Accounting for Non-technical Issues in Computer System Design

by

DARRELL LEE GRAY

S. B., Massachusetts Institute of Technology
(1974)

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1980

Signature of Author _____

Sloan School of Management
August 1, 1980

Certified by _____

Stuart E. Madnick
Thesis Supervisor

Accepted by _____

Michael S. Scott Morton
Chairman, Department Committee

Accounting for Non-technical Issues

in Computer System Design

by

DARRELL LEE GRAY

Submitted to the Sloan School of Management
on August 8, 1980 in partial fulfillment of the
requirements for the degree of Master of Science
in Management

ABSTRACT

This thesis develops a basic methodology for evaluating the non-technical issues faced by a computer system designer or implementor. It models the political and social environment that the analyst and the system must perform in, to look for potential trouble spots. Some specific recommendations are then made to assist the analyst in avoiding problems once they are revealed.

A case study provides a basis for the analysis. It offers a systematic account by an insider of a major system development effort. The case deals with several examples of resistance encountered, from users of the system and from other groups within data processing who were involved in the project. The organizational structure of data processing was found to be a major factor in the intra-departmental problems.

Thesis Supervisor: Stuart E. Madnick

Title: Associate Professor of Management Science

ACKNOWLEDGEMENTS

I would like fist to thank my thesis advisor, Stuart Madnick, for his guidance. I owe a great deal to M. Lynne Markus, my thesis reader, whose ideas and suggestions made it possible for me to get here. She helped give my vague ideas this form, and freely gave assistance and encouragement along the way.

Without the great people at Worldwide Insurance, of course, this document simply wouldn't exist. They put up not only with my mistakes, but with my writing about theirs as well. I think they will recognize themselves and accept my gratitude. And lastly, I owe special thanks to a couple friends for keeping me going through the last few months.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Many valuable insights into resolving the problems encountered in implementing computerized information systems (CIS) have come from the application of work in other fields to the problems. Early work with computers was necessarily very much technically oriented. The machines were very expensive and difficult to program. Therefore, they were only used when the value of the computer was obvious (usually because of the volume of calculations to be done). At first, the projects were also very limited in scope, because of the scale of what the computer was capable of doing.

As the machines got bigger and faster, and the programmers and support software got better, things changed. The capability grew for developing systems that no longer served a single simple function. The easy to program tasks with obvious benefits to automation had been done already. Programmers began to develop programs which took on more of the task they were helping with. The benefits were not always quite so clear as before, and the effects on task and the people involved grew. The programs began to be used to produce the reports instead of just the

contents of the reports, and now the program was affecting not just the original user, but all of the end users of those reports. Instead of just being used to process data, the computers were now being used to store data and transfer it between people. First through printed reports, then through data files that were processed again by another user's program, the computer became an information forwarder. In many cases, this meant that the human interactions were changing or being cut out entirely.

Here is where programmers began to get into trouble. Programmers were used to finding ways to make machines run better. They had no experience in dealing with human problems that their programs caused. It became difficult to separate people, processes, and programs. Before, the computer had been a tool used by a single person as a part of a system for getting things done. Now, the people were being rearranged to take advantage of the computer's new capabilities. The system for getting things done was now being changed to take advantage of the tool's abilities for improving on information flow in the organization. And the programmers suddenly needed more 'people' skills.

The data processing organizations realized changes were happening, and Systems Analysts joined programmers in those departments. They were to develop the new systems for getting the firm's work done, and the programmers would

again only have to worry about the technical problems. The analysts would develop the package of computer programs, forms, etc. that constituted the new system and aid in replacing the old system with the new. As the analysts were fitted into the organization they became centers of change in the firm. Their job was to develop systems that took advantage of the computer as a storage mechanism, a means of communication, and as a processor, to improve the profitability of the firm. But still, no one was really concerned about the effect on the people in the organization.

It was the late 1960's when a real look was taken at why computer systems failed, that the people issues really began to come out. Work from other fields began to be adopted and brought into use in the field, in an attempt to make system's theory 'people' sound as well as technically sound. The earlier work of psychologists, sociologists, and organizational specialists was applied to recurring problems with CIS. Motivation, perception, resistance, reward, Theory Y and similar words and phrases began to appear in the literature.

Through the 1970's research in these areas has continued to explain problems; the literature has grown much richer. But, to date most of the work has either been explanatory or simply descriptive. A great deal of effort

has been put into identifying problem areas to be studied, into trying to measure the extent of the problems, and into understanding the problems and why they occur. There seems to be a shortage of work that makes suggestions to the analyst about what to do to avoid such problems. My goal, then, is to try to develop a methodology for the analyst to use in accounting for these non-technical issues in the design phase, rather than in a post mortem of the project. To do this, I have drawn from recent work in political science on evaluating investments in light of political and social risk.

Approach to the Problem

In my ongoing work with Worldwide Insurance* , I had begun to run into many instances of resistance to the CIS that I was working on, the Cost Analysis System (CAS). I was fortunate to be working for a person who was well read in the current literature, and was willing to try out current ideas in CIS implementation. We had attempted to carefully follow the best advice we could find for introducing current technology into the Cost Analysis System. Many possible forms of user resistance had been thought about in advance, and we felt that we had covered ourselves pretty well. Yet we were still encountering problems, very few of them technical.

---

* fictitious names have been used throughout the case

I hoped therefore to use my work at Worldwide as the basis for a look at how one might predict and hopefully prevent "people" problems in CIS implementation. I was in a position unusual for researchers in this field, in that I was already established as an "insider" in the firm. I was a real part of the development project that I wanted to study.

There are several advantages to being an insider in this situation. I was there legitimately, and had established myself before becoming a researcher, so I didn't need to justify my presence. I already had access to people and records I needed. The subjects need never know that I was anything but a systems analyst. I had already collected most of the facts that I would need -- my files already held most of the existing paperwork documenting CAS. Records were not very good, but I had most of the old memos, schedules, and reports that remained, given to me as the person currently responsible for CAS. Much of my interviewing could be done informally as a part of the regular meetings I was having with the participants. And my note-taking in the meetings was already legitimized as well, allowing me to record points for my thesis as well as for the project.

The data collection went pretty much as planned. I actually informed people in December that CAS was the

subject of my thesis, but not what the exact nature of the thesis was. It was assumed that the topic was technical, and I did not bother to clarify. I formally interviewed several people, but I was able to discuss the system and its history informally with each of them informally as well, as a normal part of my job. There was very little in the way of written documents available that I didn't already have a copy of, but I was again able to find that out in the line of duty. With the exception of some timing questions, the written evidence and memories corroborated well. I was fortunate enough to find several fairly complete system status reports among the rest, which helped put things into a consistent time frame. Thus, there was never any question in my mind that what I had was as accurate as people remembered, and reasonably complete.

Much of the early history of CAS came from those documents, and other assorted memos, etc. that had made their way to my files. The whole was rounded out by interviews with Rob West and Don Massey, especially, who had both been on the task force. It is interesting to note that no one from the original six members of the task force was still at Worldwide four years later, and only Rob and Don were left of those who had served on it at any time.

It was easier to find people who had been associated with the system since the task force dissolved. I was able

to talk with several programmers, analysts, and accountants who had worked with or on CAS at some point. I was also able to reach a couple of the original programmers from the task force period for their comments as well. Whenever possible, I took the stance of simply being interested in the past of the system that I was now involved with. By asking technical questions about the early system, I was even able to legitimately take notes at these times without making anyone uncomfortable.

My thesis thus provides what is probably a more accurate account than an outsider would be able to produce. I have had the advantage of first-hand knowledge of the thinking behind many of our decisions. Thus, this thesis serves the purpose of providing a systematic account of the resistance issues involved in a CIS implementation, both resistance from users and from the rest of the data processing group. In addition, I have tried to provide a framework for previewing the non-technical issues that might become a problem in CIS design and implementation. And finally, I have attempted to make concrete recommendations for avoidance of some of those problems.

Plan of this Thesis

In Chapter 1, I have presented the reasoning behind this project and some of the conclusions developed.

Chapter 2 will provide the background necessary to understand the case, in terms of the major actors and the early history of the first CAS system. To make some sense out of some of the problems that came up during the project, a fairly complete outline of the earlier interactions is important. Chapter 3 takes up the history of CAS from the time I began work for the company, through the decision to rewrite the system. The story continues with the development of the new system in Chapter 4. I take a more detailed look at the problems encountered with the project in Chapter 5, and complete the analysis and conclusions in Chapter 6.

A summary of acronyms appears at the end of the Appendix, as a reference in case the abbreviations become overwhelming.

CHAPTER 2

BACKGROUND OF THE CASE

This chapter will provide the necessary background to understand some of the problems encountered in the development of CAS. It begins with an overview of the firm and a look at the major actors. The early history of the first CAS is then outlined to establish the setting for Chapters Three and Four.

The Firm

Worldwide Insurance is the American affiliate of a London based firm, with its U.S. headquarters located in Boston. It collected nearly one billion dollars in premiums in the U.S. in 1979, primarily in commercial policies. The firm has grown rapidly in the last few years, resulting in a lot of pressure on computer services and other support areas, to cope with the rising workload and to expand the services available. The corporate management has made many major changes, including corporate reorganizations, in the past few years, trying to manage the rapid growth. Unfortunately, a number of these changes with far reaching consequences (especially for data processing) have been made with very little notice to those

who might be affected. This has resulted in a number of problems, such as monthly reports being published 6 months and more late, and in a lot of bad feelings toward management.

The MIS Division

The Management Information Services Division (MIS) has overall responsibility for all data processing activities at Worldwide. MIS has its own senior vice president who reports directly to the executive committee. Within MIS there are four departments, plus an administrative group.

The Applications and Programming department (APD) has responsibility for all application system development and maintenance for non-MIS departments. Within the APD department the current organization is along functional lines, parallelling the structure of the rest of the organization. Each programming team is often referred to by the name of the major system or subsystem it is responsible for. Thus, the group that I am a member of, within the APD financial reporting section, is usually called Cost Analysis or simply C-A-S. This corresponds to the same area of the accounting function of Worldwide. Within MIS the financial reporting group consists of about 50 people headed by a junior vice president, with responsibilities for general ledger, accounts payable, various

WORLDWIDE INSURANCE
Corporate Organization Chart
(Partial)

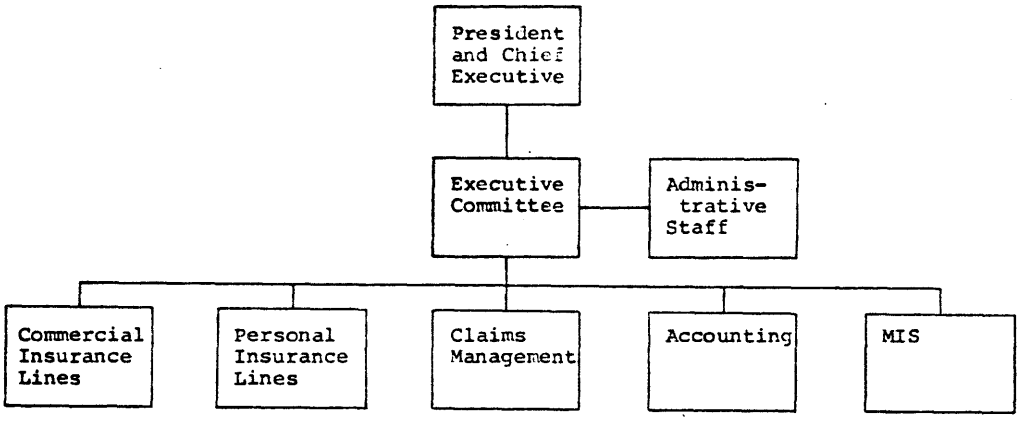

Figure 1

WORLDWIDE INSURANCE
MIS Organization Chart
(Partial)

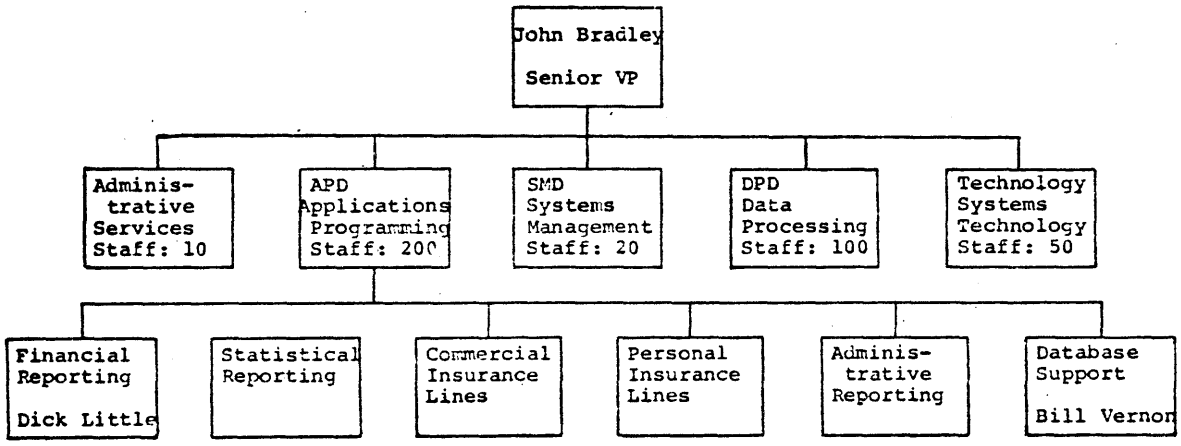

Figure 2

federal and state required financial reports, insurance profitability, premiums, claims, and outstanding risk systems, some workload measurement systems, and the Cost Analysis System.

The Data Processing department (DPD) provides for all day to day computer operations -- data entry, scheduling, preparation, and quality control for running existing programs.

The Systems Technology Department (Technology) is responsible for all hardware, operating systems and other system software. They are also charged with providing technical expertise to APD when required.

The Systems Management Department (SMD) has a role as a technical quality control monitor. They assist in interfacing with user departments, develop the functional specifications for most major MIS projects, and provide cost--benefit analyses and post-completion audits. In addition they monitor ongoing application program usage and performance. Sue, our representative from SMD, started at Worldwide at about the same time that I did. She immediately became involved with CAS as one of her first assigned systems, trying to manage and interpret the stack of old (and new) CAS project requests. Much of her time since then has been spent with CAS, despite her other responsibilities, because of the scope of the modifications that

WORLDWIDE INSURANCE
Corporate Accounting
Financial Analysis and Reporting Section

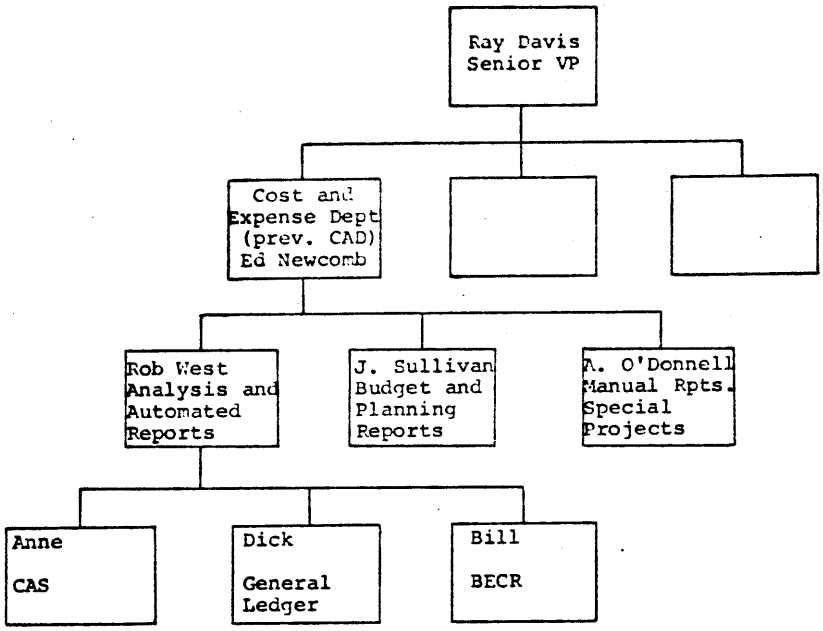

Figure 3

WORLDWIDE INSURANCE
MIS - APD
Financial Reporting Section

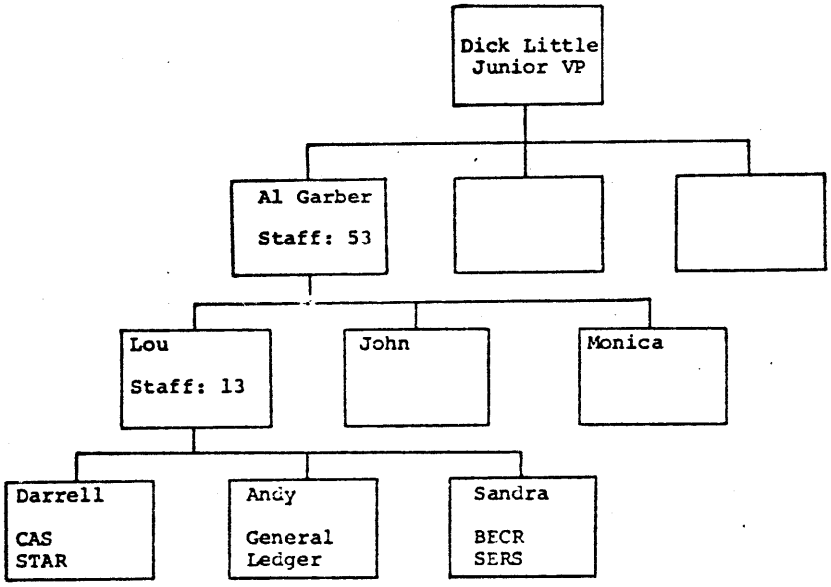

Figure 4

we decided to make to the system.

MIS at present includes over 300 staff members and has about 50 additional people as contract programmers at any time. Despite investments in advanced hardware and software products and the availability of a good training/educational program, the MIS group suffers from very high employee turnover (the average new programmer will not remain a full year at this point). Part of the problem now is that Worldwide has gained a bad reputation in the job market as a high turnover firm. This tends to perpetuate any earlier personnel troubles, and makes it hard to attract needed people. As a result, Worldwide is forced to maintain a large number of contract programmers to fill slots that should be filled by permanent staff.

Most of the MIS staff are issued copies of a six volume loose-leaf manual which describes policies and procedures of the division. This provides guidelines of general MIS policy, as well as attempting to routinize some of the normal operations. For example, standard procedures for submitting tests to be run are layed out in detail in it.

Mr. Bradley, the vice president who runs MIS, like several of the other corporate officials, appears fond of regular change in his organization. Like corporate management, he tends to mandate change and only later discover

what the effects were. A recent example is his decision in August 1979 to disband the SMD department. The immediate response of several users was to create a similar functional unit outside of MIS. One such group was the accounting area's financial reporting group who took two SMD people (including Sue) and an APD manager and formed a new section themselves to replace the SMD functions. Although no official retraction was made, two months later some of the remaining SMD people were casually informed that the group would not be dissolving in the near future.

A less drastic but typical change is the rearrangement of senior managers under Bradley. In the less than two years that I have been at Worldwide my boss, Lou, has reported to five different people. Lou's function has really only changed once, and that did not correspond to any of the 5 changes above his level.

MIS Project Requests

The MIS Division is run as a service organization. Their services must be requested formally in writing in most cases. There is no billing mechanism for any of these services, so these requests are really the only way of monitoring and controlling the volume and disbursement of services. The suggestion for a project may come from within MIS, but the request itself must be made by a user

of the system. For some services, such as a request for DPD to run an existing program, this may take the form of a memo. For the development or modification of a program or CIS, the request is made by completing an MIS Project Request Form.

Requests for programming go to a coordinator in MIS, who forwards copies to the manager of the appropriate APD programming team, and to the SMD representative assigned to that user. The coordinator also assigns a project number and records the project in the Project Control System, which is used to monitor the status of all projects.

The APD manager and the SMD representative then consult to determine whether the project is significant enough to require formal specifications (the informal rule of thumb in use is that any project with more than 2-4 weeks of programmer time gets formal specs). If the Project Request Form is not completely clear, then SMD may either choose to return the request to the user for further clarification, or to consult with the user and APD to develop formal specifications. Otherwise, if the request is clear and complete, APD may choose to accept or deny the project as defined in the request.

Whenever the intent of a requested becomes clear enough, the APD manager has a right, after consulting SMD, to reject the request if it seems unreasonable. This may

be for reason of excessive resources required, or for the inappropriate nature of a project, such as a system that benefits the requestor rather than the company, a duplicate of an existing system, or a simply impossible idea. Otherwise, APD may accept the project for immediate work, or accept it and defer it for available staff time or other resources. (In practice, as far as I could determine no project has ever been formally rejected. Instead, they are accepted and deferred permanently).

In any case, the user is entitled to a written response to a request in a "reasonable time". (Again, the manual fails to specify what reasonable is; normally it seems to be about one month).

If accepted, the project is then scheduled and programmed by the responsible APD group. Results of tests are supplied to the SMD representative, who in turn delivers them to the users. When all stated requirements (on the formal specs or the original request) of the project have been satisfied, the user must "sign off" on the system. APD then completes documentation and submits the system to DPD to be put into "production". Once DPD accepts it for production, it may be run at any time the user makes a written request.

The user has no option to change requirements or to cancel the project once programming commences. Until that

time they may withdraw any request. They are charged neither for development nor for computer usage, so their incentive for making reasonable requests depends largely on the inability of APD to handle many simultaneous major requests.

The project's status is reported throughout the cycle through the Project Control System. Various reports are issued on a weekly or monthly basis, some going to users and some to MIS, and to all levels of management.

The Cost Analysis Department

The Cost Analysis Department (CAD) was established within the accounting function in the early 1970's to evaluate the worth of various portions of the company's business. They were to look into the profitability of each office doing business for the company, and the profitability of each line of insurance business. They would use cost chargeback analysis, which invoved finding appropriate ways to allocate each of the indirect costs of doing business to offices and lines of insurance. Then, profitability statements would be developed for each office and line.

The goal of the department has remained consistent through several staffing changes, changes both in number and in actual personnel. The staff has been made up mostly

of cost accountants and cost accountant trainees. In August 1979, this department was reorganized along with most of the accounting area. Its function was expanded to include additional responsibility for a wider range of corporate financial reporting. The staff was increased back to about 15 and the name was changed to the Cost and Expense Department to recognize its larger role in the organization.

Throughout the life of CAS, this department has been the primary user of the system. The system was designed essentially to automate and expand the profitability reports that CAD was already producing. Rob West, a manager in the department, has been a primary user and contact with MIS since his arrival in early 1976.

The Early CAS

The Cost Analysis System had its beginnings in January 1975, with a directive issued by the president of the company. That created a task force with a charter to review existing (manual) methods of allocating expenses, and where necessary, to develop new procedures for charging out those expenses. The goal was a fair and equitable allocation of the full costs of doing business to each office and to each line of insurance. The expectation was that use of the computer would aid in producing such

reports in a regular and timely manner.

The project team was formed from the top three people in the Cost Analysis Department and three people from MIS. They quickly decided that the procedures developed would indeed have to be automated to be used effectively, so the MIS staff continued with the project. From February to September 1975, the team gathered information, reviewed local and regional office operating procedures, and then began to develop allocation methods and report layouts.

In October a software house was brought in to handle the bulk of the programming. They worked with the MIS members of the team to develop final system specifications. In November 1975, programming was started using an initial design that incorporated all of the specifications that had been completed at that point.

The task force was under pressure to complete the project, and since the software firm had limited staff to devote to the project, it was decided to split the project into two pieces. The first piece would be the part that would extract all necessary information from other systems and format it for use by the second part of the system. The second section would do all of the analysis and reporting to accomplish the project's objectives.

The first section was brought back in house, and several contract programmers were hired to do the work on

it. The second section, which allocated the expenses according to the procedures developed by the task force, remained with the software house.

During the next two years, the system was developed and implemented in two steps. The first step was completed in June 1976 and offered all the features that had been requested prior to November 1975.

The second step was to resolve any problems found in the step one design, and in addition included many enhancements that were developed by the task force during the programming of step one. At the end of December 1977, the system was considered to be more or less complete. The project team had gradually dissolved towards the end of 1976 as the members began to meet less and less often. The primary users, the Cost Analysis Department (CAD), had begun to request further changes through the official request process by that point, rather than by working through the project team.

Between February 1975 and June 1976, management of both MIS and CAD had changed. The MIS members of the task force had changed several times; none of the original members remained by mid-1976. The new CAD manager replaced the old in the team. According to an August 1976 project status report, this turnover had not only resulted in a lot of new ideas and points of view, but in a nearly complete

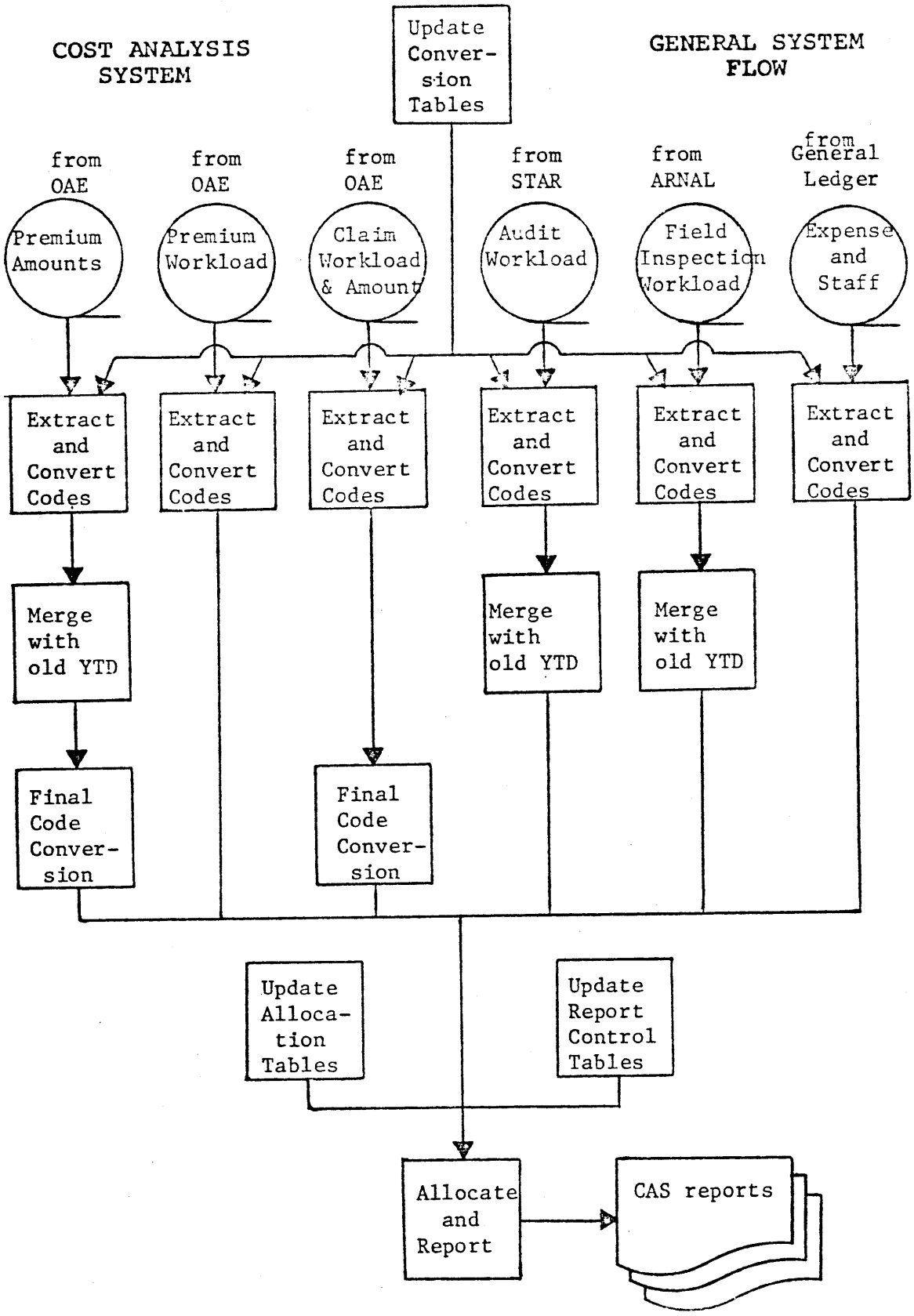
COST ANALYSIS SYSTEM
GENERAL SYSTEM FLOW
Update Conversion Tables
from OAE
Premium Amounts
from OAE
Premium Workload
from OAE
Claim Workload & Amount
from STAR
Audit Workload
from ARNAL
Field Inspection Workload
from General Ledger
Expense and Staff
Extract and Convert Codes
Extract and Convert Codes
Extract and Convert Codes
Extract and Convert Codes
Extract and Convert Codes
Extract and Convert Codes
Merge with old YTD
Merge with old YTD
Merge with old YTD
Final Code Conversion
Final Code Conversion
Update Allocation Tables
Update Report Control Tables
Allocate and Report
CAS reports

Figure 5

redefinition of the original plan. Meanwhile, there was continuous pressure from top management for immediate results.

Because of the pressure, there was no chance to stop and review the program design and make any changes. As decisions were made about what to add next, the task force produced specifications and passed them on to MIS and the software house for programming. Each enhancement was fitted into the system as well as possible, but the basic design was the one developed from the November 1975 specifications. Therefore, it was not easily changed to accomodate some of the early oversights. At the end of 1977 all of the task force specifications had been implemented and the known problems resolved.

In January 1978, preparations were being made to put the CAS into 'production', that is, completing documentation, etc., so that the responsibility for regular running of the system could be given to DPD. A list of changes requested by the user dated back to 1976, but the original request as proposed to MIS had finally been fulfilled. So, the system had been "signed off" by the users, to meet MIS project request rules, and to make requests for changes to CAS legitimate.

Even before it was accepted, the "live" tests showed that the system was an operational nightmare, requiring

massive amounts of machine time, tapes, and disk space for its monthly runs. The allocation processing required a final monthly run of 15-20 hours of continuous time in the computer with no possibility of restarting in the middle in case of problems. At the same time, DPD was averaging about 10 hours between machine failures, making a successful CAS run a case of extreme good luck. The reports produced were completed 2-3 months after the close of the month that they were reporting. And there were still major reporting problems that made it impossible to determine from where and how expenses were allocated to produce the final numbers. According to Rob West:

> There was no way to audit the results that came out of the system. The Policy Transfer reports didn't match the Allocated Cost reports or the extracts. The programmers called it a rounding problem and said it couldn't be helped. We were using a dozen allocation passes at the time, so the results were next to impossible to recreate by hand, not that we didn't try. Even our extract reports had to be manually totalled to get the base figures for checking percentages.

As a result, the reports were not accepted, trusted, or used by the offices that eventually received them.

At the end of 1977 Don, the lead analyst working on CAS transferred to SMD. This was a major step up for him, which he credited to the success they had had with CAS. Although he kept some responsibility for the system initially, he was "quite fed up with CAS and all the people involved with it". During January, responsibility for CAS

had been shifted to a new manager in MIS (my eventual boss, Lou). At that point, Tom was the only person left working on the system. He had worked on CAS for a little over a year by then. Lou had assigned a second programmer, Beth, from his group to work on CAS as well. But Tom, the last person with any real knowledge of the programs in the system, left for another company in March of 1978. This was just before the February processing run. By the time he left, Beth was somewhat familiar with how the system worked, but lacked any real knowledge of the internal workings of the programs or of the allocation procedures being used. She was further hampered because the major processing was done in a single huge PL/1 program and she was a trainee and only knew COBOL.

When the February processing was attempted in late March/early April 1978, the PL/1 program failed, grinding to a halt part way through the several hours it was supposed to run, and printing in explanation only a brief message about an attempted illegal division. Beth made several attempts to fix the program using available manuals, but she was unable to solve the problem. No one else in MIS with enough knowledge of PL/1 to help was willing to try to fix the program. At that point a decision was made to try to hire a PL/1 programmer to help, and in June 1978, I joined the CAS group.

CAS HISTORY IN BRIEF

1975	January	. Directive creates Task Force
		. Task Force begins Analysis of needs
	November	. Programming of CAS begins (stage 1)
		. Task Force continues analysis
1976	June	. First round of programming completed
		. Stage 2 programming begins
1977	December	. CAS "Signed off" by CAD
1978	March	. Tom leaves, CAS stops working
	June	. Darrell hired
		. Corporate Reorganization announced
	August	. CAS running, Improvements underway
1979	January	. First suggestion of rewrite to CAD
		. Regular meetings with CAD begin
	May	. CAS rewrite proposed again
	July	. Rewrite proposal to CAD management
	August	. Feasibility study requested
		. CAD reorganized
	October	. Feasibility Study results presented
	November	. Rewrite requested, terminals ordered
1980	March	. User terminals installed
	April	. Interface problems resolved
	June	. Scheduled system delivery date

Figure 6

CHAPTER 3

THE HISTORY OF CAS

When I arrived at Worldwide, CAS had officially been accepted by the primary user, CAD. This does not mean they were happy with it, only that under the rules of the MIS project request system they had to "sign off" the project once the request, as stated, had been fulfilled. And CAS met its original specifications, however badly it did so.

But, at that point, the February report was 4 months late and CAS wasn't working. The system still hadn't been accepted by DPD as a production system because of the continuing operational problems and the lack of documentation. Some programs had been almost continuously in a state of flux since they were written, sometimes with more than one change being tested at a time. As a result, DPD had never even been requested to accept those for production. Reports had been issued from the system to the eventual users several times, but there remained so many problems and questions about the validity of the results, that the reports probably wouldn't have been used if they had been timely. There was a general consensus among both MIS and CAD that although the basic idea of the system was good, the output, such as it was, had never been put to any

real use.

The extract programs that made data available for processing, which made up the front end of the system, worked. They consistently ran to their normal completions, and produced reasonable results. The programs had been changed by a number of programmers over the two and a half years of their existence. Some had already been changed by more than a dozen people, most of those being contract programmers. There was essentially no documentation for the programs and no consistency in either methods or naming conventions. Internally, the programs operated in a style which was far from accepted accounting principles. Any data that failed to meet the "rules" of the program were simply rejected, with no warning message, explanation, or even totals of rejected data. Bad data, untranslatable codes, data that failed edit or selection criteria, were all simply ignored.

The reports, as a result, were typically a few per cent short of the totals required to match the feeder system. But, since there were no obvious biases in the results, and they were only used to develop percentages for cost allocation in CAS, those errors were accepted (that is, the users considered the problem much less serious than many others with CAS). The only saving point about that part of the system was that the programs were fairly simple

to start and somehow they still produced acceptable (although incorrect) output.

The rest of the system, however, which did the actual cost allocations, consisted of a single huge PL/1 program. I think the program may have violated every modern programming rule. Trying to work with that program would convince anyone of the value of modular program design and structured programming techniques. It is simply not possible to read and understand several hundred pages of code at once, let alone trying to follow the complicated logic of this program. At the same time, it used none of the ideas that make a programmers life easier. There were no blank lines, and the margins were all different, yet indicated no structure. Variable names had almost no meaning, one or two letter names were used where ever possible, and were used again for different purposes elsewhere. On one page 'XX' might be a string of characters containing the name of a line of insurance, and a few pages away 'XX' will reappear as a counter or a premium dollar amount. There was no documentation, either internal or external, and in addition the program used a number of advanced features of PL/1 that made it even more difficult to understand (such as the extensive use of pointer variables).

There was a suspicion that the last programmer had sabotaged the program when he left, but the program was so

complicated that chances are that if he had had the urge, he wouldn't have bothered. When it worked, the program required 12-15 hours of continuous computer time and tied up several disk units and several tape drives. At the moment it wasn't working at all; something in the current data was causing it to reach a point where it simply quit processing. It had been designed under several assumptions that had changed even before the first code was completed.

When they started the design process, the organization was simple and compact, as was the reporting structure for lines of insurance. The table structure they chose to build the system around was reasonably efficient under those circumstances. Under the new conditions the old design was not practical, the table structure became very inefficient with a larger, more complex organization to be represented. Also, some of the simpler processes were originally combined in the PL/1 allocation program for processing efficiency. But, as those processes were redefined, they often became more complicated, and the logic to allow them to remain combined became more complex as well, until eventually, they should have been separated again for efficiency. But the software house had neither the time, the inclination, nor the incentive to rewrite the allocation program during the middle of the project. The extract portion of the system had fared much better. The programs were so simple to begin with that they had simply grown

messy with changes.

During the remaining week of June and through July 1978, most of my time was devoted to trying to unravel the hundreds of pages of PL/1 in "the" allocation program. I needed to find out why it wasn't working, and to try any possible solutions that might solve the problem temporarily until I could figure it out. I was introduced to the PL/1 "experts" of the firm, but it quickly became apparant to me and my boss that it was not worth the effort of asking them questions. They had no particular interest in helping to solve the problem. In any case, it began to appear that it was not a technical problem with PL/1 as it first appeared, but a problem in the complex processing that was going on internally.

Originally, as it failed, the system had given a message about an illegal division computation. We traced this to the specific numbers however, and found that they and the result should be legal. It therefore looked like PL/1 was failing for some reason. But then I discovered that the variable containing the result was redefined at the time of failure from what we thought it was. PL/1 allows you to specify the number of digits to be reserved for a number, and the result here had one digit too few reserved.

I began a systematic brute force attack on the problem, changing anything that might lead to the problem,

and then changing any new problems that those first changes had caused. For the most part, this involved increasing the number of digits reserved for variables. Increasing the number allowed for the result of the division had caused another problem later on, when the result was used in a computation that in turn became too large. I couldn't simply increase all the fields, or our already large requirements for computer memory and disk space would get much worse (one digit would cost roughly 10% more storage space).

So I proceeded one step at a time. After repeating the cycle several times, I was finally rewarded with a test that worked. Eventually, the method had paid off, and at the end of July, although I didn't really understand what had been wrong, the program was working and CAS was back in operation. The problem had been that the numbers had grown too large for the program to handle, but it wasn't clear why. It could have stemmed from the data or from the control tables set up by CAD, but it hadn't appeared until far into the processing. It wasn't until two weeks later that I tracked it down to an unusually large (but legitimate) data item.

By the first week of August 1978, then, the system was again working. Rob West suggested to his superiors that they skip most of the monthly reports that were overdue, in consideration of the large amount of resources required for

a single run of the system and the amount that we were already behind schedule. They agreed that those reports weren't worth producing at this point. An effort would be made to produce a June report more or less on schedule and to keep up on time delivery from that point on. The last week in July a major corporate reorganization had been announced, but so far no one knew what the impact would be on MIS.

At this point I was responsible for 3 programmers and 5 systems. Beth was working on the COBOL part of CAS (the extract programs that provided data to the system). She was also working on the only CAS feeder system that we controlled, but that system required little support and had no requested changes. The other two programmers, Sharon and Bob, worked on the remaining three systems under our control. These were vendor supplied packages that had few problems, but required many small maintenance jobs, mainly changes in report headings, etc. At the end of August I went back to school full time and became a part time employee. My boss began to look for a replacement for me as the group leader.

During August, Beth began to try and get CAS back into shape and try again to put it into production. I added a couple edit programs to the system to allow the users to make their own changes to system tables. We also began to investigate the backlog of requested changes to CAS.

Although the system hadn't been accepted until the end of 1976, there were several projects outstanding that dated from the last few months of that year, requesting changes to CAS. On reviewing them, we discovered that they ranged from simple report modifications to major processing logic changes that would correct, simplify, and extend the system. For instance, they wanted to not only report data rejected by the extracts, but to be able to correct it and include the corrections in later processing, so that the data used would match what we expected from the feeder systems. They also wanted to be able to trace how numbers were produced for the allocation reports.

We discussed these with the users, and I was able to make changes to the system to get two of the simpler requests out of the way. One additional project, involving a major change to the way one of the tables was used in an extract program, was also chosen. Beth and I began to work out the best way to handle the change to processing. When I went back to school she was beginning to make program and file changes necessary to the project. But other things came up and that project was set aside for a time.

At that point we began to get more information about the corporate reorganization. It involved reassignment of most offices to a new regional and territorial organization. Agents had been reassigned to new offices, and

the whole change was to be effective back to the first of January 1978. All reports were to show the realigned structure in the July runs. That meant that year-to-date and other long term reports (such as our own) would have to have all of their data realigned for all earlier periods to meet this requirement. Also, because of agent reassignment, much of the source data for a number of systems would have to be entirely reprocessed. This is because premiums and claims are reported to the office that the agent responsible for that policy is currently assigned to. Since most reports are only at the office level, data files are frequently summarized to that level as they are first processed, thus losing the information needed for this reorganization. And because office numbers in most of the numbering systems being used include a region indicator as one of the code characters, many office location codes would have to be recoded.

This is not the first time this had happened. The last reorganization, during the initial CAS development in 1976, had been the cause of much of the inefficiency of the system. Fortunately, CAS didn't have any data to worry about that time. It had been the other systems that suffered. Once the executive staff makes such a decision, it seems that they are unapproachable about changing it. Our response was limited to persuading Sue to champion a

proposal for a corporate-wide numbering system. It would provide a common set of codes for all new systems and reports, to replace the codes developed individually for each system. It wouldn't avoid reorganizations, but would make the consequences easier to bear, as well as making reports from different systems much more comparable.

The several systems which provide information to CAS unfortunately each use a different numbering system for coding offices, lines of insurance, etc. Not only are the codes different, but the level of detail is very different from one system to another. Yet in order for CAS to utilize the data, it has to be matched across systems. To accomplish this, a number of translation tables are maintained. The system then attempts to translate each code it receives in the extract processing to a single common numbering system set up by CAD. Since the incoming codes would all be changing, the cross reference tables would all have to be rebuilt to process the recoded data. We would have to wait for the source data to be converted anyway before we could run the July report. There were some minor changes necessary to the extract programs to print the correct headings, etc., but no changes were required to the PL/1 program to handle the reorganization.

The project to modify the extract tables was put on hold while Beth worked on the changes for the reorgan-

ization. But the first week in September, Beth announced that she was getting married -- and that she was taking a job on the other side of the state to be closer to where they had decided to live. I was now the only person that knew anything about the CAS programs, and I hadn't ever bothered looking at the COBOL programs. Fortunately, Beth had almost everything set up for the July run before she left. Sharon, one of the other two programmers in our group was reassigned to work on CAS. Initially, Sharon was asked just to figure out the many necessary steps to get the system through the next processing cycle, from which order to run the jobs and which files to use, to which version of which program was correct. (There were in several cases multiple versions of the programs that Beth and others had set up for testing various changes. Beth had made the changes for the July run, but had forgotten to tell us which version to use.) Eventually, Sharon was to take over maintenance of the extract programs as well.

In October, we actually got the July report out. The size of the translation tables used by the extracts had increased significantly because of the new organizational structure. Because of the increased complexity of the new organization and the larger tables generating extracted data, the processing tables that drive the actual expense allocation program became much larger as well. The July

CAS REPORTING BACKLOGS

1976	June	First complete test (Step 1 reports only)
	July	
	August	
	September	(occasional tests)
	October	
	November	
	December	
1977	January	
	February	
	March	January reports run
	April	
	May	February and March reports run
	June	April reports run
	July	May reports run
	August	
	September	June and July reports run
	October	August reports run
	November	September reports run
	December	CAS "Signed Off" by CAD
1978	January	October and November reports run
	February	December reports run
	March	January reports (first officially issued)
	April	
	May	
	June	
	July	February reports run
	August	March reports run
	September	
	October	July reports run
	November	
	December	
1979	January	October reports run
	February	November and December reports run
	March	
	April	
	May	
	June	March and revised December 1978 reports
	July	June reports run
	August	
	September	
	October	September reports run
	November	October reports run
	December	
1980	January	
	February	December reports run
	March	January reports run
	April	February reports run
	May	March and April reports run

Figure 7

run did complete successfully, but required nearly twice as much disk space and twice as much computer time as it had previously. For several months then, they continued to add more and more to the processing tables each run, in order to handle the many special cases that had been dropped or done incorrectly. So, the resource requirements continued to grow every time that CAS was run.

A decision was again made to skip several months of processing because July came out so late. CAD's management wanted to get the system closer to on schedule rather than have the additional reports.

In October, my boss hired two new people. He had finally been successful in finding a new group leader, but Andy didn't know PL/1. So, instead of Andy taking over the CAS group, a new group was formed for him. He took over the three systems unrelated to CAS that we supported, along with Bob from my group, who worked on them (and another programmer from my boss's other group).

The second person hired was a PL/1 contract programmer. Marie was hired to join CAS until a permanent person could be found (and probably also to serve as a backup in case I should decide to leave). She and Sharon once again began the job of trying to get the system ready to put into production. And I again began to review the list of project requests for CAS. I attempted to explain

the complicated table structure necessary for the project Beth and I had started, but neither Sharon nor Marie seemed to understand. So, I left that project on hold for the time being and we made plans to begin work on a couple of the simpler requests.

At the end of October, management announced their desire for us to run an October report (November had been the next one we planned on producing). One of the projects that we had activated required a complete run of the system with a special set of processing control tables. (It was to produce a test file for use by another system). We decided to make the special run for October as well, so that we wouldn't have to make any extra extract runs and we would have a normal run to compare totals with. We encountered a number of problems with the extracts ourselves, and were running late, when OAE notified us that one of the files they provide us had not been realigned properly before. That problem was corrected by mid-December. But because of our growing resource requirements and the increasingly heavy demands on the system in general, our final processing run for October wasn't completed until New Year's Day, 1979. The special run was completed the following weekend.

During January we didn't have any projects directly requested by the users underway. Sharon was working on

several changes that would be required in the extract programs to process January data. These were still primarily changes resulting from the corporate reorganization -- often undoing temporary changes that were made mid-year to handle the conversion.

Marie had been brought in as a PL/1 programmer, but as it turned out, she had had a course but had never used the language before. She had no interest in learning about CAS, and since she was temporary it didn't seem worth forcing the issue. She continued the process of cleaning up and documenting the system so that we could put it into production. My boss Lou and I agreed that Marie was not going to be of much use to us after the documentation was completed, and she was told that her contract would be allowed to expire at the end of the month.

We felt that the operation of the system was becoming so impractical, that rather than my working on user projects, I would spend my time on trying to cut the resources required for a run of CAS. I then spent a couple of weeks analyzing CAS, trying to isolate areas that we could improve the system. I read through the code looking for obvious flaws that could be streamlined, but most of the code was so involved that rearranging the logic was dangerous. So I looked for places that we might effect change, without affecting logic, such as disk usage.

Toward the end of January, I had come to some conclusions. First, the system would have to essentially be rewritten to avoid some central problems with the design. The entire system was table driven, under the control of the users. The idea of processing tables, however, had been firmly followed throughout the design, with no account taken of the fact that the tables might not be "full". The incoming data and the flexibility of the users' controls allowed the tables to be very sparsely filled. Imagine a 10 by 10 table to be processed, with all but 8 of the entries being zero. If we want to get row and column totals for the table, we have to look at each of the 100 numbers even though most are zeroes. All of the processing in the system worked on that basis, except that the tables were much larger and the calculations, often quite complex, were performed even if the number was a zero. Each individual process in the program would have to be looked at to determine what rules might be used to recognize that some work might be bypassed. Or the system could be rewritten to ignore the irrelevant parts of each table or to not treat it as a table at all.

The second conclusion was that because the disk space required was managed the same way, there was probably very little data using all of that space. When I did some tests, I discovered that only 5-10% of the records created

had any non-zero entries (and many of those had very few non-zero entries). So, without changing the logic of the program, I could decrease the disk requirements by almost 90%, if I were to write some routines to intercept all disk read and write requests and manage the space myself independant of what the program was doing. I could that way "fool" the program by not storing the record unless there was a non-zero entry. Then, when the record was requested, if it wasn't in the file my routine would recognize that a record of all zeroes should be returned. In this way, I could save eight of the nine disk drives that we currently required without affecting the program logic. Compared to improving the processing, improving the disk space requirements would be fairly simple. At the same time, it would isolate all data storage and retrieval into routines that could later be replaced by DBMS access routines.

There were some other thoughts at this point. The system was complicated, poorly understood, difficult to use, and impossible to run. Many of the problems seemed to stem from the original design flaws. Rewriting the system would allow us to take advantage of all the things that had been learned the first time, everything we had learned since, and design for all of the changes and proposed changes. It would also be possible to design the system to

be run as a series of transactions instead of a single massive run. The idea of an allocation is to take an expense that is identified by a coded location, insurance line, etc. and split it up among the locations, etc. that we have determined should be charged for part of that expense. For instance, regional administrative expenses may be split among the branch offices in that region on the basis of the total number or size of policies written by each. The processing was controlled by exhaustively listing all expense codes in tables, with instructions for how the money is to be divided, and which codes should receive the allocation. Those tables amount to no more than a list of instructions, each saying "take the expense from there, divide it up this way, and put it here and here and here."

If we could somehow process each one of those allocation instructions separately, it would be possible to stop the system at any time. Assuming that we had marked each of them as done when we finished processing it, we could start back up without losing what had already been accomplished. As it was, the system was trying to do too much "in its head" instead of keeping records. If stopped, it not only "forgot" which entries it had already processed, but lost track of where it had recorded everything it had "written down". Ideally, if enough information were recorded after each instruction was processed, we could

process each individually, or in any size group. Then, we could have transactions that said "do everything necessary to report office X", or "run the next 5 instructions".

This would make it possible to break any run into small enough pieces to make it much more viable operationally. That, in turn suggested that we might be able to allow on-line inqiries of the sort that would process a single transaction to answer a question (such as: What were the total expenses charged to fire insurance at the Houston office?). The input and output files were reasonably small, only the intermediate work space was at present too large. If all the data were stored in on-line files, we could also allow on-line access to original and processed data to replace thousands of pages of reports. Simply by capturing the output in a file, we would have the advantage of being able to reprint the reports. As the system was, a lost report or bad heading could mean a complete 15-20 hour rerun; all of the output was printed directly to paper as a product of the monstrous final run. The massive temporary files would probably benefit from being put onto a DBMS.

I also realized that development of a new system could proceed much faster than fixing the old. The reason was that where it was impossible to get more than one test of the old system in a week, we would be able to test a better

designed system as often as needed, both because it would require far less resources to begin with, and because we could run much smaller, more specific, tests. The testing situation alone might make it faster to rewrite the system completely than to make just a few of the requested changes to the old one. At the same time it seemed that it would be much easier to rewrite than to fix.

In February, we convinced Rob West and the other users that we couldn't continue to run the system without making some improvements. I began to make the changes to use disk space better. At one of our meetings, we also suggested to the users that they should consider a complete rewrite of the system soon, since these changes would solve only the disk problems. We explained the reasoning for the proposal, but at the time they were more interested in seeing some of the improvements they wanted than in considering any proposal that would hold up progress for several months.

The last week in February we completed the December 1978 reports. A decision by user management followed shortly to run the system on a quarterly reporting basis instead of monthly. This was based on the high resource requirements and long time delays in getting reports out making any more frequent schedule wasteful.

Sharon left in March for higher pay, leaving me as the

entire CAS group. My boss's response to her notice was prompt, and he was fortunate for once to find a PL/1 programmer quickly to replace her. Brad was able to start two days before she left. He didn't know any COBOL, but was a good programmer and was able to pick up what he needed quickly. He took over the extract programs, and in the next few weeks he learned enough about them to begin to question the way they worked. For the first time someone looked at the programs to see what they were doing. He began to find problems and suggest changes to CAD to improve the reports. The totals on our reports began to resemble those on the systems we were extracting from for the first time. Instead of blindly rejecting errors as they had in the past, the programs were changed to report and total the rejected data. Errors in the processing logic were found that had made some of the reported figures false for the past 3 years. And he slowly replaced all of the confused, patched code with consistent, internally documented programs. When I explained the deferred project with the table changes to him, he understood and suggested further changes. So we reopened the project and he rewrote the programs to complete it.

While I was making the changes to improve disk usage, I was also able to include a couple of other requested improvements from their list of requests. We were slowly

convincing CAD that we really did know what we were talking about. We were serious about wanting to improve CAS and were really giving them results for the first time since the task force dissolved. In May, the system was tested with a number of changes in place. We again approached CAD with the suggestion that we rewrite the system, and this time they were more receptive. We had begun regular meetings with them in February to monitor the progress of changes in the system and to discuss other changes that they felt were needed (many of which they had never bothered to request). So, when we again brought up the subject of a rewrite in May, both groups were much more familiar with the problems etc., that the other group was facing.

The discussions continued for the next two months; we were now meeting once a week rather than twice a week as before. Frequently discussion turned to what might be required for a rewrite and what advantages there would be to it. Not only would we be able to include all of their outstanding requests in the new design, but a new system would benefit from the latest in program design. These were benefits to us, as the people who would have to maintain the system, but also would make it easier for us to modify the system, so that in turn we could be more responsive to future need for change. They could run what

they needed, when they needed it, as well. In July, CAD was not only convinced but presented the idea to their management officially. The first week of August their management decided that the idea was promising and requested a feasibility study for a rewrite of CAS.

In the meantime, it had been discovered in May that one of our source files had not been realigned correctly after the reorganization. The 1979 data was OK, but the December 1978 report would have to be rerun when the input data had been reworked. The December 1978 revised report was finally issued in early June, shortly before we completed the March 1979 run. (The March report also served as the test of our changes.) The second quarter CAS reports were completed at the end of July. From then on, the system was capable of being run roughly a month behind the end of the reporting month, without any significant intervention from us. Brad and I finished the job of putting the extract programs into production.

From this point on, most of our efforts went into the new CAS. As it existed, CAS was still nearly unmanageable for the users and for DPD, who had to run it. The reports were not being used as planned. However, as the former head of CAD told me:

> We thought perhaps for a change we had a system that we could sell to other companies. It didn't run very well, we knew it had its problems and its limitations -- but we also knew it was better

than anything anyone else had. We had bought so many turkeys from other companies that I don't think anyone would have felt any guilt.

Besides, even though managers weren't using the reports, CAS did serve a purpose. Those reports were inaccurate, but they were the best we had to prepare the (manually prepared financial reports to the government) from. We hadn't planned it, but those reports had become critical to our operation. We couldn't have produced any kind of estimates -- good or bad -- at that much detail without a staff of fifty.

CHAPTER 4

THE NEW CAS

The new system developed slowly from its original conception. There were a number of immediate problems that we faced. First of all, we had a very poor understanding of the existing system that we proposed to replace. Although we were beginning to develop some credibility with the CAD department, Brad was the only full-time person programming. Yet we were proposing to rewrite in a few months time, a system that had taken longer than that to develop originally. If we could do what we said, there was no question that the user wanted it done. But why should they trust us?

By now they were fairly confident that if we continued as we were, that we could and would continue to improve their system. They were bargaining several months of a sure thing, slow progress, against a risky proposition, that we could rewrite the entire system in that time. In addition, they would have to put a great deal of their own effort into the project if they agreed to go with our suggestion.

Through the summer of 1979, we met with CAD and Sue twice weekly and sometimes more often. We discussed in

detail what CAD felt should be in any new system, what they thought was wrong with the old, and how they might like the system to handle each segment that we had identified. Our goal in the discussions was both to educate the users in what might be available and to get as many ideas as possible on the table, so that we wouldn't be surprised by a request that ruined our design later.

In early August, CAD was reorganized and their function was expanded. Anne was hired as Rob's assistant, and began training to take over most of the responsibility for CAS. Bill and Dick were transferred into the department from other parts of the accounting area, and were also assigned to Rob to assist with his other duties. The meetings became especially important, therefore, as a means of bringing Rob's new staff up to date and explaining CAS to them.

Having Anne in the user department began to make things easier for us in MIS. Where Rob West had been with CAS since he joined the task force, his thinking tended to be very much in terms of how the current system worked. In dealing with him, we had to always be careful to use the accounting terms he was familiar with, because the data processing terminology confused him. And we had to always relate our discussions to concrete examples from the old CAS (which we weren't able to do easily). Anne provided

some fresh thoughts, and was quick to understand our suggestions for change. She was an accountant, and talked in terms that Rob understood, but also knew more about data processing than he did, which facilitated communications with us. As she learned about the old system, she was able to provide the concrete examples that helped Rob understand us. She quickly took on the role of intermediary between the two groups.

Shortly before the CAS feasibility study was actually approved in mid-August, we began the design of the database files. Because none of us had ever tried to design files for a DBMS system before, we decided to use a formal design technique. Lou had seen some recent mention of a method referred to as Entity-Relationship Modelling (E-R) in his computer magazines, and located a copy of an article (Chen, 1977) that explained it.

The E-R approach involved separating the underlying structure of the firm from the data being captured. The important elements were defined as Entities, and the Relationships that existed between them were diagrammed. Any relationship that might affect reporting was included. A file structure was then developed from the E-R diagram to allow data to be captured about any entity or relationship as needed. This allows data to be captured in a manner that fits the underlying structure of the firm instead of

collecting it in a form only suitable for the specific application in question. The result should be a database that is useable for any reporting needs, instead of being forced to duplicate the data in a different form for each application. We decided after reviewing the paper, that we would attempt to use E-R and see what we could get out of it.

For the next few days, we reserved a conference room and spent all of our time in it. Sue was invited up, and joined us the first afternoon. She spent what time she could with us until the design was completed (several hours each day). The CAD staff joined us during our normal twice-weekly meeting times, and they, Anne especially, spent a lot of additional time with us as well.

The sessions were very informal, probably best described as "brainstorming sessions". Lou and I, having a better idea of what we were trying to accomplish, generally served the chairman function when it was needed. Everyone seemed to participate freely, each taking the floor to make suggestions, explain points, etc. The wall was covered with charts and diagrams, and we had a blackboard to sketch out ideas on. We were constantly redefining our understanding of the E-R approach as well as restructuring the resulting diagrams. Slowly, a complete relationship diagram for the whole financial area of the firm was devel-

oped.

Once the E-R diagram was completed for this segment of the firm, we began the more technical job of planning the actual data files we needed to capture the required data. This task was handled by the CAS staff, and when we had completed it, we again met with CAD and Sue to explain how the new structures would work. We then took the final step of taking our actual current data files and showing how they would fit the new structure. By demonstrating that we could do that, we satisfied ourselves and our users that we had developed an adequate E-R diagram to meet all current and planned data needs for CAS.

At that point we went back to a more normal operating routine. We met with the Database group to discuss the use of one of their DBMS software packages to store the CAS files we had developed. Bill Vernon, the head of the Database group, was present at that meeting and after some discussion about our plans, he announced that we would use ADABAS and gave us a list of restrictions he "had" to put on us. Since Lou's boss had already told him in a meeting the day before that we would have to use ADABAS for a file of that size, the news came as no surprise. We had heard about most of the restrictions as well in talking to Vernon's staff earlier. But, this meeting served to make both groups officially aware of the conditions we would be

developing CAS under.

During September, we continued work on the CAS feasibility study. We looked back at the minutes of our meetings with CAD from the past few months, to review the items that they felt should be important in a new CAS. Our meetings with them continued twice weekly, reviewing what they had already told us, and getting new ideas and clarification of some earlier points. The MIS team was fairly confident of how to proceed, and began to develop concrete plans for a system to accomplish our goal. We laid out the basic processing steps that would have to be at the core of any such system. Then, based on user requirements, we decided what the initial system would logically contain -- either because they were basic features required by the user or because they would have to be included as part of the core of the new system anyway.

We then proceeded to set up a basic design for the system and developed design rules for individual modules. The system would be highly modular to allow for flexibility as well as for the programming advantages. Functions could be easily added or changed by inserting or modifying modules in the basic system. The system would be based on the processing of individual transactions to accomplish the smallest discrete piece of work possible at one time. This would offer the advantage of making the system interrupt-

able. The old program ran unstoppably for hours regardless of which outputs were really needed, and with no chance to check intermediate results. This, instead, would run only those transactions that were needed to produce the desired results, split into as many steps as necessary. We actually began design and coding of some of the core modules in order to test out the design rules. We also set up the file descriptions for the three ADABAS files we would need, and requested the Database group to develop the interface modules for us.

In order to isolate our system from the problems encountered in reorganizations, we came up with a scheme for storing data separately from the codes for the office, cost center, etc. that the data belonged with. What we did was to establish an additional set of cross reference tables. The first set of tables was to translate the codes in the feeder systems to a common numbering system, for processing and reporting purposes. The new set would then translate those codes into an arbitrary set of sequential numbers for data storage. This way, the data would be stored with codes whose meaning was limited to locating data in the file. The organizational hierarchy and other information in the reporting codes remained separate, so that by simply changing the cross reference tables, we could immediately report in <u>any</u> organizational format desired.

The end of September was then spent trying to establish estimates of how much time and resources would be required for each part of the project. We split the requested features into several groups according to need, difficulty, desirability, etc. After establishing what the minimum requirements would be for an initial system, we split the rest into three logical follow-up development phases, and then made estimates for time and resources for each.

The idea here was to provide a plan that would allow quick initial development, and flexibility in what to add next. Living with the initial system would doubtless suggest some group of changes that would improve its usefulness most for the next phase. But we also wanted it on the record that this was intended to be an evolutionary development, and would not stop after the first version was delivered. We therefore needed to document some plan for future phases and resource estimates, even if they might be completely revised before we got there.

The last week of September, the feasibility study results were written up as a proposal to proceed with the first phase of a proposed four phase development of a new Cost Analysis System. The initial phase would quickly replace the functions of the existing system, and following phases would add on the other features desired. A long

range plan to replace all financial systems was also sketched out.

Sue and Lou put together most of the final document and developed a presentation for user management. It emphasized the capability and flexibility of the database design we had developed and the proposed modular program design. It also stressed some points that the user thought critical, such as auditability and testability, that would be features of the new design. A six to eight month period was suggested for the first development to be completed, depending on specific options chosen. The technical issues involved and specific content of the proposed system (aside from the selling points) were left vague. It did include a proposal that the design be considered as a basis for similar rewrite of all financial systems if it was as successful as we expected.

The following week, Sue made the presentation to CAD and their management. When they reponded favorably, she and Lou helped CAD management revise it for presentation to the senior corporate officers who would be affected. Ed Newcomb, head of the financial reporting area, made that presentation in mid-October. It varied very little from the original, but emphasized even more the future potential of the new design, and dwelled even less on the actual content of the first version. The immediate response was

guarded, but positive.

During the early part of the month, Lou discovered Stan at an open house recruiting session, and hired him for CAS. Stan had no data processing experience, but was bright and had taken PL/1 courses. He looked like a very promising trainee candidate.

Also in October, we received the first ADABAS interface modules. We did the necessary programming and began to test the ADABAS routines. It was immediately apparant that there were problems, and we requested help in solving them from the Database group, since the problem seemed to be in the interface modules. We were told that they would get to it when they could, as they were very busy at the time.

Meanwhile, we were making some initial progress with design and coding of modules for the new system. Since we had no access to the ADABAS files until the interface problem was straightened out, we were concentrating on developing the translation tables that would match the codes from feeding systems. The tables would be necessary for loading the data files into ADABAS, since we intended to use our own translated codes for data storage, and we would need them to translate the data back for reporting purposes as well.

Unfortunately, Brad and Stan didn't seem to be hitting

it off too well, but that hadn't become a real problem yet. Sue officially transferred from SMD to the new Financial Systems group as a result of the breakup of SMD that had been announced in August, but her job changed very little. Stan was assigned the job of defining all of the system we had designed into the data dictionary. We hoped to use this as a quick introduction for him to what had been accomplished so far, by involving him somewhat in all of the modules. Use of the data dictionary system was expected to further improve our ability to make modifications to the new system easily, by allowing us to quickly find all of the modules affected by any change.

In mid-November we were officially requested, by the Controller and the Treasurer jointly, to proceed with a rewrite of the Cost Analysis System. We were to follow the recommendations that we made in the feasibility study proposal for Phase 1 of the project. They requested that the project begin immediately and be completed by June 30, 1980.

We immediately ordered the terminals that our users would require for on-line use of the system. The ADABAS interface still wasn't working, so we again asked for help to get it fixed. We also requested assistance from the group that handled CICS interfaces, which would provide terminal session monitor and control facilities for the

user terminals. (All user department terminals run through the CICS monitor system, which accepts typed input and forwards it to processing programs. CICS only allows users with proper authorization to run each program, which provides a good measure of control.)

Design and coding of CAS modules was proceeding reasonably well. Meetings with CAD were now focusing on the specifics of what they wanted to see in a given situation: formats, prompting, etc., and exact contents of reports, ie, what fields do we need to add up to create this reported number. December arrived and progressed with little major to show. We still had no access to our ADABAS files, so we couldn't put together the quick prototype that we had hoped to provide.

Toward the end of December, there was still nothing to show for our effort, and Rob and Sue were beginning to be visibly uncomfortable with not having any specifications in writing. Over the next couple of meetings, CAD presented their view of what they expected in the new CAS. We, in turn, presented an explanation of how the system would accomplish those tasks.

Finally, just before Christmas, we got some help with the ADABAS problem. The solution was to give us an all new interface, which operated somewhat differently than the other had been intended to. In early January we got some

successful tests accessing the ADABAS files, but due to a couple of major changes we had just made to our programs, we needed to fix the existing modules before completing a prototype. We still hadn't received any response from the CICS group, and approached them again. They announced that they would get to us when they had the time. Lou decided that we didn't have the time, and sent Andy to a CICS class for a week, hoping that he would be able to help us set up the interface ourselves afterward.

Brad and Stan seemed to rub each other the wrong way, and were unable to discuss programs quietly with each other. They had had several arguments for no good reason in the past few weeks, usually about unimportant points of programming style. More of my time was being spent trying to provide the necessary communication between them, so that program development could continue at a reasonable pace. All too often, each would make assumptions about how the other's module would work that turned out to be wrong.

The problem was not helped by the fact that everyone was working at different times. Brad took Wednesdays off and worked Saturdays, and he started mornings at seven so he could leave an hour and a half early in the afternoon. Stan worked normal hours, and I tended to work mostly evening and weekend hours during school. Because the core of the system still wasn't working, each person was likely

to come in and find a program critical to his testing not working. If the person responsible wasn't around, the options were to be frustrated, or to try and fix the other person's code. So it was not unusual to come in and either find that the code you were last working on had been changed, or that someone else had caused a new problem for your testing. Brad was about ready to quit.

At the end of January, Lou was having problems locating an experienced PL/1 programmer to join the group. We decided that even a trainee would be better than no programmer, and he hired the next promising PL/1 trained programmer who applied to the company. Carol was quiet and pleasant and seemed that she would be able to work successfully with Brad and Stan. But it seemed that the tension between the two bothered her, and although she got along much better with them than they did with each other, her working relationship with each of them was less than ideal.

Stan had been elected to write an interactive program for our terminals that would run CAS, until we could get CICS help, so that we could try to get a prototype system running for demonstration. He was still having problems getting that to run properly. Brad, in an effort to avoid conflict with Stan, started work on revising the extract programs to provide data properly for the new system.

Sue had persuaded the group to develop more formal

specifications for the system. Our meetings were now directed at finalizing those. CAD and Sue wrote up the specifications together, and then we reviewed them in the meetings and agreed on any changes needed. By early February a fairly complete set of requirements was documented in that way.

When Andy came back from CICS class, he understood the problem well enough to make some suggestions. But we had to explain the system to him before he could be of much help. Unfortunately, he had never sat in on any of our design sessions, so he was coming into this cold. After a couple weeks of trying to explain the system to him, we finally just gave him a copy of part of the system to work with. He then tried to modify that series of modules to run under CICS. He soon discovered that once he had converted the programs, he couldn't use them because the necessary interfaces between PL/1 and CICS hadn't ever been used in our shop. We again requested that the CICS group give us help, but this time limited to providing the missing interfaces.

However, Andy's suggestions about how files were accessed under CICS, etc. allowed us to make a few design changes that would simplify the conversion later. The most important of these changes was our decision to put the cross reference tables onto the RAMIS DBMS. This is some-

thing we had planned to do eventually, but because CICS wouldn't allow the files we were currently using, we decided to proceed immediately. This required, however, that we get an interface between our PL/1 programs and RAMIS set up. We went to the Database group for assistance, but they wouldn't be able to give us much time for several weeks. Instead, they offered to give us the information to write our own interface modules. With this new change, we were again without a working system core until it worked. We had access to the data in ADABAS, but now couldn't access the conversion tables, and we still didn't have a working monitor program.

We went to talk to OAE about getting the data that we needed from their file. They sounded confused, but there didn't seem to be any real difficulty in getting what we needed. Brad's work with the extract programs was nearing the point where he would need that information from them soon.

Our new terminals arrived in MIS, several months late, but the user's terminals still hadn't been delivered. We continued on with nothing to show for our effort into March. It was discovered that no one in the firm had ever used the RAMIS interface with PL/1 before, and a module would have to be generated in the RAMIS system before our interface would work. Again there were delays.

The terminals were finally installed in CAD. The RAMIS interface was established and tested successfully. The programs for the CICS interface were delivered by IBM and Andy began testing on those. Meetings with the user were changed to once a week due to lack of anything to talk about. Relations within our group remained strained; Brad was again talking about leaving. Most of my time was taken up with trying to keep up enough communications to keep the group working.

By early April we were finally showing signs of progress. Stan's on-line monitor was fixed, and it was beginning to look like we would soon be able to edit tables and a few other simple functions on-line. There were still a few problems in the programs, but the system core was reaching the stage where problems were generally isolated to the function being tested. CICS problems were now past the interface and in the actual programs being tested. RAMIS and ADABAS interfaces seemed to function well. It seemed that we could finally spend our time working on our system instead of interfaces with others.

The old system was run for February 1980 and failed for the first time since we started the rewrite. The problem was in the allocation processing in the big PL/1 program. When I found the problem, it turned out to be a weakness in the original rules for allocating expenses, and

it had died trying to process legitimate data. It had just happened that that data combination had never before occurred since the system was first run. The solution was simply for the user to change the control tables to bypass the illegal calculation. They made that change and tried again, but this time we ran in to operational problems, and our files were destroyed mid-run. We had more operational problems for the next two attempts, and then finally a successful run at the end of April. We then tried to run March reporting and had similar problems because of the huge resource requirements. All told, I spent nearly all of my time working on the old system for a month.

When I returned to work on the new system, I found that communications had broken down again, and several modules that had been working didn't work any longer. Within a couple days, we were able to isolate and solve the problems. We got the table edits working successfully on our terminals, and invited the CAD staff to come up and try them. This is the first thing we had been able to present them.

The first week in April, the system was very far behind schedule, and far from complete. But we were making visible progress, and all of the known roadblocks were finally out of our way. Andy finally had a PL/1 program run successfully under CICS and was preparing to convert

the working part of our system. There was even a chance that we might meet the June 30 completion date that we had originally promised.

CHAPTER 5

PROBLEMS ENCOUNTERED WITH CAS

This chapter looks more deeply at some of the problems we had in dealing with other groups in MIS. The focus here is mostly on why, from their point of view, these problems might have occurred.

Technology

The Systems Technology Department is responsible for all hardware and systems programs -- operating systems, programming languages, telecommunications software, etc. This includes providing and maintaining these facilities as well as providing technical expertise to those using them.

In the summer of 1978, a personal difference of opinion between the head of Technology and his boss led to his being fired. Shortly afterwards, a group of his staff responded to what they perceived as an unfair firing by themselves quitting. This cut the programming staff in half and essentially removed the entire management group. The remaining programmers in the group were very overworked and, to keep things running, several of them were asked to take on management responsibilities as well. Many of them were very technically oriented and balked at being asked to

serve as managers. None were happy with the extra unfamiliar workload. For the next couple months a number of system problems resulted. As Technology's work fell behind, the online systems slowed to unacceptable response times, batch jobs began to take days to get back, and many ongoing projects were simply frozen because no one had time for them.

New equipment had been installed by IBM in June and July and had to be signed for soon or it would be removed. No one in Technology had had time to fully test it yet, and no one left really had the authority to sign. None of them wanted the risk of signing for untested equipment to be on his or her own head. Reportedly, someone finally solved the problem by signing the fired manager's name one night to avoid losing the equipment.

Not surprisingly, under pressure and an increased workload, without management, and with little but criticism heard from other departments, one by one, the remaining Technology staff also began to leave. The firm was trying to fill the vacant slots, but it was taking time.

It was early 1979 before the system problems began to be resolved. A management structure was slowly built back up, mostly by hiring from outside the company (it isn't clear whether anyone outside of Technology, with the exception of two DPD people who actually took positions,

were offered a chance to move into the department. However, no such offer was ever made public). The staff remaining within Technology without exception refused any offers made to them. All who were offered advancement to management emphatically refused. The reasons they suggested to me were: 1) they were technicians and didn't want to be managers, and 2) they didn't want their necks on the chopping block. The managers seemed to pick up most of the blame for any problems, and most of them had had enough of being the scapegoats.

Toward the summer of 1979 there were rumors about in the MIS division that a third mainframe computer was being ordered. And, in fact, in early October a new machine was installed. It was bigger and faster than the two existing machines and was to be dedicated to testing. It was assumed by the Technology staff that such a large increase in computer resources, nearly doubling existing capacity, would solve many of the problems of long online response times and unacceptable batch turnaround times. It was the simplest solution to a pressing problem, and the argument convinced the people who controlled the funds. (A pool was started in APD betting on the number of days it would take for the new machine to be as saturated as the old.)

It helped for about a month. The programmers quickly began to do more work, taking advantage of the improvements

to submit more tests, with the hope of seeing results in a reasonable time. The online environment improved so that in the same amount of time using a terminal they could make more requests of the machine. Very soon, all three machines were full to capacity and the problems returned. New measures were called for.

There had been an informal online users group for the past couple of years. It consisted of several of the junior managers in APD who had people trying to do online program development. They had met regularly with each other to discuss ways that the testing resources could be used more effectively, especially with the goal of improving system responsiveness. They believed that such an improvement would in turn improve programmer productivity. They had sent memos and occasionally met with the Technology group to suggest things that they had come up with. But, until late 1979, the group had not received much enthusiasm from Technology. The group hadn't yet been able to push through any significant changes. However, with the change of staff in Technology and the growing pressure for solution of the system problems, the user group was suddenly taken more seriously. They began having weekly meetings with Technology staff towards the end of December 1979.

One of the first results was a plan to try putting a

terminal on every programmer's desk. The online usage would be restricted to make only the most efficient commands available to programmers. It was felt that putting terminals on every desk would encourage programmers to reduce their usage of printed output and cut back on many other wasteful testing practices (such as submitting several versions of a single test at once, so that when results were finally returned there was some hope of having a solution among them). The hope was, that although online usage would be greatly increased, there would be more efficient use of available resources and an overall drop in the load on the computers.

CAS was chosen as a pilot group for testing the concept. This was because we were beginning a major development effort; because we were believed to be heavy but intelligent users of the system already; and because our manager was the ringleader of the user group that suggested it. Our terminals were scheduled to be installed at the end of December 1979. A few other programmers were also selected for the pilot group, on roughly the same basis (especially that their managers were part of the user group).

Due to technical problems encountered in setting up the necessary hardware and software for the test, the Technology group was unable to get everything ready to

support the new terminals until late February. IBM, as it turned out, was unable to deliver all of the necessary hardware until the first week of February anyway. When installed, the test group terminals more than doubled the number of terminals connected to the test machine.

In January, when our user's terminals were scheduled to be installed, we contacted Technology to find out what the status of those terminals was and discovered that they had "forgotten" to order them. In fact, they had forgotten to order terminals that had been ordered for several other user departments as well. It appears that they had decided that a moratorium on user terminals was called for until the effect of the new programmer terminals was determined (even though the user terminals were intended to be connected to an entirely different machine!).

They immediately ordered the missing terminals when they "discovered" their mistake, but by that point IBM was backlogged on orders. It seemed unlikely that we would have terminals for users until late May at least. However, just a few weeks later, it was noticed by an SMD manager that some of the programmer terminals were being replaced with newer models. It was "suggested" that the Technology department install the old terminals temporarily in the user departments rather than send them back to IBM. This was done in early March.

Considering the problems that Technology faced already with extremely heavy machine usage, it isn't too surprising that they might be reluctant to install more terminals in user sites. Where programmer terminals might conceivably reduce the machine load, it was very unlikely that any new user application would, especially if it required online terminals. There was an attempt made in January, shortly after the forgotten order was brought to their attention, to officially halt the installations. They wanted a postponement on the basis that they didn't have enough information to adequately assess the impact on existing systems, or the resources that would be required to support them. Although they had initially approved our request in October when we made it, they didn't feel that they could go ahead with installation of our users' terminals until we had better explained what facilities we had planned on providing through the terminals and what volume of usage we envisioned. But, when we were able to immediately provide that information, they dropped the effort to halt our user terminals.

OAE

The OAE group, like CAS, is a financial reporting group within APD. They are responsible for one of the company's most important internal operating documents, the

Operations Analysis Exhibit. It is a monthly report that matches premium income and claims against policies to show underwriting profitability of agents and local offices. The system accounts for many of the complexities of the insurance business (such as the practice of selling portions of policies to other firms to spread the risk), and it carries a lot of highly detailed information to provide a number of reports showing office workloads, etc.

The reports that it generates are used by many managers to measure and monitor performance. Because of the system's size and perceived complexity, OAE is fairly well respected by those in MIS. The significance of the reports and their good record for on-schedule delivery of reports each month means that they are also well regarded outside MIS as well.

The OAE group provides 3 tape files to CAS each month. These 3 files, along with 3 additional files from other systems, together provide all of the information necessary to allocate costs within CAS.

We approached the OAE group about the availability of additional information from their master file during the course of our feasibility study. Our needs at that time were fairly vague and the contact was obviously due to a feasibility study. We made it clear that we were simply gathering information rather than intending to use the

results in the near future. Their responses were very positive; yes, the information we wanted was available from their files, and yes, they could provide it for us without too much effort or warning. Since our mission was simply to get needed facts, we did not think to document the conversations. We simply incorporated our findings that the required information was easily available into our planning.

I think it was Rob West who, during the database design sessions, suggested that it was beginning to look like we could take over OAE. The files would include much of the OAE data anyway, and our design included the necessary provisions to duplicate much of the OAE processing to produce some of our reports. The MIS people present probably took it more seriously than he did, because technically it really did seem feasible. The suggestion was laughed off at the time, but was brought back up several more times in the following months. Sue, our SMD representative commented that that was how empire building was accomplished in organizations, by gradually replacing other groups. The original suggestion was revived yet again in March 1980, when we began having problems getting the cooperation of the OAE group.

In January of 1980, we went back to OAE looking for help in finalizing plans to use the additional information.

We needed file names and descriptions, etc., from them. Brief discussions were carried on over the next two months between OAE and CAS. Their responses to us seemed to change from day to day. But, as yet no one was bothering to record exactly what was agreed on in the meetings.

We slowly began to realize that from one meeting to the next the files that they recommended that we use would change, that we didn't know exactly how many files thay had or what exactly was in any of them. We were dealing almost exclusively with one person, Jerry, who was a senior programmer in the group. Jerry seemed to have a better knowledge of their system overall than anyone else that we had dealt with. Still, it seemed that either we were getting a runaround or he didn't really understand what we needed.

He had once suggested that we use his source file and duplicate all of the daily processing done by their system, as necessary, to produce what we needed. At other times, he insisted that it would be far more practical to use an appropriately summarized and processed file, with monthly or daily data. Yet other times it seemed best to him that we duplicate a small part of their processing to use a file at a much more "appropriate" level of detail. The possible files at that level of detail as he described them seemed somehow to have the same detail as the most detailed and

also as the least detailed.

Finally, the first week in March, Sue (who had also sat in on some of our meetings with OAE) decided that enough was enough, and started recording her understanding of each meeting as it was held. She sent the resulting memo to Jerry and several managers who might be interested.

Two weeks later (and two memos later), we had a tentative set of files, their contents, and a description of which processing exactly we would have to duplicate to get our extracted information to the right level of detail, etc. But, suddenly Jerry was unable to find a half hour to meet with us "because of the heavy demand" for his time. So we were again being held up; we had almost enough information to write specifications for the extraction of the new data, but not quite enough without talking to Jerry again.

From their point of view, it now seems obvious why they would not be too eager to help us. The OAE system was a large, many faceted system. It provides several critical corporate reports and has always performed reliably. The system supports a large number of programmers, and it commands priorities over MIS resources because of the importance of its output. As a result, the programmers in the group enjoy a certain amount of prestige in the MIS

organization. That their system runs well is also important in a shop where many of the systems have major bugs and/or operational problems.

We essentially came to them asking for their help, having nothing to offer them in return. If our system is a success, we will get the credit; if it should fail, we take the blame. The OAE group might get some credit if we were to request them to do the programming to provide the new data, but it would be a small, rather unimportant project that they would get credit for. Otherwise, their time and effort in helping us would officially get them no credit whatsoever.

Meanwhile, we were asking to take a part of their system. At the very least, we would be copying more of their master file than at present, and we even discussed going back to their source data and duplicating a significant part of their system. That would be valuable to us, since we would get the data a month earlier and could skip a number of processing steps that they require. Without adding a lot of overhead to our system we would be able to start processing their inputs to our system three weeks earlier, making the timely production of our reports more likely.

But, in doing so, we would not only reduce our dependence on OAE, but we would have an opportunity to show them

up by publishing our reports before their own were ready. Because we were using advanced programming techniques that their system did not incorporate, we also might demonstrate how inefficient their system is. Although the plan to use CAS as a first test before converting all financial applications had not been widely publicized, it is quite possible that there had been rumors about it. And, if we were to fail, their system would remain untouched. But if we did well, their system would be demonstrated to be a good candidate for conversion, because we would be duplicating part of it anyway.

The Data Base Group

The database group in MIS has responsibility for all database management systems (DBMS) and the data dictionary. This is the largest group in APD without a corresponding function outside of MIS. They are responsible for providing interface modules, maintaining the integrity of all DBMS files, assistance in design efforts to put these facilities to proper use, and maintenance of the DBMS and data dictionary software.

Bill Vernon, the head of the group, is a senior manager (VP level). He is widely considered to be brilliant, but most people in MIS seem to find him obnoxious and difficult to deal with. He apparantly is a good

problem solver to have around in a crisis (which may explain his power in MIS), but otherwise tends to spend more time at lunch than he does working. The group, like most of MIS has had major turnover problems; very few of them have been with the company for more than one year.

Our dealings with the database group have been pretty successful in general. There have been a lot of delays because their workload is heavy, but the individuals that we have dealt with (aside from Vernon) have been friendly and helpful. There was one problem period in September 1979 when Vernon himself decided to write an interface program that we needed. It didn't work, and he wouldn't fix it or give it to one of his staff to work on. We were unable to get the information from him that was necessary for us to fix it ourselves. However, one of his staff came to the rescue, suggesting that we try a new type of interface that she wanted a chance to test. Although there was a longer learning period involved for us than we had hoped for, this averted a major holdup, and by using the new interface we were able to deal directly with her from that point on.

Some of the more interesting interactions with the database group have been indirect. Our decision to use the ADABAS DBMS to store our data was essentially ordered by our senior management. Apparantly the basis for the deci-

sion was a study done at some time in the past (the database staff members I asked were unaware of any such study in the last two years -- the longest any of them has been at Worldwide -- but agreed that the choice was appropriate from what they knew. But, since both the available DBMS's are updated several times each year, the results of such a study could be quite wrong by now anyway.)

Of the restrictions placed on our use of ADABAS, some, such as the number of files we were allowed to use, were seemingly arbitrary decisions offered by our managers. Most often such rules seemed to have originated from the head of the database group, even though we didn't hear about them from him. For instance, we were warned earlier by one of his staff that Vernon would not allow us to use more than 3 or 4 files, but the actual restriction was placed on us in a meeting with the manager above him.

The important thing here is that these decisions were made for us without any real knowledge of what we were trying to do or how. They were handed to us as mandates. Yet, when we asked the people in the database group later, there seemed to be logical explanations for the decisions (some made with incomplete knowledge however). But, we had not been offered explanations at the time; they were management decisions as far as we were concerned, and we simply had to design around them.

CHAPTER 6

CONCLUSIONS

We tried a new (to the company) approach in project design. The standard procedure was for SMD to develop formal specifications based on the project request (and supplemental work with the requesting department when necessary). Then the programming group would work directly from the completed specifications to complete the project. We, however, attempted to begin work without any preset specifications to restrain the design. The idea was to give them a "quick and dirty" prototype system that solved some of their problems as soon as possible, and then to use that as a basis for further discussion. We could then evolve a design that met their needs, and then go back and "clean up" the programs in each section as the design became finalized.

This was attempted for several reasons. First, the project was not well enough understood by either us, SMD, or the users to provide detailed specifications. Rob West had the best knowledge of the old system, and probably the best idea of what was wrong with it from the user point of view. But his long exposure to the old system had narrowed his perception of possible improvements. Anne, and Rob's

other new people, had some good ideas but, like the MIS staff, had little understanding of the complications of cost allocation. The MIS staff had many ideas about how to improve the interface with the user, making the system easier to work with. We also had ideas about specific technical improvements, such using a DBMS to improve disk usage, to solve known operational problems. But we really didn't know what the processing steps in CAS were. No one person had the complete picture.

The mechanisms of processing data, etc. still needed to be worked out, and the options available for handling inputs and outputs needed to be explored. We hoped that our approach would help speed up the process of finalizing the system design. On our side, it seemed like the only way to give the users an idea of what we might be able to produce so that they in turn could make reasonably informed choices between options we might suggest.

By giving them something to work with, we hoped to encourage insights and suggestions before the design of the system became fixed. Rather than explain, we would be able to show; we could test their ideas instead of forcing them to live with them. It would also be a vehicle to encourage user participation in the design. It seemed otherwise very difficult to develop much user participation in a project requiring as much technical sophistication as this would.

This would force the issue. There would be no chance for them to be frustrated by a "fixed" design; the design would not be fixed until they were satisfied with it, and there would be no finalized specifications put into writing until the function was tried in the system. Only the broad system requirements contained in the original feasibility study description were to be required from the start. Besides benefits of their learning, with this approach we would also be in a much better learning position ourselves. We could hope to have an understanding of their needs before we committed ourselves to impractical goals.

The users accepted the approach pretty well. They were uncomfortable with not having any preset guidelines to measure the results by, but they did like the idea of participating in the evolution of a system that truly met their needs. Management accepted the expediency argument for beginning without complete specifications as being reasonable, if not ideal.

But, we were unable to follow through on our part. Because of the many technical problems encountered we didn't manage to develop a prototype in the time we had hoped. The system itself wasn't too difficult, but there were many untried interfaces with other systems which were necessary to connect together all of the pieces that we wanted to utilize. Unfortunately, we ran into problems

with every one of those interfaces. This caused many long delays.

Although we had been trying for early November, with the many problems, we were unable to deliver anything until March. In January, it became apparant that management was getting edgy about not having either specifications or results. SMD and Sue (who by that time had left SMD to take her new position) began to push for more written specifications to resolve the problem. The minutes of some of our meetings were reworked as the basis for more complete written documentation. During January and February the meetings between the two departments were used almost exclusively to discuss problem areas and work up written requirements for the system.

By the end of March, the users and management were more comfortable with our progress because fairly detailed system requirements had been completed. They now had a much better idea of what to hold against us if we didn't deliver a working system on time. But we still had essentially nothing working, and our programming team had no idea in many cases of how we should approach meeting those needs. The programmers were beginning to grow very uncomfortable with our progress, just as we were beginning to resolve all of the many interface problems so that we could finally begin to work on the system itself.

In this case, we successfully introduced the idea of the change. We bypassed most of the formal design /development requirements. But by failing to deliver, we jeopardized our chance of doing so again in the future. We ended up reinforcing their preference for the official system instead of the change we were attempting to introduce. Clearly, it would have been seen as a much more promising technique if we had done a better job of planning its introduction. By investigating the problems beforehand, we might have come up with a more reasonable schedule (or more reasonable promises).

We would have had trouble here if we hadn't tried to change the standard methods of system development. I don't believe that we could have completed the design work alone in six months, if we had done everything by the book. Aside from the problems inherent in fixing the specifications before providing any feedback, we simply didn't know enough when we started to write them.

The users were hampered by having a lot of experience with a poorly designed system, and very little idea of what might be done to replace it. Our SMD representative, being both new and non-technical, didn't understand the system being replaced or the technical issues involved in replacing it. And the APD staff was likewise short on understanding of the old system. While we understood what

features we could offer the user, if we had completed programming specifications without any experimentation, the resulting system probably wouldn't have worked. Any innovative system design is likely to suffer from such an initial shortage of specific knowledge and skills. But if we failed to meet our promises, it was very unlikely that we would be given a second chance at using this development technique.

Just selling the project to the user in the first place required some understanding of their resistance. They had had poor experience with the MIS staff in the past, with unfulfilled promises and unexplained problems. The existing system had technical problems that they had been told repeatedly that they would have to live with -- from lack of auditability, to difficulty of use, to operational problems. Rob West had a relatively poor understanding of data processing, but had picked up a lot of "facts" from previous MIS people that he had dealt with. MIS had never been able to solve the problems that CAD saw with the system, apparantly using technical "facts" as an excuse for anything they couldn't program. We then came in, offering to do things which seemed more difficult to the users than projects others had failed at. In addition, we offered things that they had been told explicitly couldn't be done. We clearly had to establish our believ-

ability.

At the same time, our suggestions for a new system posed a threat. CAD, particularly Rob, had helped develop the original system, and had worked hard to understand and use the system well. It was far from ideal, but by the time we arrived, they were comfortable with it.

Our proposals to replace the complicated tables that controlled the CAS processing with a greatly simplified mechanism scared them. They were used to doing things for the machine, instead of it doing things for them, and they really didn't trust us. They "understood" how the machine was performing its part now. By simplifying their interaction with the programs, we reduced their perceived control over the system, even though their actual control would be increased. But also, we were taking away their job security, saying in effect, "now anyone can use the system".

None of us at the time, I'm sure, explicitly recognized any reasons for the users' reluctance to change the old system. But we were building on trust through good performance on other projects, and met resistance with education. Our meetings served to give them exposure to many of our ideas about what could be done to improve on their system. We explained many of the technical issues as best we could in non-technical terms, and explained the value to them, as well as to us, of such things as database

systems and modular design. The meetings also helped us get to know each other, so that we were quicker to recognize when some point wasn't being fully understood or accepted. We would, whenever that happened, try to create an example for the issue in question. Then we would demonstrate how it might be handled, and relate that back to how it would work in the old CAS.

We also tried, as a general practice, to emphasize the importance of the users to the project. We couldn't write the system without incorporating their knowledge about how it should work. Likewise, they wouldn't be able to use the system effectively without fully understanding what it could do and how it would do it. The idea was that we wanted to provide them with a better tool, so that they might do their job better.

Internal Problems

Our dealings with the users were clearly far more successful than our dealings with other groups within MIS. We planned our work with the users pretty well, taking heed of the current literature on CIS implementation. But we failed to make the logical extension and worry as much about the problems of interaction with other MIS groups. Yet the problems of dealing with the user were for us, at least, far easier than those of dealing with MIS.

Once we had convinced them that a new CAS would be worthwhile, we had something our users wanted. They stood to gain from making the project successful, and obviously understood that, since they requested us to do the project. CAS itself would be their reward.

Our system, however, needed support from several MIS groups as well. We needed the cooperation of the five systems feeding CAS, and specialized support for each of the technical features, like ADABAS, that we incorporated. But there was no incentive for the MIS groups to aid us. The reward system in MIS will reward only us for the completion of our project, regardless of how many people in other groups contributed to that success. And if we fail, it is only we who fail. There was no inherent incentive for them to help. In fact, in a case like OAE, the potential threat posed by our success provided an incentive for them to help us fail.

Jerry, in OAE, whether consciously or unconsciously, seems to have found a perfect strategy for protecting his system from encroachment from other systems. By making everything look easy while we are in the planning stages, he encourages us to plan for an easy task. Later, when we are under the optimistic schedule we were deluded into setting up, he delays until we fall behind schedule and our project fails. If that delay is not enough to hurt us, we

may still fail because the actual work involved is likely to turn out to be much more than we estimated.

We were fortunate that the information we got from OAE was not required for Phase I of the CAS rewrite, but simply something we had hoped to throw in then, if possible. When Phase II is planned, we'll have a much better idea of what is involved in using the rest of the data we want from their system (and of how to deal with Jerry). Had we needed everything we went looking for, for successful completion of the initial project, we would have been even worse off. We simply hadn't planned for encountering this kind of problem.

A stock solution to this type of problem would be to make formal project requests of each group for needed assistance. This gives them the incentive to help, and covers us if they don't, since our project completion can be made contingent on the completion of all the others. But we would be left to suffer with any delays they should choose to put into their schedule, and we were hoping for speed as well as success. Rather than ensuring success, this method merely avoids blame for failure. It also doesn't solve other problems, such as our being perceived as a threat. It would take extremely good project specification to make this solution work if the other group wants us to fail, since results of the request depend on its

interpretation. No reward system is going to make them cooperate fully if they perceive us as a threat.

Despite the fact that everyone in MIS is ideally supposed to be on the same team, it isn't any more true than saying all of the departments in the firm are on the same team. The organization and the reward structure give each group its own goals that aren't truly meshed with those of other groups.

A Model for Analysis

In order to adapt the work of political and social risk analysis to the problems of CIS implementors, I would like to draw an analogy. Let us assume that each department will be a country for purposes of the model, and the MIS group setting out to implement a CIS is an international firm. The MIS department will then be the home country. In order for our firm to do business, it must have a product that someone in the other country wants (a system).

While there, they will be forced to obey the laws of that country. (Since the staff is not moving permanently, but only working there, they had better obey the home country laws as well.) To help the firm become successful, and remain around for a while, they probably don't want to simply obey the laws, but to try to fit in to the environ-

ment. They will want to pay attention to the political and social background they find themselves in, rather than just the financial and economic. They will perhaps try to mold this foreign firm to be a model citizen.

I think this is a pretty sound model for much of the interaction that an MIS group gets involved in. It extends to cover a lot of interesting points, such as the language problem. The MIS citizens all speak the same language (data processing), although perhaps different dialects, but the citizens of the host country (department) usually don't. Too often, MIS-ers forget to use the language of the host country when they are away from MIS-land on business. Many of them don't even bother to learn the language of the country they are working in, which can cause some major communication problems (even though once upon a time the languages of all these countries sprang from a common tongue).

For programmers or others in MIS to deal with accountants, they should know some accounting. They don't have have to become accountants, but they must learn the terminology well enough to communicate. They must be able to understand questions and be able to make their replies understood. And while it would be nice if everyone in the firm knew some data processing, MIS is most often the intruder, so the burden is on them to be able to explain

the necessary data processing concepts.

The analogy holds for many of the problems that we ran into in rewriting CAS. We were so busy making sure that we and our product were accepted by CAD's government and citizenry, that we forgot to worry about suppliers in our home country. The help we needed at home, of course, was technical assistance with interfaces and feeder systems.

My intent is not to push the model, but to suggest the validity of borrowing the techniques being developed for international firms, for application to CIS development efforts. Recent work in evaluation of investments in other countries has begun to make people realize that you can't limit project evaluation to economic factors alone. It has been necessary to try to find ways to incorporate social and political factors into investment decisions. Situations like the recent upheaval in Iran have pointed out the need for a better understanding of those non-economic issues. Any U.S. investment in Iran at that point was probably in an unprofitable position regardless of the economic value of the project, because of purely political/social reasons. If your company is shut down or taken over by a foreign government, it doesn't make much difference how profitable you expected it to be. (Again, the threat of expropriation corresponds to decentralization in the model -- a takeover of the facility by the foreign

government.)

Structured Method of Analysis

Political scientists, in response, have been trying to develop structured methods for looking at the non-quantifiable areas that can impact the firm. Some of the work has been based on attempts to quantify political and social variables, usually through polling expert opinion. The more interesting work, however, simply attempts to provide enough structure to force you to think about all of the important issues. The idea is not to find one ideal list of problems to avoid, but to identify how to structure your planning to be sure and consider all the issues. An individual analyst might then use that to develop a particular scheme that works well for her or himself. This will probably amount to some kind of checklist or table as a basis for an initial analysis.

With data processing projects like CAS, we begin to see the need more and more for some such political/ social analysis. Although the specific variables they are looking at are different, the approach of the political scientists is applicable to a similar analysis of a CIS implementation. As with foreign investment, until recently evaluation of CIS projects has rested almost entirely on economic cost/ benefit analysis. I don't hope to offer a complete

solution, but perhaps making use of their work will give us some insights into CIS implementation problems.

For a CIS, the analysis might be done in four parts. First, we might list the actors in each of three categories: users, suppliers, support groups. Users of the system are those who receive copies of reports or have on-line access. They might be further classified as primary (major) users (usually those who control access to the data), important secondary users, and tertiary or less important secondary users. CAS was structured in such a way that only two groups would be identified in this analysis -- the primary users being CAD, and secondary users being all of roughly equal importance to us.

We should separately identify suppliers of input to the system if they are not users. They may be divided to direct suppliers, and indirect suppliers, on the basis of whether they provide the information only to this system, or provide it to another use which then feeds this one. CAS had known problems in this area. The only direct supplier to CAS was the primary user, but the input from secondary sources was already known to be inaccurate. Validity of CAS results depends on the quality of information extracted from other systems. Improving the inputs from these suppliers could thus improve CAD's product.

For the third category, we need to list the technical

groups who will be affected. CAS needed support from feeder system support groups, from technical support groups (ADABAS, RAMIS, CICS, etc.), and from data processing operations to run online.

Next, the system should be evaluated on its own. Is it important to the organization? Is it a major drain on machine or people resources? Does its operation affect other systems? And do other systems likewise affect it? What kind of information base does it contain? Is that a large or small amount of data? Is it widely known information or a closely guarded secret? Will the system affect the way the data is distributed in the organization? Will someone have access to more information as a result of this system? Is the system simple or complicated to use? to run? to maintain? How will the new system compare on these points with whatever it replaces?

The answers to these questions establish the inherent impact of the system. Since data is a major source of power in an organization, the data content and access to it can largely determine the potential of the system to hit political obstacles. While even a small amount of relatively unimportant information can bring on problems, putting the complete financial reports of the firm into an online system with unlimited access is a sure-fire trouble maker.

Thirdly, we need to go back to the list of actors, and using the previous analysis, decide whether the system will have a positive impact, a negative impact, or no impact on each actor. Some of the results may be mixed. A system may require more effort of a department, but bring them more power in terms of additional access to data. Or access may be restricted to some due to the importance of the data now being collected. But if we can identify the major issues this way, we are a lot closer to being able to plan for them. By isolating specific potential trouble spots, it can alert the implementor before problems arise.

Finally, we should go back and look at how powerful the actor affected is, and how much he will be affected, to determine the overall issues. If actor A is likely to be unhappy, but his boss will benefit, there may be other factors acting to balance the added burden on him. If overall, actor A gains and actor B loses, perhaps we can arrange some kind of trade of benefits, with A doing something else for B. If the actor who gains isn't in a position of power over the loser, the system is probably in trouble, unless we can find some mechanism for equalizing benefits. In the case where there are just too many or too important losers, the system stands no chance. For example, if we had examined in advance our relationship with OAE, we could have recognized both the threat we

posed, and the lack of incentive that they had to help us. We might then have modified our approach to them and improved our chance of successful dealings with them.

I believe that this type of structured analysis of political and social issues, done beforehand, could prevent a lot of CIS implementation failures, because it allows you to plan around problems before they arise, giving the project the best possible chance. While CAS was not a complete failure, we would have been alerted to many of our oversights, and could probably through better planning have averted many of the delays we experienced. Many systems with little chance of success due to political and social factors within the firm might be redesigned or never attempted at all. By lowering the failure rate of systems this way, we might further improve relations with user departments and make still more gains on future projects.

Conclusions

The computer system designer/ implementor today faces many problems. They tend to center more and more around the "people" aspects of the system. Current literature points out many of the potential stumbling blocks in dealing with users, but gives the analyst little to help organize and plan around people problems.

A structured approach to analysis will help to uncover

pitfalls in advance or avoid them altogether. Published structured design methodologies cover technical design well and economic valuation to some extent. But the people issues are for the most part untouched. I propose integrating a more comprehensive structured evaluation of political and social issues into such a methodology. At an early phase in the project life cycle, the effects of the proposed system on all of the actors should be evaluated.

What we have to be particularly sensitive to, as seen with CAS, is problems at home. The literature has sensitized analysts much more to user problems than to intra-departmental problems. There is, however, a large potential for such intra-MIS conflict in an organization like Worldwide's. So we must be at least as aware of those problems as we are of the ones with users.

The structure of many data processing departments is like that at Worldwide. With the increasing amount of interdependent systems and the use of advanced software tools, this structure is becoming inadequate. It is no longer able to reward people appropiately for their work because the systems have become more complicated than the organization. A matrixed organization should probably be adopted. A task force structure for projects which cross system lines is justified on the same basis that task forces have been set up across departmental lines in the

rest of the organization. This would allow a reward structure that provides incentives for intra-group assistance.

Hopefully by doing upfront political/ social analysis, we will not only uncover problems, but be able to find ways to avoid them. Whenever possible, changes should be planned, along with any incentives needed to elicit cooperation. For some problems, simple steps such as documentation may be sufficient to greatly improve results. As with OAE, interviews may become much more satisfactory if you let it be known that the discussion is going into a memo, and will be distributed to others. The memo provides exposure -- good or bad depending on their response at the time.

The critical point is to be able to know where to expect problems, in order to be able to plan for them. Being caught off guard, especially during early development, is apt to cause a major setback to your system. The way systems in general are headed, it will increasingly be political and social issues that pose the biggest threats. A structured analysis can prepare you for an OAE or a database group that isn't willing to cooperate. Hopefully, forewarned you will be able to develop a solution in time. And more successful implementations will be the result.

APPENDIX

SOME NOTES ON THE ENVIRONMENT

At this point, I would like to try to clear up some points that seem to require further explanation. While I don't believe that the environment at Worldwide is very unusual, it perhaps fits less well into a description of a "typical" data processing shop in some specific areas than in others.

Production Systems

The process of putting a system into "production" is probably not too clear. It involves a number of things that would probably best be done by a well trained clerk instead of a programmer. It is a time consuming, detail oriented job, that must be completed for an operations clerk to be able to run the system. (In some organizations, in fact, the position of data processing "librarian" has recently begun to appear as a part of system development teams.) There are a large number of standard forms that have to be filled out. These are intended to allow the clerks in DPD to correctly plan and schedule jobs, set up inputs, verify and distribute reports, etc. They are necessarily quite detailed, including file names, report

names, and names and locations of support programmers and those who are to receive reports.

Completing the forms and getting them typed correctly is time consuming and not very challenging (mentally, at least). Because MIS typists turn over even faster than programmers, they seldom remain long enough to be very familiar with data processing terminology and formats. For instance, IBM Job Control Language instructions (JCL) must be typed exactly as written, with all capital letters and spaces in the right places. Most typists at first, being unaware that there is a difference, treat JCL like all programmer scribbles, using lowercase, spaces, and better punctuation to make it look more like English.

The result is that forms are often corrected and brought back for retyping several times before they are acceptable. After a complete set of forms is ready, it is sent to DPD for review, along with copies of successful test runs. After 3-6 weeks, DPD is required to either accept the programs to be run in production tests, or return the documentation with an explanation of why it was rejected (this is usually missing forms, mistyped JCL, or instructions that are difficult to follow).

CAS is even more of a problem than most systems. With several incoming files from other systems, its documentation must be updated every time the other systems are

changed. Program changes and changing MIS standards have regularly caused changes in descriptions, sizes of temporary files, program run times, report titles, etc. Even such things as program names changed frequently until we set up standards for the system.

Contract Programmers

The use of contract programmers at Worldwide is cited as a problem area, perhaps unfairly. Contract programmers are widely used in the industry to meet short term staffing needs. Programmers are often a scarce resource in firms, so a project will be forced to go to outside help to meet temporary staffing needs. This is normally confined to the actual programming phase of a project, where there is a large volume of work and little knowledge of the company is required.

Contract programmers in general aren't expected to have the dedication to the firm or to the specific project that a company staff member might have. Because of the short time they are assigned to a firm, they supposedly don't care about the long term viability of their problem solutions. The fact that they will probably be assigned to a project only during the development stage is said to make them indifferent as to how well the completed program will run. Their concern is thought to be only with short term

successes and meeting specific current goals. Because of their short stays, they are not expected to know or to learn the non-data processing skills necessary to write good systems (such as good accounting practices for financial systems).

But, at a firm where the "permanent" staff is no more permanent than the contractors, most of these arguments don't hold. (Contract programmers at Worldwide actually remain longer on average than MIS staff.) Contractors can often be selected from several that are available, and chosen for specific needed skills. They are generally better paid by their own firms than regular MIS staff. Therefore, better quality people are found there than through the personnel hiring process. And some of them definitely do care about their projects. They care about producing good work anyway, and want a good reputation to follow them later. Call-backs and referrals are important to their standing in the firms that sell their services.

Computer Service Chargeback

As Gibson and Nolan (1974) and others have pointed out, chargeback of computer services should normally begin very early in the life of a data processing department. Cost information is critical to the evaluation of the relative merits of any two proposed solutions to a problem.

Without cost chargeback, the organization has no fair way to decide which projects should be undertaken, and their computer systems will tend to reflect the political power of various user departments, rather than reflecting the value of particular applications to the firm.

Worldwide has managed somehow to continue without charging for computer services. They have developed a respectable set of computer applications, but no one really knows what the cost has been.

SMD attempts to judge systems by their dollar value to the company, but their analysis suffers from a lack of real cost data because there is no billing for computer services. Where other firms record usage of the computer, storage space (tapes and disks), and supplies of paper, etc., and bill the users in detail, Worldwide has chosen to ignore this possibility. Instead, they rely on the limited ability of CAS to allocate the cost of data processing to departments. MIS expenses show up as a single overhead item (some $15 million a year), allocated on the basis of some departmental workload measures in user areas.

Instead of bills, the user managers receive highly summarized reports of computer usage, giving only the total CPU usage in hours and total number of jobs run for the quarter. No attempt is made to relate cost to those figures or to charge for usage. In fact, our user Rob

West, compares his report with other managers' totals and complains when his "share" of available resources isn't used. When we were trying to convince him that rewriting CAS was necessary, one of his arguments was that we weren't using nearly as much CPU time as OAE was (at that point his figures already showed us with several hundred CPU hours for CAS for the year -- which would amount to a sizeable bill other places). He didn't feel that OAE was any more important to Worldwide than CAS. Therefore, in his opinion we deserved to use at least as much resources as they did. It didn't matter to him that they ran 30 times as many jobs as we did and produced many more reports.

This lack of awareness of the cost of computer services makes SMD's cost-benefit difficult to estimate (due to lack of figures) and still more difficult to demonstrate (what is the cost to a manager of a free service?).

Summary List of Acronyms
and Terminology

ADABAS	DBMS used for our large data files.
APD	Applications Programming Department, a part of MIS.
CAD	Cost Analysis Department, primary users of the Cost Analysis System.
CAS	Cost Analysis System, used to refer to the system and the programming group assigned to it.
CICS	On-line terminal monitor program available on user terminals.
CIS	Computerized Information System.
COBOL	Computer programming language used in extracts.
DBMS	Data Base Management System.
DPD	Data Processing Department, a part of MIS.
E-R	Entity-Relationship method for database design.
Extracts	The front end of CAS, programs collect and convert feeder system data for allocation.
IBM	Manufacturer of most computer hardware at Worldwide, also provides much of the systems software.
MIS	Management Information Systems, a division of Worldwide Insurance.
OAE	A CIS and the support group for it in APD. Provides three input files to CAS.
RAMIS	DBMS used by CAS for smaller data files.
PL/1	Programming language used in the new CAS.
SMD	Systems Management Department, a part of MIS.
Technology	Systems Technology Department, a part of MIS.
TSO	On-line terminal monitor program available on programmer terminals.

Figure 8

BIBLIOGRAPHY

Chen, Peter; "The Entity-Relationship Approach to Logical Data Base Design"; Wellesley, Mass.: Q.E.D. Information Sciences, Inc., 1977.

Gane, Chris, and Trish Sarson; Structured Systems Analysis: Tools and Techniques; New York: Improved Systems Technologies, Inc., 1977.

Gibson, C. F., and R. L. Nolan; "Managing the Four Stages of EDP Growth", Harvard Business Review; Jan-Feb. 1974.

Jackson, Michael; Principles of Program Design; Academic Press, 1975.

Kobrin, Stephen J.; "When Does Political Instability Result in Increased Investment Risk? ", Columbia Journal of World Business; Fall 1978.

Kraar, Louis; "The Multinationals Get Smarter About Political Risks", Fortune; March 24, 1980.

Markus, M. Lynne; "Understanding Information System Use in Organizations: A Theoretical Explanation"; Unpublished doctoral dissertation, Case Western Reserve University, 1979.

Pettigrew, Andrew M.; The Politics of Organizational Decision-Making; London: Tavistock, 1973.

Yourdon, E., and L. L. Constantine; Structured Design; Yourdon, Inc., 1978.

POSTSCRIPT

On May 5, a memo was released in MIS announcing a reorganization of the department. Among other changes, SMD was officially resurrected, but as a part of each APD group. Existing SMD representatives were reassigned to the APD managers that they supported (an effective downgrading of their status, since formerly they were considered equals of those group managers.) At the same time, we heard unofficially of a "minor" corporate restructuring to take place soon.

On May 6, Rob West announced that he was leaving for a new job. By the time he had left, Anne had given notice as well. Dick, also on Rob's staff, transferred to the Financial Systems group that Sue was in. He was to support some of Rob's work from that area. But, after less than a week, he too announced his intention to leave Worldwide.

At the end of May, our original contacts in the RAMIS and ADABAS support groups left within a week of each other. Andy moved to Technology the first week in June. Brad and Stan are both actively looking for jobs. I left Worldwide at the end of June, on the planned completion date of CAS.

The project to develop a consistent corporate-wide numbering scheme is still "being considered". Chances are that it will be "considered" until everyone involved has left or lost interest.

I expect that CAS will be completed by the end of the summer. This is somewhat behind schedule, but CAS has already proven enough to the firm that they will proceed with our long range plans. An advanced development group, headed by Lou, has been set up to begin plans for conversion of the next financial system. CAS has essentially no users at this point, but it has the support of top management, so it is expected that Rob's replacement will have an interest in making use of the system.