PROTOTYPING: * AN APPROACH TO INFORMATION AND COMMUNICATION SYSTEM DESIGN

Mitchell G. Spiegel
International Computing Company**
4330 East-West Highway
Bethesda, MD. 20014

This paper describes prototyping, a state-of-the-art methodology to assist a design team in making a through definition and analysis of new requirements, feasibility, alternative selections, workload impact, system and/or application specification, implementation, and testing. Suggested prototype tools and techniques are presented, and guidance is included to aid a design team in obtaining accurate and timely results. This paper is not intended to be a complete text on design. It should be enhanced with a design team's expertise, consultation from sources with design experience, and reference to other design literature.

*Prototyping is a process (the act, study, or skill) of modeling an information-communication system architecture in one or more levels of detail, using descriptive models, abstract models, and working models of the system and its component parts (synonym: archetyping).

**This work was completed while the author was working with prior employers.

1. Introduction

A succession of events in communication information systems development has changed the modeling state-of-the-art. Consider the following system trends: users linked up to their data on-line; data files interconnect to form data bases; lower hardware costs; distribution of functions; and the development of software with engineering like discipline. One result of these events -- complex systems architectures that must be sufficiently resiliently to withstand continual changes in specification.

Although the procedure for strategic modeling has changed little over the last decade, the use of abstract design models has declined. The principal problem is the poor reputation of abstract design models caused by the practices of many systems designers. Many system designers seek a specification with a few binding performance commitments as possible. Architectural proposals dwell on the demonstrated performance of off-the-shelf hardware and software. However, the user ultimately obtains system performance of augmented off-the-shelf software.

The designer usually describes system performance to the user in connection with an abstract design model. The user concentrates on the structure of the model and on the results that purport to show that the system architecture as modeled will outperform the specifications, instead of stating simply that system performance will be reached or exceeded and demonstrated by the designer under realistic conditions.

With lower hardware costs, the role of design models is changing. Today's abstract design model explores the characteristics of the proposed system to identify candidates for working models. Issues that cannot be resolved be abstract design models is the elimination of grossly infeasible alternatives while steering the design team to an appropriate working model selection. The scope of modeling is broader, continuing across the architectural design cycle in a sequence of connected efforts wherein the results of one model provide the foundation for the next experiment.

Designers emphasize high-quality modeling, rather than the choice of modeling techniques or language. Hybrid design models comprised of multiple techniques predominate. Techniques and languages are selected more often for their familiarity, than for their absolute accuracy, or for their adaption to the allocate funds and allot time to complete the prototype project, guide and track project progress toward a successful satisfaction of the design objective, and understand the uncertainties associated with the use of design model results.

The objective of this paper is to guide a design team toward accurate, timely, and thorough information in design projects, by example, and through consistent application of a design methodology employing models; i.e., prototyping.

The reader must keep two basic principles in mind while reading and using the prototype methodology and guidance. One is that the methodology and guidance are descriptions of good practices for most

design projects. They do not cover, nor are they applicable in, all situations. The second is that the methodology and guidance stress reasonableness in all practices and procedures. The designer is responsible for determining the exact approach taken, constructing the models, and extrapolating the results to support the requirement. Any question of procedure or technique should be evaluated in this context.

The prototype methodology steps usually cannot be followed as a "recipe" with successful results. Instead, this paper represents good practices associated with areas of concern. In this sence, the paper is useful as a checklist and, to some degree, identifies areas where special competence, expertise, or particular attention may be required.

2. PROTOTYPE METHODOLOGY OVERVIEW

2.1. Modeling and the Information/ Communication System Life Cycle

Personnel involved in decisions affecting a computer system must be aware that design models can support decision making throughout an information or communication system's life cycle. Design models can be used to reduce certain design costs associated with conventional approaches, and to improve the ability to meet performance objectives. In each life cycle phase, design models can provide information required to make an effective and efficient decisions. Because of this, a design model constructed for a new requirement should satisfy the design objective (primary) but also provide a foundation for future models in the system's life cycle (secondary).

The project design team is urged to follow a "prototype" strategy rather than a traditional "linear" strategy to minimize conflict and encourage communication among groups. The "linear" strategy requires each successive activity to follow logically from its predecessor. When detailed analysis reveals problems, a loop back to an earlier phase is required. Decisions must be made at each phase. The tendency is for specification to be frozen without test at successively lower levels of detail. Changes in design are discouraged at all levels once such decisions are made.

The prototype strategy follows the same sequence. However, budget resources (typically 5-10% of systems cost) are allocated to produce an initial, highly simplified scaled-down prototype version of the system. The version is described, analyzed, designed, implemented, tested, and brought into operation using various types of models. Users, management, and designers review the prototype. Prototype cycles are repeated in greater detail until the actual is in full operation. Prototypes can be constructed from such tools and techniques as benchmark experiments, analytic and simulation models, and documentation charts or figures (descriptive models).

The following elements are essential in a prototype project:

1) A clear statement of a project's objective(s), including time and funds available

2) A methodology to characterize and analyze the workload, design experiments, identify alternatives, conduct experiments, analyze findings, and present results

3) Calibrated modeling tools and techniques which can provide information (in appropriate units of measure) needed to satisfy the design objectives

4) A team of knowledgeable application, communication, and ADP analysts

5) Assistance from individuals or groups with design expertise and available design literature

6) Management interest and participation.

2.2 Management's Role

For a design model to provide good decision-making information, participation by management is required. At a minimum, management must select the team members, provide guidance for defining the project, allocated funds and allots time to complete the prototype project, guide and track project process toward a successful satisfaction of the design objective, and understand the uncertainties associated with the use of design model results.

The team leader should be an ADP/ Communication/Software specialist (as opposed to a functional specialist), since it is their responsibility to design the most cost effective alternative that will meet the users requirements. Ideally, the team is composed of 1) functional analysts familiar with the user requirements to be automated or augmented, 2) specialists (in-house, contractors, or others) familiar with design methodologies, tools, and techniques, and 3) systems analysts familiar with both user requirements and the organization's objectives. Depending upon the project scope, the team size may range from two to several members. Intially, a small group (two to five people) should be identified to assist management in defining the objective and the effort required to complete the project. Based upon this group's recommendation, management can then assign additional personnel to the project.

The initial group of analysts selected for a project is tasked with defining its scope. Objectives, constraints, assumptions, and initial list of alternatives, information needed, and known sources of data are included in the project definition. During this task, management frequently interacts with the team members. Once the scope of the project is determined, it is management's responsibility to review the definition and ensure that it will satisfy their information needs within funding and time constraints. During this review, management should check that the definition does not constrain of limit the alternatives.

The results of a design model are estimates with some degree of uncertainty. Uncertainty can enter into the results from assumptions and constraints, tools and techniques, data collection, workload projections, subjectives interpretation, and team error. Most likely, all factors will affect the accuracy of the results. Because of this, users of the model's results must understand what these uncertainties are and what impact the uncertainties have upon the accuracy of results. Management must insist that the impact of these uncertainties be identified in the final documentation.

3. Prototype Methodology and Guidence

3.1 Introduction

This section is subdivided into the logical steps of the problem solving methodology, so that any prototype project will roughly parallel this section's flow. The intent of this methodology is to guide projects toward a thorough analysis and a timely and accurate assessment of alternatives. It can be supplemented by other information and individual expertise, and referance to an example incorporated into this paper.

Two very important principles apply to the use of this methodology. First, document assumptions, observations, procedures, and intermediate results as the project progresses. Good documentation significantly reduces the time required to report the results at a project's conclusion, and provides an excellent reference for the design team. Futhermore, the production of many portions of the system documentation might be reduced, and created only for working models. The working model may obviate the need for many elements of conventional system documentation.

Second, return to previous steps in the methodology and review all completed work if significant aspects of the design change. Where applicable, redo all work affected by changes. Assumptions, constraints, or objectives must sometimes be modified as additional information becomes available. When this occurs, identify the impact of these changes in the model. If time and funding permit, incorporate these changes into the model by redoing the appropriate steps. If this is not feasible, then, as a minimum, document the change, the parts of the model the changes affect, and, if possible, the relative impact the changes have on the results.

3.2 Define the Scope of the Project and Perform Macro-Analysis

It is essential that the scope of each project be clearly defined. This definition must convey the project's objective, constraints, and desired results. It is the design team's responsibility to identify the relevant assumptions and constraints. An initial list of data elements required to conduct the model should be developed. For each data element, the list should include the element's use (describe workload, describe hardware characteristic of an alternative, etc.) and potential sources for collection. It is important to identify the major data elements during the macro-analysis, since it must be determined 1) if the data is available and 2) if tools/techniques can be found to collect the data.

The three major categories of tools required for a design model are 1) data collection, 2) data analysis, and 3) modeling. The data collection tools and techniques assist the design team in defining the workload, human factors, and alternative hardware, software, and communication characteristics. They include hardware monitors, software monitors, accounting packages, documentation, interviews, and questionnaries. Data analysis tools allow the team to screen the raw data and establish relationships among variables. They include statistical techniques, display tools, and data reduction methods. The modeling tools provide the team with an aid for translating workload and requirements into ADPS alternatives. The three major types of modeling tools are descriptive models (charts or figures), abstract models (analytic, static,

and simulation), and working models (benchmarks and simulators of live programs, data, and procedures (synthetic and actual).

Prior to planning the details of the project, the design team must ensure that the project can be completed within the fund and time limitations. This determination is based on the results of the previous analysis (objective requirements, list of alternatives, data availability, assumptions and constraints, and selected tools/techniques) and the use of a skeleton model of the proposed approach. The results of this feasibility analysis must be reviewed by management before the design team proceeds. Concerns about time, fund, or data constraints should be voiced at this time.

3.3 Plan Detailed Modeling Approach

The information gathered to this point is used to prepare a schedule of modeling tasks and their sequence. At least the following elements must be included in the schedule:

1. Documentation from the previous tasks stating the project objectives, data requirements and availability, assumptions and constraints, time limitations, funding constraints, desired results, and selected tools and techniques.

2. A description of the experiments needed to achieve the desired results. The experimental design must detail the data sources, analysis procedures, type of desired results (data), and expected accuracy. This is an extremely important element that must be well developed in the modeling approach.

3. A description of each task and subtask to be performed. Each task or subtask description should inclued: a) required input for task and source of input; b) required output from task and destination (follow-on task) of output; c) procedures, tools, and techniques to be used; d) time schedule; and e) personnel assignments and responsibilities.

4. A set of evaluation criteria. This set of criteria should be based on the project objective. To the extent possible, the criteria should be quantifiable (cost, performance characteristics, variances, etc.). It may be necessary to weight (quantify or qualify) the criteria based upon their degree of importance. At the completion of this step, management should review the modeling approach for consistency with the project objective.

3.4 Obtain and/or Develop Tools and Techniques

The design team may need to obtain, develop, modify, enhance or define procedures for the use of the selected modeling tools and techniques before performing the experiments. Guidance for this phase can usually be obtained from documentation, vendors, developers, or users of the selected tools and techniques.

The team may also have to verify and calibrate the selected tools and techniques. Verification ensures that a developed method behaves as the team intends, while calibration tests and rectifies the accuracy of the method results with empirical data. A typical calibration includes generating sample data similar to that being collected for the experiments), conducting an experiment with that data, observing the method and the accuracy of obtained results, and modifying or changing the method if it will not meet the objective. Calibration requires the comparison of an existing system with the method's ability to predict the size and performance of that system, given a description (input data) of the existing workload's characteristics. The extent of calibration that can be performed, of course, is dependent on the availability of existing workload and system.

3.5 Define Workload and System To Be Modeled

The data elements used to describe the workload should be grouped according to a scheme (e.g., by functional area and processing category). Functional area groupings separate the workload by ADP user applications (personnel, civil engineering, operations, maintenance, word processing, etc.); these are the groupings used to predict future workload growth. Processing category groupings identify the workload's mode(s) of processing (e.g., on-line, batch, remote job entry, time-sharing, message-swiching); each of these groupings requires a different set of data elements to describe its workload. For example, within a functional grouping may be subgroupings of on-line inquiry and batch.

Within each grouping, two types of data elements are needed: load descriptors and workload characteristics. The number of data elements required for each of these is dependent upon the model's level of detail, desired results, and selected tools and techniques.

The selected groupings should facilitate prediction of future workloads and their characteristics. Predictions can be made by 1) identifying additional groupings and the expected data of their implementation or 2) identifying changes in loads of existing groups or 3) producing predictions of configurations required (sensitivity analysis).

Guidance for current workload collection includes: The number and kind of measurment periods; the stability of the workload, the environment the workload relates to, the defined level of detail, accuracy requirements, alternative sources, available time and funds, and the number of workload groups. The design team should understand the workload to be measured, so that a minimum number of

periods are needed to collect a workload description. It is the team's responsibility to select an adequate number of periods in order to meet the project objective, and then to carefully conduct the required data collection. For example, if a maximum on-line response time is specified, the measurement of on-line workload must be at a peak period. If, however, no critical performance constraints exist, measurements during or near an average level of activity should suffice.

Guidance for future workload data collection includes:

a. Future workload should be estimated in the same terms (load descriptors, characteristics) as the current workload (if one exists).

b. Groupings for the current workload should be carefully examined for similarities and differences with future workload estimates. When possible, future workload should be estimated in terms of current workload groupings.

c. Designers should take special care to identify the differences (if any) between current and future workload environments. Environmental changes to an application frequently impact the nature and frequency of system workload.

The information necessary to describe the application software characteristics is best obtained from functional analysts, programmers and program documentation. It is the design team's responsibility to obtain this information through interviews, questionnaires, or existing documentations, Some ADP and communications load characteristics can be extracted directly from the workload definition and user requirements. Transaction volumes, sizes, and responses should be identified; additionally, sources of remote activity should be stated for the communication model. Unique terminal requirements (hardcopy, optical-character read, hand-held data entry, etc.) must also be collected. Besides extracting this data from the workload definition, the team can also gather information from functional users (through interviews and questionnaires).

A detailed description of the system design must be developed for the experiments. This description may be a set of hardware, system software, and communication component characteristics, or an actual benchmark configuration. The exact method of describing each alternative depends upon the modeling tools being used in the study.

3.6 Perform Modeling and Analysis

The experiments consists of applying the developed tools and techniques (section 3.4), using the defined workload, software, hardware, and communication data (section 3.5). The number of experiments that are performed depends upon the time available, resources, desired results, and required degree of accuracy.

Prior to conducting the first experiment, the design team should first review the data from all previous steps. This review should last anywhere from two days to several weeks, depending upon the study's level of detail.

3.7 Interpret, Validate, and Report Results

After the experiments are completed, the design team must carefully review, interpret, and validate their results. The team must consider the project objectives, assumptions, and constraints when reviewing the results. Sensitivity of input data (workload, descriptions of alternatives), assumptions, and constraints should be carefully analyzed.

Documentation of the results and the techniques must receive considerable attention. It is the final documentation which will be used in future decisions and actions. The following guidance should be used in preparing this documentation:

1) Report the results in terms of the objective. Do not increase the document size by reporting additional results that do not support or relate to the objective.

2) Report the team's confidence in the results, and identify how uncertainties in the study might affect the results. Five areas of uncertainty which should be addressed are a) assumptions and constraints, b) collected data for study, c) tool/technique accuracy, d) workload projections, and e) subjective interpretation.

3) Structure the report to briefly cover all steps in the modeling process. Highlight those sections which specifically satisfy the project objective.

4. CASE STUDY

4.1 Development of an Information Service For AVCO Financial Services (AFS)

The AVCO case study demonstrates the many facets of on-line system performance and the protype methodology used to obtain that performance.

AFS's major business objective was to change the orientation of branch operations from accounting to financial consulting and marketing of various financial services by implementing an information system. Branch profit margins were tight under the manual operation. Growth could be achieved only by streamlining clerical operations and enhancing the ability of branches to handle

additional lines of business and volume increases without adding personnel. The information system was to provide better control over assets and revenues and improved customer service (accuracy and speed of consummating a loan and maintenance of up-to-date records).

4.1.1. Design Strategy

AFS management concluded that mid-1970's on-line system technology required a highly centralized system approach. Such future directions as distributed processing and data bases could be integrated into the operation when they became proven technologies. Automation of the most important business services took place first.

4.1.2. System Architecture

The automated system is called the Branch Operating System (BOS). Simply put, the BOS is a communications network composed of a terminal at each branch office connected to a central computer at AFS headquarters. At Headquarters, the system maintains up-to-date files from which reports are prepared for management's use in supervising branch operations. The system automates routine business transactions performed by the branch office.

The system supports over 1,000 on-line terminals and requires an IBM S/370 Model 3033 with 6 million bytes of memory. The daily transaction rate varies between 100,000 and 170,000 transactions per day. BOS consists of approximately 600 programs and 600,000 source statements. Over 90% of the programs are written in Cobol. The data base has over 2 billion bytes stored on IBM 3330's. A COMTEN 476 communications processor controls the network, and performs message switching.

To ensure orderly development, the system was planned in four design phases. The first design phase took ten months and cost approximately $1.5 million. It included the system design and specification of the software needed to operate the system. Additional phases were undertaken only after the first design phase was successfully completed. A feasibility study preceeded the four design phases.

4.2 Assembling the Design Team

4.2.1 Management Participation

The project was given maximum visibility by having the project director report to the Vice President of the Financial Services Division. A technical administrator reported directly to the project planning, technical design, monitoring, and system integration activeities. A postion of technical documentation requirements and implement, establish, and maintain the BOS document library.

4.2.2 User Participation

In general, the users participating in the project were required to 1) define their requirements clearly and in detail; 2) furnish information to project personnel on request; 3) modify and develop branch and home office staff and procedures as required to support BOS development and operation; 4) assist in the development of BOS manuals, forms, and documents; 5) review and physically sign-off on all system specifications at various points throughout the project's life; 6) notify BOS project personnel of all changes in law, regulations, policies, or Company plans that add, delete, or modify BOS requirements; 7) assist in developing branch conversion and installation plans; and 8) physically execute acceptance tests.

4.3 Consultant and Vendor Participation

Consultants and vendor technical personnel were employed as technical support personnel during the design phases since in-house skills in that area were in short supply. Management consultants interviewed user and management groups to obtain an unbiased statment of requirements. Another consultant developed models that produced predictions of quantitative standards for system performance. The hardware vendor's technical personnel provided system architectural alternatives, supplied results of benchmarks for similar finance industry requirements, and demonstrated performance models of selected alternatives according to specifications supplied by AFS. A software vendor either designed those parts of the system software which were either unavailable off-the-shelf or modified those parts of the software that were considered to be performance liabilities.

4.4 System Development Methodology

Because of the involvement of several outside hardware vendors, several consultants, a contract programming company, and the magnitude of the project, the procedures for controlling the project were formal and structured. The project was phased; work plans for all activities were detailed; performance and product quality standards were established prior to development; and organization components such as quality control, performance control and project control were created. During the course of the project, clear visibility of the current status was maintained at all times. Models were an integral methodology for all phases of the project.

Management followed the prototype approach because it enables better handling of frequent changes in user requirements or for design inadequacies. The trade-off involved was to accept a higher initial project cost to develop and cycle through prototype versions of the system rather

than attempt to define and freeze requirements and design early-on, thereby risking expensive loop-back periods to accommodate changes.

4.5 Feasibility Studies

4.5.1 Strategie Performance Issues

One of the first tasks of the design team was to define the performance attributes and measures in detail. Definitions of the performance attributes were esential to the plan for producing predictions according to the indicated measure. Performance predictions were subsequently used to make strategic decisions and establish performance standards for every successive phase of the project from concept of operation to full operating capability.

4.5.2 Requirements Specification

Business level processes, performance, and workload requirements were defined. A study of manual office operations was conducted for a full one-month period at 85 branch offices in 45 states. Generally, two offices (one large and one small office) were selected from each state. The concept of operation was discussed with branch office personnel, division representatives and Headquarters management. A functional process description of the BOS operation was developed.

Summary statistical data were obtained for branch office operations. The average number of transactions per account per month was calculated. An analysis of daily transaction activity indicated that the single significant peak for systems design consideration was the Monday morning condition occuring at the beginning of a month. Thus, the period studied included this condition.

Assumptions about the occurrence of transactions during the eight-hour period in each time zone were based on the typical business day. A profile of "peak Monday" traffic volume by hour was developed from these assumptions.

The project team employed the first of a series of prototypes to derive system performance requirements. Acceptable system delay times were developed by examing the operator and manager activity workloads for major transaction types at a prototype of the BOS terminal. Extreme and average elapsed arrival time distributions of major transaction types were also constructed from scenarios acted out by users at the prototype terminals.

Operators and users management specified the following major performance measures:
1) a 15-second delay for 90% of the transactions and a maximum delay of 60 seconds (based on the number of people in an office, the office workload, and goals for the amount of work to be accomplished during average and peak business periods); 2) system configuration was required to sustain a combined throughput of 19 BOS transactions per second and 0.4 administrative messages per second in a peak period; and 3) a central site(host and front-end) availability of 95% up time per month, network (high-speed line) availability of 99% and local net (low-speed line) and terminal availability of 95%.

4.6 Acquisition and Design

4.6.1 Vendor Evaluation and Selection

Specifications were furnished to interested vendors and three proposals were received. IBM proposed the PARS/Financial System. "PARS/Financial" is a derivative of the Airline Industry ACP package for financial institutions (Airline Control Program). ACP had proven itself as a large on-line communications system, capable of handling high-volume message and/or transactions throughput with rapid response. Two IBM Systems 360/Model 165's were required to support all AFS performance requirements. AFS management rejected the IBM proposal for several reasons -- OS software subsystems could not co-habitate with ACP; the 3705 communications controller could not be attached; terminals could be locked-out on multi-dropped lines; lack of special terminal features; and high configuration costs relative to the two other proposals.

IBM also proposed a second hardware/software configuration. The features of the second configuration-- OS/370, CICS, TSO, Programmable front-end 3705's, better line control under BTAM, new financial terminals-- and lower cost (by using two 370/155's in place of the 370/165's) made this proposal more attractive. AFS requested IBM to provide a demonstration of a prototype version of the system. AFS furnished a team of six people for one month to describe the functional, performance, and workload requirements to IBM personnel.

A prototype demonstration was constructed. A 360/50 using a forerunner of the teleprocessing network simulator (TPNS) acted as a driver system (RTE-Remote Terminal Emulation) to the system under test (370/155). Transmission of transactions over actual lines to and from simulated concentrators was initiated by exercizing the normal host BTAM polling function. Transactions were described by type, frequency of occurrence, data access requirements and message content. In the host, synthetic applications were developed by linking CICS macros to other programs that accessed files and executed an average number and mixture of instructions

equivalent to estimates of application activity. Seven files were formated to replicate a miniture version of the data base. Files contained 10% of the actual record volumes estimated for the real system.

Measures of work accomplished, system behavior, and utilization of resources were obtained from the prototype demonstration. Each unique message rate was measured for a period of six minutes following a six-minute period to allow the system to stabilize. Typical RTE measurement periods require 10-to-15 minute intervals for startup and statistics gathering. The interval length was reduced because the workload was homogeneous (only one-mode--transaction processing) and a small number of transaction types accounted for 95% of all activity (payments alone were nearly 70% of the total).

The results indicated that further alternatives be explored because the CPU resource was heavily loaded, primarily with network and data base I/O control activity. It was too early in the system life to have little or no resource safety factor. Furthermore, features desired for audit, control and error handling were limited with this approach. AVCO requested proposals for "intelligent" front-end computer capable of handling numerous functions and off-loading the host and a special purpose data base management system to reduce file I/O activity. An independent software house proposed a dual front-end that would require only a single host system. The proposal appeared to have desirable performance features similar to ACP (efficient host performance) coupled with the advantage of transaction inventory control and auditability of the system was about the same as that of IBM's configuration, while system cost and flexibility were improved. The principal disadvantage was the risk of designing and implementing special software in both front-end and host systems. The proposal was accepted.

4.6.2 Detailed Design

The second prototype cycle sought to minimize the risks inherent with the independent contractor's approach in Phase I, (the detailed user requirements and system design phase). A consultant was retained for the purpose of developing models of end-to-end performance. A peak-hour model was constructed using a packaged simulator. The data obtained from the requirements study were used to develop descriptions of applications process. Network characteristics were obtained from a discrete FORTRAN line simulator model of the communications network. A simple central server queuing model of the front-end processor was also developed. The results of the package simulator were calibrated to the experimental data collected with the RTE. A "performance budget" was created, allocating a maximum value for each component of end-to-end response time. The "performance budget" allowed trade-offs to be made between process, workload, and response time components. Analytical models for each on-line hour of the peak-day were developed. Results indicated that the software house proposal would meet performance requirements with a slim reserve of CPU cycles on an IBM System 370 Model 158 host. The front-end resources appeared sufficient to do the job.

Model data were provided to the design teams. Application programming received reports enumerating estimates of application core, CPU time, and working set size requirements and I/O operations performed. Data base designers were provided estimates of access patterns and access times for various transaction types, based on account number, key mapping schemes, load factors, file block size, empty space management strategies, and overheads for data and process recovery schemes. Operating system programmers were given estimates of CPU time and space requirements for special software. Throughput and turnaround time reports were studied by management. The off-line functions (unattended terminal) at each branch were examined to see if work could be completed before the next business day. Prototype models of various software and configuration options were used by most of the staff.

The third cycle of the prototype methodology began with the selection of equipment for installation. Management opted for an early prototype system to study design concepts in detail and permit BOS programs to be compiled and tested in a VS environment. An IBM System 370/Model 145 was acquired. The system was instrumented with hardware and software monitors. Measurements of software path lengths and I/O counts were made, and compared with model data. Models were updated and new forecasts made.

Another study conducted with the prototype 370/145 system examined the long-range feasibility of using the in-house system for time-sharing (program development)resources. The amount of special software required a sizeable number of maintenance programmers. Results of measuring TSO software and its interaction indicated the need for more processing power than has been scheduled to support in-house programming. Because a multiprocessor version of the IBM System 370/Model 158 was announced, it was decided to stay with the system on order and obtain a second processor if and when necessary.

Front-end computer measurement data indicated that the front-end system would not be able to achieve anything approaching its forecast response time budget. AFS obtained the services of an independent consultant to study the problems observed in the front-end software and hardware. The consultant concluded that the front-end would not satisfy BOS long-term requirements. Performance requirements and specifications previously established by prototyping and modeling were advertised in a request for proposal for off-the-shelf front-end system. After the bid solicitation, AFS decided to replace the front-end with a COMTEN 476 front-end system which would handle up to 1,500 terminals.

4.6.3 Test and Implementation

AFS assumed full responsibility for BOS testing and implementation. Testing requirements had been grossly underestimated by the application software vendor and a comprehensive test plan was non-existent. AFS was consequently required to develop a test plan and to re-test much of the system. At this point, AFS defined the following plan incorporating a fourth prototype cycle to ensure that the BOS system would be ready for mass conversion of the branches:

Regression Subsystem Testing - A retest of all BOS Subsystems.

System/Pilot Testing - The entire BOS System would be tested using a controlled test script of historical data taken from selected branch offices.

Shakedown Test - A series of tests using all facilities of the BOS System to prove that the system functions accurately according to state laws where AFS operates.

Parallel Pilot Test - Operation under BOS in selected branches in parallel with existing systems.

Live Pilot Test - Starting with two California branches and growing to six, BOS would process data in a completely live mode. Any network problems would be confined to one geographic area.

As AFS progressed into mass conversion, it was recognized that the on-line system would require more CPU resources than originally anticipated. A performance task force was established during mid-1976. Through their efforts, mass conversion was completed on schedule, without additional CPU upgrades. To emphasize the improvement obtained from this performance activity, measurement statistics are presented in Table 4.1.

The statistics below demonstrated the significant improvement in transaction throughout rate and a reduction in transaction path length. The data indicated the dramatic growth in path length from conventional CICS 1) to a highly modified CICS 2) with a special data recovery design (to meet audit and accuracy requirements), and the effect of moving to MVS ((2) and to a later version of MVS with improvement in path length(3).

4.7 Current Operational Status

At the present time, the Branch Operating System is maintaining 1.1 million loan and sales finance accounts receivable records. During the first month of operation after mass conversion was completed, the business was conducted in 956 branches with a terminal base of over 1,000. The peak day transaction volume during June 1977 was 115,000. The front-end communications system offered message switching capabilities to these branch offices with an average of 11,000 messages per day.

4.7.1 Feedback from the Operational System

Calendar year 1978 marked the attain-

Table 4.1. Host Measurement Statistics

	BENCHMARK TEST	TRANSACTIONS PER SECOND	NUMBER OF CPUs	CPU %	TRANSACTION PATH LENGTH
[1]	May 1972	8.0	(1)	65	52,000
[2]	Nov 1976	5.9	(2)	126	393,000
[3]	Jun 1977	9.1	(2)	137	111,200

ment of the terminal and transaction volume estimates made five years earlier. The system is meeting its performance objectives as originally specified. Management feels that the effort was worthwhile to consider performance a strategic issue and a general and continuing concern. The strategic design criteria were sound and their validity is improving with time.

In retrospect, some improvements could be made to the prototype strategy. The requirements study was an expensive, time-consuming effort. With such a homogeneous user population, a much smaller sample could have sufficed. Recent advances in sampling techniques serve to reduce the number of interview hours and concomitant costs of workload characterization.

At the time of the implementation phase, the vendor's RTE (TP Driver) was still experimental. Problems discovered only after exposure to an intial group of users could have been located by stress-testing the system with the RTE.

Of all the models employed, the least sucessful was the analytical model of the front-end computer. The front-end was modeled as an I/O bound system. Efforts were concentrated on accessing problems and queue delays in the I/O subsystem. The models failed to take into account such processing activities as polling, buffer management, traffic accounting, journaling, and serial use of resources. The path lengths to perform transaction activities and control I/O, created a processor boud front-end system with the same observable surging characteristics prevalent in the host RTE demonstration.

4.7.2 Conclusions and Recommendations

By acknowledging that system performance is a problem from the outset, and that most performance issues are settled during the strategic decision-making periods, a company or agency can remain in control of the situation. Capacity planning after the fact, without knowing what the strategic performance factors are, will not be as effective, and will likely increase the risk of loops back to earlier phases, or failure to meet performance objectives. Instead, management must obtain performance data from prototypes and models to make the best decisions under the risk about on-line systems. The source of good performance data for on-line systems are prototypes and predictive models, whose estimators can be used to manage a "performance budget". Measures of on-line systems on an ad hoc or post hoc basis reveals little of the problem. Resources must be allocated far in advance of on-going operations. Once in place, these resources and their users are not easily recast.

This case study has discussed only a few of the many problems which can occur in a design project. Many more have been experienced, and design teams should seek out and attempt to identify these problem areas prior to undertaking their own studies.

5. CASE STUDY BIBLIOGRAPHY

5.1 Prototype Example

1. Kirrene, M.J., Design of Operations-Oriented Decision Support Systems, Workshop At Ninth Annual SMIS Conference on MIS Productivity, Los Angeles, September 1977.

2. Bally, L., Brittan, J., and Wagner, K.H., A Prototype Approach to Information System Design and Development, Journal of Information and Management, Volume 1, 1977, pp. 21-26.

3. Giles,H., Successful Network Management Hinges on Control, Data Communications, August 1978, pp. 33-41.

4. Statements on Auditing Standards: Number 3 (SAS-3), American Institute of Certified Public Accountants, 1974, pp. 1-12.

5. Nielsen, N.R., Brandin, D.H., and Placko, M.A., Design and Simulation of the Man-Machine Interface Using Microprocessor Systems, SRI International, October 1977 (unpublished).

6. Kirrene, M. J., and Spiegel, M. G., Design for Performance, Proceedings of the Computer Performance Evaluation Users group, 15th Meeting, October 1979, NBS, Washington, D.C., pp. 129-140.

7. Remote Terminal Emulation Specifications for Federal ADP System Procurements, Automated Data and Telecommunications Service, General Services Administration, October 1978.

8. Use of Remote Terminal Emulation in Federal ADP System Procurements, Automated Data and Telecommunications Service, General Services Administration, March 1979.

9. Chambers, J. F., Findings on Hardware Monitor Measurements of OS/VS2 Release 1.6 Nucleus, SHARE Computer Measurement and Evaluation Newsletter, Number 28, November 1974, pp. 27-51.

10. Chambers, J. F., The FCB HASP Marco Can Reduce System Performance, SHARE Computer Measurement an Evaluation Newsletter, Number 30, March 1976, pp. 78-79.

11. Chambers, J. F., Communication Line Monitoring with a Hardware Monitor, SHARE Computer Measurement and Evaluation Newsleter, Number 35, January 1976, pp. 24-28.

12. McQuillan, J. M., and Falk, G., A Comprehensive Approach to Network Testing, Data Communications, August 1977, pp. 63-66.

13. Jenkins, C.W., Application Prototyping: A Case Study, Proceedings of Computer Performance Evaluation Users Group, 16th Meeting, October 1980, NBS, Washington, DC, pp. 311-315.