking terminology. At each state, the succeeding question depends on previous responses, and whole areas of questioning are eliminated by each response. Such interaction makes possible the creation of a model during one or more sessions at a terminal.

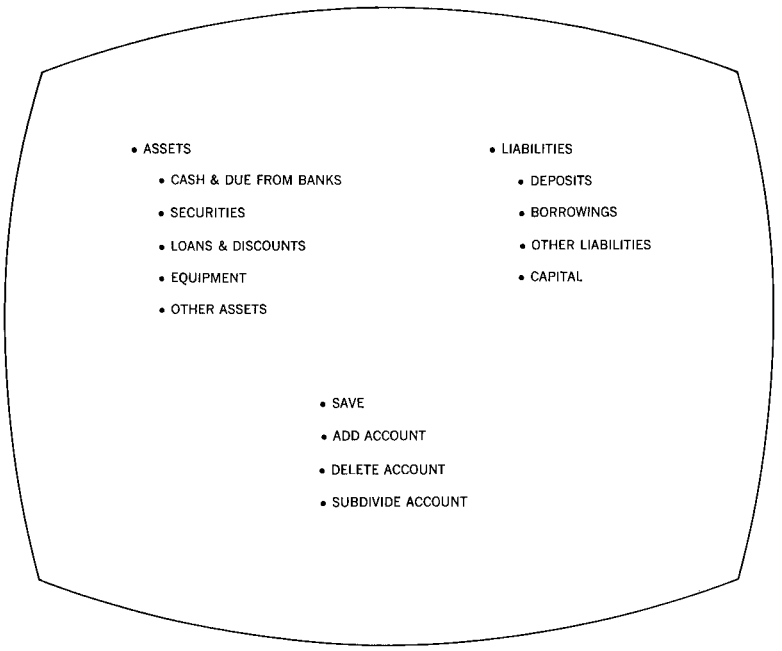
We have considered such a logical set of questions and have defined requirements for the necessary source code and decision tables. Such source code would represent both English language question-and-answer phrases and the interactive simulation language statements. Phrases or statements can be built from partial strings (down to the character level) or they can involve subroutines or whole programs. The decision tables, in combination with the responses of the user, determine the program output of our interactive application generator processor.

The first decision of the bank model user concerns the balance sheet accounts that are relevant to the goals of the model. If, for example, the purpose of the study were simply to learn how to properly characterize demand deposit activity, there would be no need to include an account for municipal bonds.

example

Consider the user who wished to construct a simple Deposits and Loans model. He enters the following statement: DEFINE ACCOUNTS. The system responds with a display such as shown in Figure 1. If the display device includes a light pen, the user may touch DELETE ACCOUNT and then SECURITIES, EQUIPMENT, OTHER ASSETS, BORROWINGS, OTHER LIABILITIES, and

Figure 1 Constructing a simple deposits and loan model



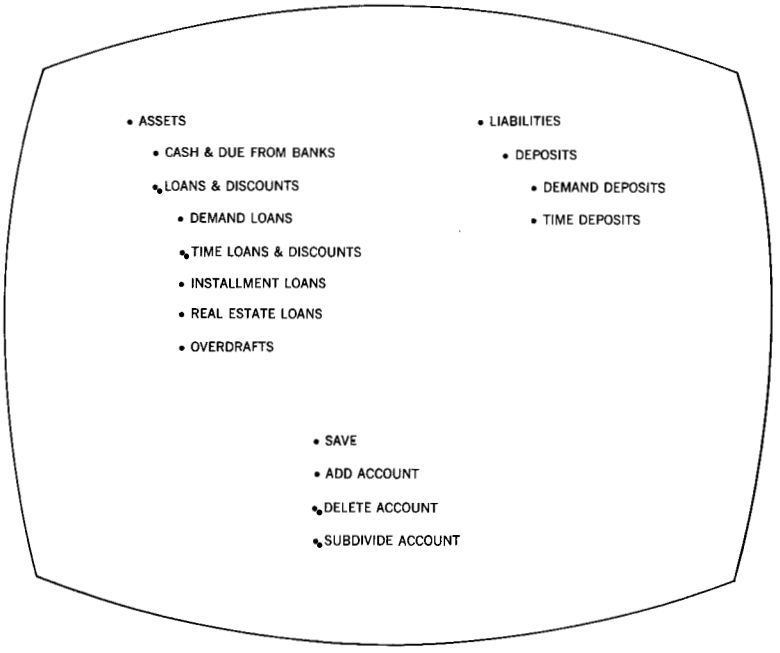
CAPITAL. He may touch SUBDIVIDE ACCOUNT followed by LOANS & DISCOUNTS, and DEPOSITS. Such actions would result in the display shown in Figure 2.

The user may continue in this manner until he is satisfied with the account structure. He may then proceed to describe the various banking activities relevant to this structure.

characteristics of interactive modeling

Some general comments are in order regarding simulation problems and the use of interactive, interpretive models versus modeling in a batch environment. It is true that a basic problem with many models is one of computer resources. Models often take excessive time to execute and frequently require a large amount of main storage. Reasons for these conditions may vary, and overcoming them often results in models that are not homogeneous in detail and level at which events are modeled. An example of this type of structuring difficulty might be a model design to study long-range asset management in a bank in which many events characterize every individual deposit and withdrawal, while at the same time attempting to simulate long-range investment policies. Although both types of events occur in a real bank, a simulation model that spans both types of events could require enormous amounts of computer time for the simulation of the individual deposits and produce negligible results. This type of problem is independent of the implementation language

Figure 2 Modifying the deposits and loan model



and can only be solved through consistent abstraction and aggregation of real-world events into a homogeneous simulation model.

A more difficult problem in many models is the dynamic core requirement that is inherent in many model executions. Characteristically, models contain many queues and time series that fluctuate in both size and content during execution. Compilation of the source language statements does little to alleviate dynamic core requirements, since about all that can be done during compilation is the generation of CALL statements to system subroutines that perform storage allocation at execution time by interpreting the calling sequence.

For certain types of model studies, any extra time taken by interpretive execution of statements (as compared to precompiled instructions) may be offset by the user time saved by interacting with the model during execution. Thus, by having information displayed during model execution, many fruitless runs may be interactively truncated at an early stage. It is also likely that the storage used for interpretive model statements (excluding the interpreting system) is less than the storage used for the equivalent compiled model statements. In either case, however, storage is much less of a problem with the currently available paging systems.

Concluding remarks

We have described our experimental approach to bank modeling, one intention of which is to permit bank management to estimate how well the policies they devise will stand up under the uncertainties that make the banking industry so dynamic. One need only look at government actions that affect the country's monetary policy to see the importance of possible but unexpected contingencies. We have modeled banking situations in a research environment. From a satisfactory demonstration in that context, we believe that our approach is worthy of further consideration by the banking industry.

CITED REFERENCES

1. K. Cohen and F. S. Hammer, "Linear programming and optimal bank asset management decisions," *Journal of Finance* **22,** No. 2 (May 1967).
2. J. F. Brown and B. A. Kalymon, *A Simulation-Programming Approach to Bank Asset Management*, Report No. G320-2645 (January 1971), may be obtained from the IBM Los Angeles Scientific Center, P.O. Box 64781, Los Angeles, California 90064.
3. K. B. Gray, "Managing the balance sheet: a mathematical approach to decision making," *Journal of Bank Research* **1,** No. 1 (Spring 1970).
4. A. S. Ginsberg, H. M. Markowitz, and P. M. Oldfather, *Programming by Questionnaire*, Report No. RM-4460, RAND Corporation, Santa Monica, California (1965).
5. D. W. Low, "Programming by questionaire: An effective way to use decision tables," *Communications of the ACM* **16,** No. 5 (May 1973).
6. M. M. Connors, C. Coray, C. J. Cuccaro, W. K. Green, D. W. Low, and H. M. Markowitz, "The distribution system simulator," *Management Science* **18,** No. 8 (April 1972).