

**A COMPUTER-BASED APPROACH FOR RESOLVING BUDGET
DISCREPANCY**

by

FAN CHUN

B.Com. (Hon.), Memorial University of Newfoundland, 1994

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE**

in

THE FACULTY OF GRADUATE STUDIES

(Faculty of Commerce and Business Administration)

We accept this thesis as conforming to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

APRIL 1997

© Fan Chun, 1997

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Faculty
Department of Commerce

The University of British Columbia
Vancouver, Canada

Date Apr 28 97

ABSTRACT

Information technologies (IT) have drastically transformed many business activities in the past 30 years. Among the numerous business activities in organizations, the budget preparation process plays a very important role. Although there are many developed systems available to automate certain aspects of the budgeting process, there is much more that can be done. This thesis studies the budgeting process and certain behavioral factors involved in the preparation of the budget. Knowing the complications inherent in budgeting, effort has been directed towards automating the mechanistic portion of the process; narrowing down the differences; and identifying the sources of discrepancy by using a computer-based system.

Based on an algorithm derived from a manually solved budgeting case, the requirements of automation were studied, a computer based system was developed and tested. In addition, directions for future studies are suggested in this thesis.

TABLE OF CONTENTS

ABSTRACT.....	ii
TABLE OF CONTENTS	iii
LIST OF TABLES.....	v
LIST OF FIGURES.....	vi
ACKNOWLEDGMENT	vii
INTRODUCTION	1
1 MOTIVATION OF THIS PROJECT	1
2 CHAPTER SUMMARY	3
DIFFICULTIES AND CHALLENGES IN THE BUDGETING	5
1 THE BUDGET PREPARATION PROCESS	6
1.1 Organization.....	6
1.2 Issuance of guidelines	7
1.3 Initial budget proposal	7
1.4 Negotiation	8
1.5 Review and Approval.....	8
2 BUDGET NEGOTIATION - A CLOSER LOOK	9
2.1 Incentive to Build Slack.....	9
2.2 Asymmetric Distribution of Information	10
2.3 Unwillingness to Disclose and Share Information.....	11
2.4 Other Difficulties.....	12
2.5 Conclusion	13
CASE STUDY - NATIONAL MOTORS INC.	15
1 CASE SUMMARY	16
1.1 Organizational structure before consolidation.....	16
1.2 Organizational structure after consolidation.....	17
1.3 Original budget after consolidation.....	17
1.4 Manufacturing Office: Supplemental budget proposal.....	18
1.5 Controller's Office: Supplemental budget proposal.....	19
1.6 Protest from the Manufacturing Office	19
1.7 Discrepancies to be reconciled	21
2 THE DIFFICULTIES OF THE CASE.....	21
2.1 Incentive to Build Slack.....	21
2.2 Asymmetric Distribution of Information	22
2.3 Unwillingness to Disclose and Share Information versus Information Overload	22
3 THE REAL ISSUE OF THE CASE.....	23
4 RECONCILIATION ALGORITHM	23
5 A SIMPLIFIED SOLUTION OF THE CASE	26
AUTOMATION - CURRENTLY AVAILABLE TECHNOLOGIES	28

1 POTENTIAL FOR AUTOMATION	28
2 SPREADSHEET	29
3 RELATIONAL DATABASE MANAGEMENT SYSTEMS (RDBMS)	30
4 OBJECT-ORIENTED DATABASE MANAGEMENT SYSTEM (OODBMS).....	31
5 ACCOUNTING SOFTWARE	32
6 SOFTWARE AVAILABLE SUPPORTING NEGOTIATION	32
IMPLEMENTATION	34
1 ALGORITHM FOR IMPLEMENTATION.....	35
1.1 Assumptions Used in the Algorithm	37
1.2 Initial Checking	37
1.3 Further Analysis - Departmental Accounts.....	39
1.4 Further Analysis - Categorical Sub-accounts	41
1.5 Further Analysis - Equation	42
1.6 Iterative Approach.....	43
2 STRUCTURAL DESIGN OF THE SYSTEM.....	43
2.1 Main Reasoning Engine	44
2.2 The Equation Simplification Engine.....	44
2.3 The Account Database.....	45
3 KNOWLEDGE REPRESENTATION	46
3.1 Basic Components of an Account	46
3.2 Account Key.....	47
3.3 Account Number	48
4 TESTING.....	49
4.1 The Screen Capture	51
4.2 Log	55
4.3 Personnel Discrepancy Reconciliation	56
CONCLUSION AND FUTURE WORK.....	57
1 LIMITATIONS OF THE STUDY.....	59
2 FUTURE WORK.....	59
3 CONCLUSION.....	60
BIBLIOGRAPHY.....	61
APPENDIX A.....	63
APPENDIX B.....	74
APPENDIX C.....	75
1 MAIN REASONING ENGINE.....	75
2 EQUATION SIMPLIFICATION ENGINE.....	83
3 DIVISIONAL SUBTOTAL	100
4 CORPORATE HIERARCHY.....	100
5 CONTROLLER'S DATABASE	101
6 MANUFACTURING OFFICE'S DATABASE	103
APPENDIX D.....	106
1 TEST RESULT	106

LIST OF TABLES

TABLE 1. THE ORIGINAL COMBINED BUDGET.....	18
TABLE 2. MANUFACTURING OFFICE: SUPPLEMENTAL BUDGET PROPOSAL.....	18
TABLE 3. CONTROLLER'S OFFICE SUPPLEMENTAL BUDGET PROPOSAL.....	19
TABLE 4. SUMMARY OF THE CASE	20
TABLE 5. DISCREPANCIES TO BE RESOLVED.....	21
TABLE 6. SOURCES OF DISCREPANCIES IN DOLLAR AMOUNT	27
TABLE 7. DISCREPANCY CAUSED BY ONE-TIME COSTS.....	27
TABLE 8. DISCREPANCY CAUSED BY SALARY MIX CHANGE.....	27
TABLE 9. EXAMPLES OF KEYWORDS USED IN THE NATIONAL MOTORS CASE.....	48
TABLE 10. HIERARCHICAL ORDER OF ACCOUNT NUMBERS.....	49

LIST OF FIGURES

FIGURE 1. BUDGET PREPARATION PROCESS	6
FIGURE 2. ORGANIZATIONAL CHART OF NATIONAL MOTORS BEFORE CONSOLIDATION (PARTIAL).....	16
FIGURE 3. ORGANIZATIONAL CHART OF NATIONAL MOTORS AFTER CONSOLIDATION (PARTIAL)	17
FIGURE 4. LOGICAL FLOW OF ACTIVITIES.....	25
FIGURE 5. LOGICAL DESIGN OF SYSTEM.....	36
FIGURE 6. FURTHER ANALYSIS - CHECKING DIVISIONAL SUB-ACCOUNTS.....	39
FIGURE 7. FURTHER ANALYSIS - CATEGORICAL SUB-ACCOUNTS.....	41
FIGURE 8. FURTHER ANALYSIS - EQUATION.....	42
FIGURE 9. SYSTEM STRUCTURE.....	44
FIGURE 10. RELATIONSHIPS BETWEEN TESTING SYSTEMS	50
FIGURE 11. CONTROLLER'S INITIAL RESPONSE.....	51
FIGURE 12. MANUFACTURER UNABLE TO PROVIDE INFORMATION.....	52
FIGURE 13. QUERY SENT AFTER SECOND TEST.....	53
FIGURE 14. MANUFACTURING OFFICE'S REPLY	53
FIGURE 15. A NEW ROUND OF EXAMINATIONS.....	54

ACKNOWLEDGMENT

I am indebted to many people for their assistance with this thesis.

First I would like to thank my supervisor, Dr. Carson Woo, for his continuous guidance and invaluable support throughout every phase of this thesis. I would also like to thank Dr. Al Dexter and Dr. Mitchell Farlee for reading through this thesis and providing me with constructive comments.

Special thanks to Dr. Sunil Dutta and Dr. Vasu Krishnamurthy who provided me guidance for the accounting part of this thesis.

Special thanks to Mr. Lin ZhuHai at South Bank University for allowing me to include his equation simplification which is an essential part of my final system.

I am also grateful to those people who have provided me help in writing this thesis: Ms. Gina Lee, Mr. Darrell Jung and Mr. Ken Leung.

There are those who continuously supported me. I cannot express my thankfulness in any words. I can only thank God for having them around: mom, sis, and my lovely wife, Mei.

Finally, I thank our heavenly father for His provision.

Chapter One

INTRODUCTION

1 Motivation of This Project

Information technologies (IT) have automated and transformed many business activities drastically in the past 30 years. So far much effort has been devoted to developing tools such as relational databases and spreadsheets that can be employed to support various business activities. We have seen many successful examples of how technologies and tools dramatically improved performance and speed. These tools, in most cases, not only improved but also revolutionized the way business activities are conducted in organizations.

Among the numerous business activities in organizations, the budget preparation process plays a very important role because “budgets are an important tool for effective short-term planning and control in organizations” (Anthony, Dearden, and Govindarajan,

1992, p. 436). Typically, organizations and their subsidiaries make estimates of future revenues to be realized over a certain time period and plan their expenses accordingly. Traditionally, budget preparation is conducted on an annual basis. With the aid of modern IT tools, budget preparations now can also be done on a semi-annual, quarterly or even monthly basis. Although there are systems, such as SAP¹, already developed to automate certain aspects of the budgeting process, much more can be done to research and develop appropriate and specific tools.

The authors of the book "Management Control Systems" point out that budget preparation has four principal purposes and they are described in the following section:

(1) to fine tune the strategic plan; (2) to help coordinate the activities of the several parts of the organization; (3) to assign responsibility to managers, to authorize the amounts they are permitted to spend, and to inform them of the performance that is expected of them; and (4) to obtain a commitment that is a basis for evaluating a manager's actual performance. (Anthony et al., 1992, p. 438)

The final products of the budgetary process, the prepared budgets, represent the organizational consensus about future operating goals and how each department is designed to work towards achieving the overall goals of the company. They act as an internal communication vehicle linking the different departments with each other and with upper management. Finally, budgets serve as standards, control devices and motivation tools that influence employees to act in ways that are consistent with effective and efficient operations

¹ Information on the SAP system can be found at the company's website: <http://www.sap.com>

and in congruence with organizational goals (Siegel, Ramanauskas-Marconi, 1989, pp. 125-126).

Judging from the important nature of budget preparation and the use of budgets, one can safely conclude that *timeliness* becomes an essential element in terms of delivering the budget. Therefore, it is worth devoting time and resources to study the possibility of automating the budgeting process.

The objective of this thesis is to study the possibility of automating the budgeting process by employing information technology.

2 Chapter Summary

The topic was dealt with first by studying the specific procedures and elements involved in budget preparation and approval. Difficulties and challenges involved in preparing the budgets were studied and discussed in the chapter immediately following this one. They were identified as the incentives in understating revenues and overstating expenses, unwillingness to share information, and the responsibility issue. All these stated issues play a role in how information is presented and handled by the different parties involved in the budget preparation (Chapter Two).

Much of this research work was spent in selecting, analyzing and solving a budget preparation case prepared by J. Hekimian under the supervision of R. N. Anthony of Harvard Business School (Anthony et al., 1992, pp. 456 - 467) which is summarized in Chapter

Three. An algorithm was generated after in-depth manual analysis of the budgeting case. The algorithm is presented in the format of a flowchart (Chapter Three).

Next the popular IT tools currently employed in assisting in budget preparation were researched and summarized in Chapter Four of this paper. These tools did bring in a certain degree of automation to the budget preparation process but they did not address all the automation issues adequately. Any non-addressed issues were subjected to further study in order to identify the potential for automation. No specific tools were available to help negotiation parties pinpoint budget discrepancies.

Due to the inadequacies of the current available tools, a system was designed and implemented in PROLOG language based on the algorithm stated in Chapter Three. All design and implementation details and experiences are documented in Chapter Five, which serve as the ground work for future studies.

Since budget preparation is such a broad topic, we devoted our research efforts to narrowing the scope of differences and pinpointing the discrepancies for the budget preparation parties so that they can focus on the real "agendas" on the bargaining table.

Chapter Six outlines issues that are not addressed by this study. This chapter also contains suggestions and opinions for future research.

Chapter Two

DIFFICULTIES AND CHALLENGES IN THE BUDGETING

Budget preparation is an important business process which serves the purposes of fine-tuning corporate strategic plans, coordinating the activities of different departments, assigning responsibility to managers, and obtaining a commitment (Anthony, Dearden, and Govindarajan, 1992). The administration of such an important process that has a high impact on the sustainability of an organization often involves grave delicacy. Therefore, it is appropriate for us to look at the process of budget preparation. The following sections describe this business process, and certain human factors that complicate the process will be reviewed. The last section of this chapter will identify the focus of automation.

1 The Budget Preparation Process

Though there are no standard rules for corporations to follow, the budgetary process normally involves the following steps described in Management Control System (Figure 1): organization, issuance of guidelines, initial budget proposal, negotiation, and review and approval (Anthony et al., 1992, pp. 445-449).

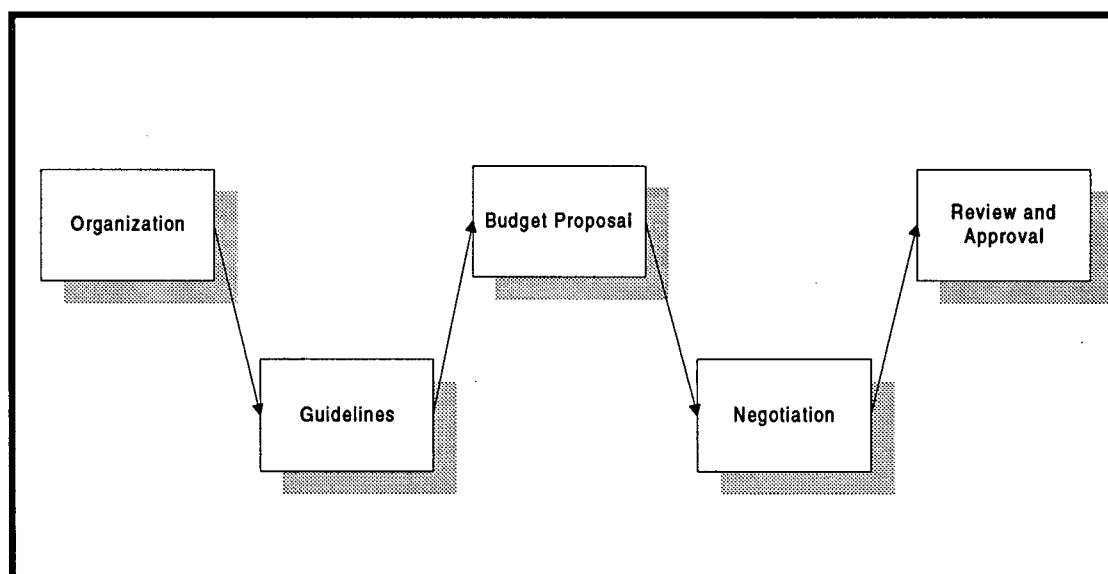


Figure 1. Budget Preparation Process

1.1 Organization

A budgetary control system is usually administered by the budget department which publishes procedures and defines assumptions as the basis for budget preparation. The budget department coordinates communication between interrelated departments for better sharing of information. During the budget preparation process, the department analyzes the proposed budgets and makes recommendation to the **budget committee**. The department

also monitors the execution of the budget by analyzing the reported performance throughout the operating period.

“The budget committee consists of members of senior management ... and the financial vice president” (Anthony et al., 1992, p. 446). The committee plays a key role in budget preparation. It reviews, rejects or approves the final budget. In addition, any revisions deemed necessary during the year must also be reviewed and approved by the committee.

1.2 Issuance of guidelines

This is usually the first step in the budget preparation process. The budget department develops guidelines that govern the preparation of the budget. The lower-level managers are expected to follow these guidelines when preparing budgets. In some cases, lower level management are consulted before guidelines are approved.

1.3 Initial budget proposal

Based on the guidelines issued by the budget committee, the responsibility center managers compile a budget request. The formation of the budget has to go through the following steps: 1. Estimation based on current period data. 2. Adjustment for changes in external factors. 3. Adjustments for changes in internal policies and practices.

1.4 Negotiation

The negotiation stage is described as the “heart of the [budgeting] process” by Anthony et. al. The following paragraph describes a typical scenario of budget negotiation:

the budgetee discusses the proposed budget with his or her superior. The superior attempts to judge the validity of each of the adjustments ... The superior recognizes that he or she will become the budgetee at the next level of the budget process, and therefore, must be prepared to defend the budget that is finally agreed to.

(Anthony et al., 1992)

1.5 Review and Approval

As the budget proposals travel up the corporate hierarchy, they are put together for analysis. If the big picture provides unsatisfactory results, the proposals would be sent back for reworking.

There are numerous tools available to automate the budgeting process. However, there are no specific tools available to automate the negotiation stage of the budgeting process. In addition, negotiation is likely to be intensified by the growing trend of globalization under the current economy environment since more variables are to be considered in a decentralized, international environment. Therefore, it is worthwhile to direct our efforts to studying the budget negotiation process.

2 Budget Negotiation - A closer look

Business budgets are similar to personal budgets in that they both serve the comparable purposes of “(1) making estimates of income, (2) planning expenditures, and (3) restricting spending in accordance with the plan” (Garrison, Chesley, and Carroll, 1990, p. 411). However, business budgets differ from personal budgets in terms of their size and scope. The former involves activities which are larger in size and more detailed in scope.

The second significant difference between business and personal budgets is the presence of extensive *negotiation* between the divisions and corporate headquarters during the preparation of business budgets.

There are certain behavioral factors that complicate the budget negotiation process such as the incentive of building slack, asymmetric distribution of information, and unwillingness to share information. We will review these factors in this section because they are relevant to this thesis.

2.1 Incentive to Build Slack

Budget processes can be initiated either “top down” or “bottom up”. Both of these approaches involve certain degree of participation. The participation gives line managers the power to establish the content of their budgets which would eventually be used as an operational yardstick to evaluate the performance of their divisions. Knowing this fact, managers tend to build “slack” into their budgets by “underestimating revenues,

overestimating costs, or overstating the amount of inputs necessary to manufacture a unit of output”(Siegel, Ramanauskas-Marconi, 1989, p. 140).

Cyert and March (1963, pp. 36-38) defined slack as the difference between “the total resources available to the firm and the total necessary to maintain the organization coalition” (Ezzamel, Hart, 1987, pp. 358). Therefore, “slack is the difference between the resources that are actually necessary to efficiently complete a task and the larger amount of resources that are earmarked for the task” (Siegel et al., 1989, p.140).

Budgetary slack represents a degree of padding introduced into budgets so as to guard against possible failure to reach targets (Arnold, Hope, 1983, p. 290). Essentially, by doing so, the budgetees are making the budget an easier target for them to achieve. Since the 1960's, numerous studies have been constructed to investigate this topic. It became apparent that slack exists within even the most efficient and tightly controlled organizations. Some researchers argue that the existence of slack is unavoidable “because human nature requires it to exist” (Arnold et al, 1983, p. 290). In addition, it was suggested that if budgetees are given no flexibility in managing their budgetary constraints, “conflict will quickly arise between the individual's personal goals and those of the firm” (Ibid.).

2.2 Asymmetric Distribution of Information

Building slack is made possible because of the asymmetric distribution of information in corporations. This phenomenon is particularly true in a budgeting situation considering the budget is prepared and submitted by budgetee regarding their own situation and predictions. Since managers possess more accurate and up-to-date information (either

quantitative or qualitative) on their own divisions when comparing to headquarters, this imbalance in information gives them the ability to legitimately manipulate data in order to build slack. For example, let's consider the case of an international company with headquarters located in North America and one of its divisions operating in Asia. Compared to headquarters, the divisional manager in Asia will always likely have more up-to-date information on local economic conditions, government regulation changes and labor movement trends. The divisional manager, upon receiving the information, can decide to react, and communicate this information to headquarters in a manner that benefits the local division. This level of autonomy and flexibility can only be carried out under the premise of not violating the internal control policies of the corporation.

2.3 Unwillingness to Disclose and Share Information

Managers are not motivated to share and fully disclose information under the current participatory budgeting practice. "Argyris noted that emphasis on budget attainment resulted in many employees being task-centered and many supervisors being department-centered" (Ezzamel, Hart, 1987. p. 352). Thus, the relationship between departments is ignored and focus is placed on individual departmental success. In order to excel over their organizational counterparts and divisional rivalries, managers will not likely share information with each other.

In addition, participatory budgeting encourages the manipulation of data and the selective disclosure of information, as proven in the case of building slack into budgets. Thus, managers are unwilling to make known what seems to be their "own" portion of

information since the disclosure of this information, regarded as the private property of the divisional manager, will likely leave them in a “vulnerable” position when challenged and asked to justify the validity of their budgeted figures.

2.4 Other Difficulties

Researchers advised us that, in the administering of the budget program, management should not use it as a pressurizing tool to force employees to unfairly assume responsibility for certain problems (Carruth, McClendon, and Ballard, 1983). However, the very same study also told us that budgeting is often used as a pressure device and great emphasis is placed on meeting the budget under all circumstances. Negative emotions, hostility, tension, and mistrust rather than greater cooperation and productivity often accompany the budgeting process (Garrison et al., 1990, p. 417).

“Similarly, Hughes (1965) described the endless cycle of the conventional budgetary control process” (Ezzamel, Hart, 1987, p.352). In Hughes’ analysis, the endless cycle of unresolved conflict is caused by the failure of top management and lower management to recognize one another’s needs. In an organization, he concluded that, lower management needs flexibility and top management needs control. “Thus, top management pursues its need for control by placing greater emphasis on the use of budgets and rules ... Lower management pursues its quest for flexibility by general avoidance of controls and rules, particularly budgets” (Ezzamel, Hart, 1987, p. 352). The more lower management violates the rules, the more top management sees itself losing control. As a result, more rules are

introduced to the organization in hope of regaining control. Therefore, lower management violates these rules to gain flexibility, and so the cycle repeats itself.

Although, strictly speaking, the issues identified in this section are not related to this thesis directly, they serve as background information to illustrate the degree of complexity of the budgeting process.

2.5 Conclusion

Among all the above discussed difficulties and challenges encountered in the budgeting process, it is suggested that budgeting is a complicated process, thus making automation more difficult.

Since human participation in the budgeting process is essential, it would be neither practical nor efficient to try to automate and eliminate human involvement in every stage of the budgeting process. Effort should be diverted to design a set of protocol and a system to assist human participants in communicating and preparing budgets. "Thus, by automating certain ... activities, human involvement is limited to those aspects that cannot be automated" (Chang & Woo, 1994). Specifically, effort should be focused on automating the information clarification aspect of the budgetary process and this will be discussed throughout the remainder of this paper. In other words, one cannot completely eliminate the human factor in negotiation. Automation can only be applied in the mechanistic steps of the budget negotiation, the narrowing down of differences by exchanging information so that the budgetees and budgetors can reach a common ground for negotiation.

Based on the identified difficulties and challenges of the budget process, and the knowledge of behavioral aspects of the participants, it should be a safe strategy to automate to the extent that encourages the parties to share a sufficient amount of information. In other words, automation should allow the parties to share and exchange only those data necessary in solving the problems and leave the rest alone. Also, with automation in place, budgeting parties should also be guaranteed autonomy. It would be impractical to expect the divisional managers to surrender all their information. Automation can be viewed as a vehicle to strip away authority.

Chapter Three

CASE STUDY - NATIONAL MOTORS INC.

The National Motors Inc. Case (Anthony, Dearden, and Govindarajan, 1992, pp. 456-467), prepared by J. Hekimian under the supervision of R. N. Anthony, serves as an excellent example to illustrate the typical budget preparation process with concentration on the information clarification aspect. The case describes a typical budgeting situation involving two parties, a division controller as the budgetor, and the manufacturer's office as the budgetee. The process is initiated by the manufacturer's office which submitted a supplemental budget proposal. Each of these parties then presented their own reasoning by exchanging data and clarifying information. Finally, after all information was presented, the controller was left with several alternatives to proceed to the next step, which is either to

concur with the supplemental budget request or to continue his position. The National Motors Inc.² Case is summarized as follows:

1 Case Summary

1.1 Organizational structure before consolidation

National Motors exercised strong budgetary control. Proposed budget and supplemental budgets were to be analyzed, revised, consolidated, reviewed and approved first within a division and then at the corporate level.

Panther and Starling were two separate division of the National Motors Inc. Each of them had its own manufacturing activities and budgets. The partial organizational structure of National Motors is outlined in the following figure (Figure 2).

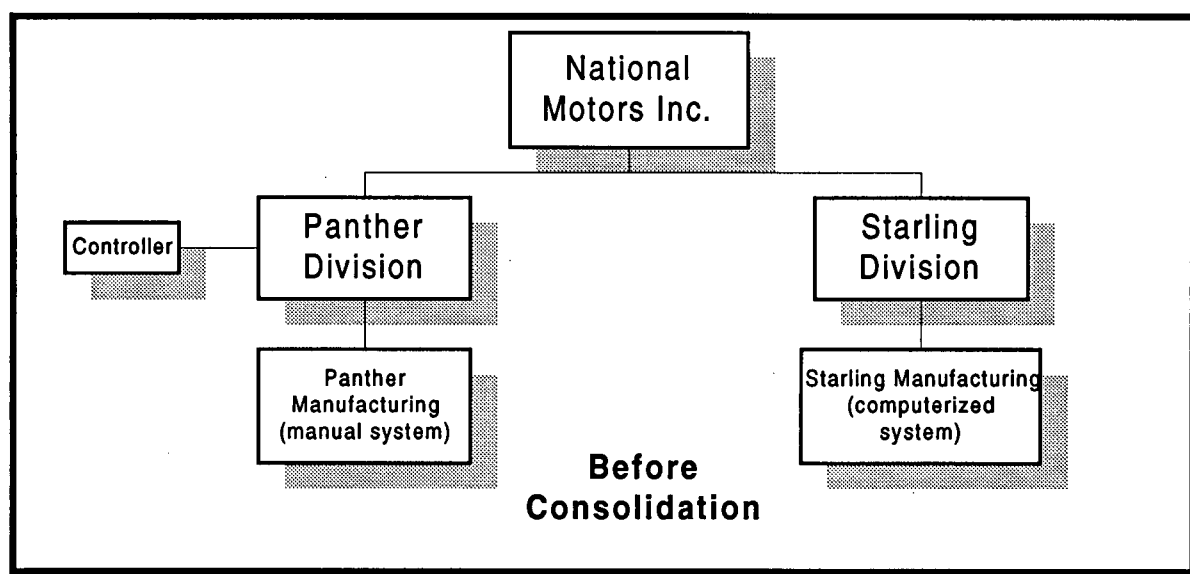


Figure 2. Organizational Chart of National Motors Before Consolidation (Partial)

² A copy of the National Motors Inc. case is included in the appendix of this thesis.

1.2 Organizational structure after consolidation

During the last quarter of 1982, Panther Automobile Division had absorbed the manufacturing activities of the Starling Automobile Division (Figure 3).

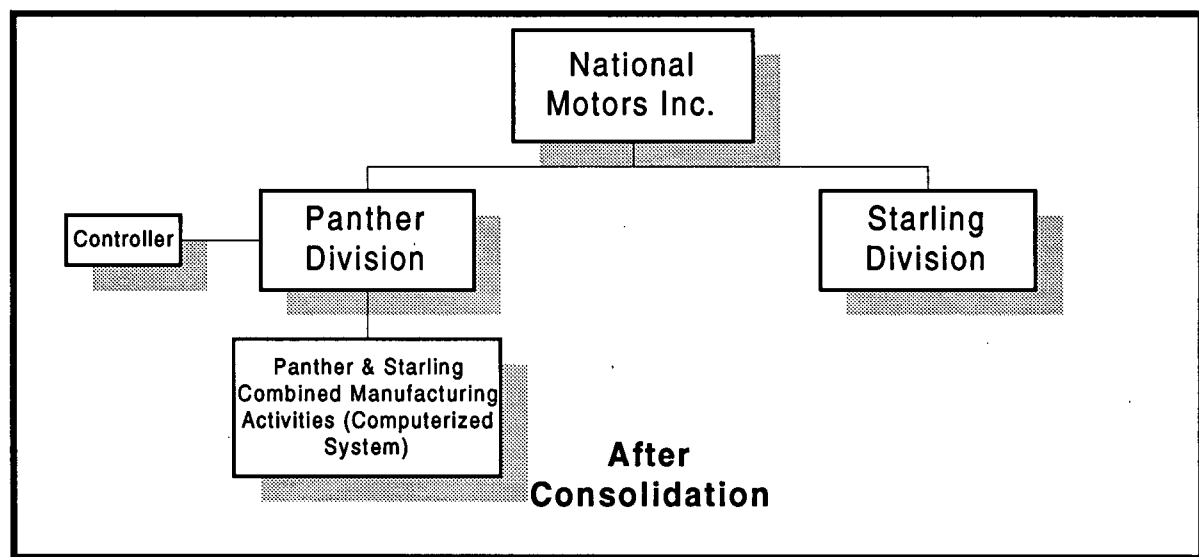


Figure 3. Organizational Chart of National Motors After Consolidation (Partial)

1.3 Original budget after consolidation

The approved budgets for 1983 for these two departments had already been submitted and approved separately prior to consolidation. Therefore, there were two separate budgets for the two manufacturing departments. In addition, the manufacturing activities of Panther division was making the transition from a manual system to a fully computerized system, similar to what Starling had already been practicing for a period of time.

The budgets of the two manufacturing offices were combined upon consolidation and treated as the revised budget for Panther's manufacturing office (Table 1). The

	Panther			Starling		
	Personnel	\$'000	Personnel	\$'000	Personnel	\$'000
Budget before Consolidation	76	\$1,444	55	\$1,336	131	\$2,780
Savings from consolidation					(24)	(368)
Savings from computerization					(23)	(276)
					<u>84</u>	<u>\$2,136</u>

Table 1. The Original Combined Budget

combined budget was created by combining the original budgets of the two divisions with adjustment for savings from consolidation and savings from computerization.

1.4 Manufacturing Office: Supplemental budget proposal

Due to a change in the activity level, in April 1983, the new consolidated manufacturing office submitted a supplemental budget proposal (Table 2) to request an increase in personnel levels in order

Personnel	Dollar Amount
109	\$2,916,000

Table 2. Manufacturing Office: Supplemental Budget Proposal

to handle the combined work load. The request was supported by detailed breakdowns of each department.

1.5 Controller's Office: Supplemental budget proposal

However, the controller's office did not concur with the manufacturing office's proposal. It disagreed with the manufacturing office's interpretation of the numbers. Therefore, it proposed its

Personnel	Dollar Amount
84	\$2,256,000

Table 3. Controller's Office Supplemental Budget Proposal

own version of the revised budget (Table 3) which was also supported by detailed calculation and breakdowns. This version of the budget proposal had a slight increase in dollar figures which narrowed the dollar discrepancy between the two parties. The controller insisted on the same staff levels.

1.6 Protest from the Manufacturing Office

The manufacturing office did not accept the controller's proposal. They attacked the invalidity of the controller's proposal and presented more supporting data to justify their original supplemental budget proposal. A summary of the case is included in the following table (Table 4).

	Manufacturing Office	Controller's Office
Stage 1	ACTION: <ul style="list-style-type: none"> Presented manufacturing's supplemental budget proposal. ARGUMENTS: <ol style="list-style-type: none"> There was a direct relationship between number of parts and personnel levels The increase in parts to be handled caused the need to increase personnel levels Therefore, the increase in budget was justifiable 	N/A
Stage 2	N/A	ACTION: <ul style="list-style-type: none"> Declined the manufacturing office's proposal. presented controller's supplemental budget proposal. ARGUMENTS: <ol style="list-style-type: none"> Computerization should bring forth major improvements when compared to a manual system. The manufacturing office had committed to saving personnel, therefore the revised budget should maintain this commitment.
Stage 3	ACTION: <ul style="list-style-type: none"> Presented more reasons and additional supporting data. ARGUMENTS: <ol style="list-style-type: none"> Although a computerized system should not cost more than a manual system, the increase in budget they requested was justifiable due to the following reasons: <ul style="list-style-type: none"> Work-load increase which was more than the estimated figure used by controller's office. Increase in Salary Mix Non-recurring cost penalties The computerization plan would bring in future savings which was not currently taken into consideration. 	N/A

Table 4. Summary of the Case

1.7 Discrepancies to be reconciled

Finally, after three rounds of information exchange, the two parties were left with discrepancies in number of personnel and dollar amount of budget to be resolved (Table 5).

	<u>Personnel</u>	<u>Amount (000s)</u>
Manufacturing Office	109	\$ 2,916
Controller's Office	84	2,256
Discrepancy	<u>25</u>	<u>\$ 660</u>

Table 5. Discrepancies to be Resolved

2 The Difficulties of the Case

The National Motors Inc. is such a complicated case that it took us a relatively lengthy period of time to comprehend and to solve the case manually. The complexity of the case is rooted in the difficulties of the budgeting process identified in Chapter 2 of this thesis; namely; incentive to build slack, asymmetric distribution of information, and unwillingness to disclose data. In addition, the abundant information presented in the case, whether they were relevant or irrelevant to the real issues, also contributed to the case's complexity.

2.1 Incentive to Build Slack

Based on the limited amount of information given in the case, we do not know whether the manufacturing office had in fact an incentive to build slack into their supplemental budget proposal, however, the controller's office perceived the situation in this fashion. The controller's responded to the manufacturing office by making the comment that "the manufacturing office had committed itself to a saving ..., whereas the current proposal was

... over the levels committed" (Anthony et al, 1992, p.462). This presupposition caused the controller's office to partially dismiss the manufacturing office's proposal and to present their own supplemental budget proposal.

2.2 Asymmetric Distribution of Information

Being closer to the source of information, in this case, the manufacturing office had more updated data on the average salary mix and the activity level after reorganization. Therefore there were essentially two versions of data residing in National Motors Inc. internally. One set of data was owned by the controller's office and the other, more updated version, by the manufacturing office. This difference caused two different sets of beliefs to arise within the controller's office and the manufacturing office.

2.3 Unwillingness to Disclose and Share Information versus Information Overload

In this case both parties demonstrated their willingness to share information as needed. However, the very same willingness to disclose information caused another problem - information overload. It was observed that, very often, a massive amount of additional information was provided to explain a point. This information overload would only cause the other party to ignore the piece of information entirely. For example, at the beginning of the case, the detailed calculations provided by the manufacturing office to support their request to increase personnel were largely ignored by the controller's office. On the other hand, the details of the controller's recommended budget which presented more detailed support to justify its supplemental budget proposal was also dismissed by the manufacturing office.

3 The *REAL* Issue of the Case

The manufacturing and controller's office focused their attention on the differences in their total budgeted figures and the total number of personnel. However, the scope of these differences was so large that the two parties did not even have common ground for negotiation. In addition, the two parties jumped on budget negotiation so prematurely that they failed to recognize the real agenda of their negotiation. Unless the asymmetric distribution of information was clarified and the sources of discrepancies pinpointed by tracing the detailed composition of the overall figures, no real negotiation could begin.

4 Reconciliation Algorithm

The logic for reconciling the differences in the budget approval process were studied. By studying the process, the mechanistic steps involving in the process of narrowing down the differences of the two parties during the budget negotiation can be identified since the mechanistic steps are the best candidates for computer automation. In other words, studying the mechanism of the mechanistic portion of the budget negotiation cycle helps to identify the activities that have the potential to be automated.

An algorithm of the process and logic are summarized in the following flowchart. The algorithm includes such functions as comparing the request against known data, raising intelligent questions to request supporting data, and marking the request with the status of "approved" or "source of discrepancy". The result of the algorithm, will essentially help to identify a list of discrepancies which can be used a basis for the two sides involved to start

their negotiations. The ultimate goal of automation should support the activities outlined in the flowchart (Figure 4).

The algorithm was constructed based on the view of the controller's office which started by comparing the total budget figures and personnel levels submitted by the manufacturing office against his/her own version of the numbers. If the numbers fell in a reasonable range, the budget would be approved and no further checking would be needed. However, if the submitted numbers did not match the controller's own version or exceeded the preset limit, a request would be sent to the manufacturing for more information. If there was no additional information available for a number, it would be marked as a source of discrepancy. If additional information were supplied, the algorithm would be repeated to check the validity of these subsequent supplied data.

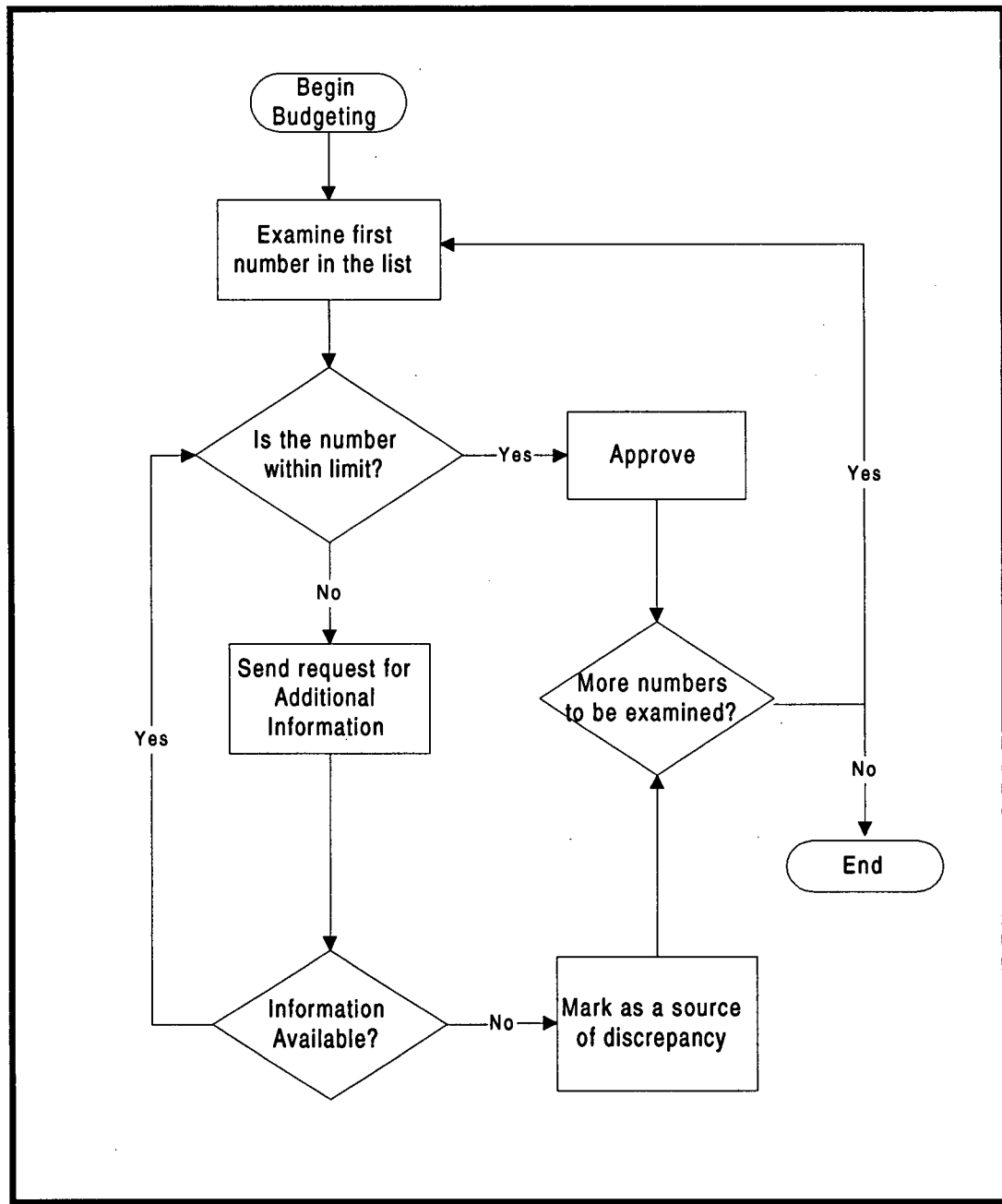


Figure 4. Logical Flow of Activities

5 A Simplified Solution of the Case

In order for the two parties to have a common basis for negotiation, the causes of discrepancies must first be identified. Only after the differences were reconciled, the factors accounting for differences identified, and the dollar impact of each factor understood, could the two parties negotiate on the real agenda.

The solution seems trivial and straightforward enough that no automation is needed. However one should bear in mind that this sense of straightforwardness is achieved after lengthy analysis and gathering of relevant data from the case. Though the manual process is time-consuming and requires a high level of sophistication in accounting knowledge, the mechanistic portion of it can be automated to save time and resources so that the human players of the process can devote their time to more meaningful tasks such as the negotiation of the budget by concentrating on the list of numbers marked as "source of discrepancy".

According to the analysis of the case, some of the disputed differences can be attributed to **workload increase**, and other parts attributable to **salary mix change** and **unanticipated one-time computerization implementation costs**. These pieces of information are, however, scattered throughout the case. The activities and data cited to support both parties' claims during the 3 stages of information exchange are outlined in the following table:

<u>Sources of Descrepancy</u>	<u>Amount</u>
One-Time Cost	\$224,610
Salary Mix Change	183,120
Work Load Increase	252,270
Descrepancy (given)	<u>\$660,000</u>

Table 6. Sources of Discrepancies in Dollar Amount

The sources of discrepancy come from three main areas: one-time computerization costs, salary mix changes and work load increases due to reorganization. The detailed dollar figures are presented in the next table (Table 6). The following table shows the discrepancies attributable to one-time costs, from computer and salaries (Table 7) calculated based on information supplied in the case.

<u>One-Time Cost</u> (Stated in the Case, p.465)	
Computer	\$80,000
Salaries	144,610
	<u>\$224,610</u>

Table 7. Discrepancy Caused by One-time Costs

In Table 8 discrepancies caused by changes in salary mix are calculated and presented.

<u>Salary Mix Change</u>		<u># Personnel</u>	<u>Total Salary</u>
Average Salary 83	\$15,768	109	\$1,718,712
Average Salary 82	\$14,088	109	1,535,592
			<u>\$183,120</u>

Table 8. Discrepancy Caused by Salary Mix Change

Solving the case manually helps to identify the objective of successfully automating the budgeting process. Specifically, an automated system should help the negotiation parties to identify and focus their negotiation efforts on the sources of discrepancy identified in the above three tables.

Chapter Four

AUTOMATION - CURRENTLY AVAILABLE TECHNOLOGIES

1 Potential for Automation

The budgetor and budgetee had to manually go through all the steps of exchange of information as described in the Case Summary chapter in order to clarify differences in information. They then had to identify the sources of discrepancy by performing a series of calculations, some of which are described in the Case Solution section of the last chapter. Unless these steps were carried, they could not begin their negotiation on the "real agenda". Therefore, if this process was automated, time and resources could be saved so that efforts could be directed towards resolving the real issues. Ideally, an automated system could be designed by following the algorithm described in the last chapter to handle the task of

narrowing the scope of differences. That is to say, the automated system should be able to “pull” relevant information from both parties databases and narrow the differences until the specific sources of discrepancy are pinpointed so that decision makers can be called upon to carry out negotiations. Therefore, the final product of automation must provide information storage, mathematical calculation and logical reasoning capabilities.

This chapter evaluates a number of information technology tools available on the market to support budget automation. Some of these tools are already employed extensively in the field of accounting, or more specifically, budgeting. On the other hand, some of the tools still in the stage of research and development and are not yet widely adopted by the business world. The following tools are being evaluated individually in terms of functionality they provide and how well they address the areas of information storage, mathematical calculation and logical reasoning capability.

2 Spreadsheet

Spreadsheet applications are computer programs that allow you to create and manipulate spreadsheets electronically. In a spreadsheet application, data are organized on a cell basis. The type of data and value in each cell can be defined individually. In addition, the relationships between cells are defined by formulas, which could be mathematical formulas, statistical formulas or financial formulas. Once the data is entered, and values and formulas are defined, any changes made to the individual cell will automatically be reflected in the related cells. This feature enables the user to perform various types of analysis. Lotus 1-2-3 and Microsoft Excel are the most popular spreadsheet applications on the market. There are other advanced features, such as graphical representation and macro language,

present in the spreadsheet to allow users to manipulate data and perform analysis in order to extract more meaningful and informative facts.

The ease of use of spreadsheets has contributed to their popularity in the business world today. It brings a certain degree of automation to the budgeting process by acting as a powerful calculator. However, it fails to deliver an answer to our requirements since it does not address the logical reasoning capability. The budgetor and budgetee can store their data and perform calculations by using spreadsheets. However, they still have to compare figures and trace differences manually.

3 Relational Database Management Systems (RDBMS)

Relational databases “organize data in simple two-dimensional tables ... which can support powerful data manipulation ... and is much more flexible” (Reynolds, 1992) because the same database can be viewed in many different ways. RDBMS is a group of applications that allow users to perform data management , which are “data collection, storage, and retrieval” (Rob, 1993), on relational database. Although there are a number of different types of DBMS, RDBMS is still by far the most popular one in the market and in fact “almost every vendor’s announcement for new database management software is a relational product” (Reynolds, 1992).

For the purpose of budget automation, again like spreadsheets, RDBMS can provide a partial solution by providing data storage and retrieval capabilities. However, they suffer from a similar drawback as spreadsheet applications. Using RDBMS, users are still expected to calculate and compare results manually. In addition, this technology does not

provide the budgetee and budgetor a mechanism for interacting and communicating. A higher degree of automation, which exceeds the capability of RDBMS, is required.

4 Object-Oriented Database Management System (OODBMS)

“During the past few years the data-management and application environment has become far more complex” (Rob, 1993) and such complex application environment called for a new type of DBMS, Object-Oriented Database Systems. OODBMS “use a subset of Object-Oriented concepts and the Object-Oriented data-model features”(Rob, 1993). Although OODBMS is still in the infancy stage of research and development, it offers a much needed **versioning** feature for automating budgeting process.

It is not unusual for negotiating parties to revise the estimates of their accounts when preparing budgets many times, therefore for recording purposes, it is necessary to maintain the record of different versions of the account. Versioning was originally designed “to allow the tracking of the evolution of object states” (VERSANT ODBMS, 1995, p. 7-1) and “allow multiple users to share access to the same objects, enabling each to update those objects as necessary without interfering with other users” (Loomis, 1995). When automating the budgeting process, this feature provides a convenient way to keep track of the changes made to accounts.

However, OODBMS fail to provide a mechanism for users to reconcile differences. It is far from satisfactory to support the budgeting process.

5 Accounting Software

Accounting software is the name of a class of computer programs that perform accounting operations. Popular accounting software packages available on the market generally support the functions for general ledger, accounts receivable, and accounts payable. More sophisticated systems also support functions for payroll, inventory, invoicing, and fixed assets. Some high-end systems even support sales analysis, budgeting and forecasting. Products like SAP, PeopleSoft, ACCPAC Simply Accounting etc. provide a spectrum of choices for businesses (information of these applications can be found at their company websites). The size of these applications range from providing support for business transactions of large corporations to small grocery stores. Using these products, users can carry out detailed accounting functions, generate reports and perform advanced analysis, but none of them provide support for budget negotiation.

6 Software Available Supporting Negotiation

Narrowing the scope of difference is both studied and documented by other researchers. However, no work has been done to automate this stage. For example Researchers, such as ManKit Chang and Carson Woo, recognized that clarification of information and narrowing of differences before negotiation are important steps involved in negotiation. However, their Speech-Act-based Negotiation Protocol (Chang and Woo, 1994) only lay down communication protocol. Because the actual automation of narrowing the differences is only one small step in the negotiation protocol study, it is not covered here in depth.

In addition, the concept of narrowing down the scope of differences introduced in budgeting is not the same as "Narrowing the difference" in Gulliver's 1979 paper since his eight-phase procedural model of negotiation does not involve the exchange of information. The negotiators mentioned in his model are merely trying to narrow the differences by examining identical sets of data, whereas in budgeting new data can be introduced by one of the negotiation parties at any point in time.

Narrowing the differences consists purely of exchange of information and no tactics are involved. Yet, this is an essential step that the negotiators must go through before two parties can have common ground for negotiation. Group decision support systems such as PERSUADER (Sycara, 1991) and NEGO (Kersten, 1991) that support negotiation attempted by researchers do not help in narrowing down the differences between negotiation parties, rather they try to help the players win the negotiation. However, this is not the objective of the automation proposed in this thesis.

Chapter Five

IMPLEMENTATION

Based on the information collected and analysis performed on the National Motors Case, a system is created to demonstrate the use of information technology in automating part of the budgeting process. In addition, this system is used as a prototype to study the benefits and limitations of this type of system and to uncover the difficulties encountered during the development of the system for future reference. The system runs on a Windows platform with SWI-Prolog installed. This system is written in Prolog language in order to take advantage of its “powerful inference mechanism based on resolution” (Van Le, 1993). Also the declarative style of predicate calculus of the language provides a convenient way for data storage.

This chapter explains the logical and structural design of the system by presenting the design algorithm and the system structural chart. In order to store the accounting data in the

system so that the system can function based on the existing known data, a mechanism for representing the knowledge in Prolog was developed and presented in the chapter. The data provided in the National Motors case were fed into the system for testing. Finally the testing results are summarized in the last section of this chapter.

1 Algorithm for implementation

The information clarification process and budget approval process summarized in Chapter Three (Figure 4) of this thesis were analyzed and revised to serve as a foundation algorithm for building the system. The revised algorithm is summarized in the flowchart below (Figure 5).

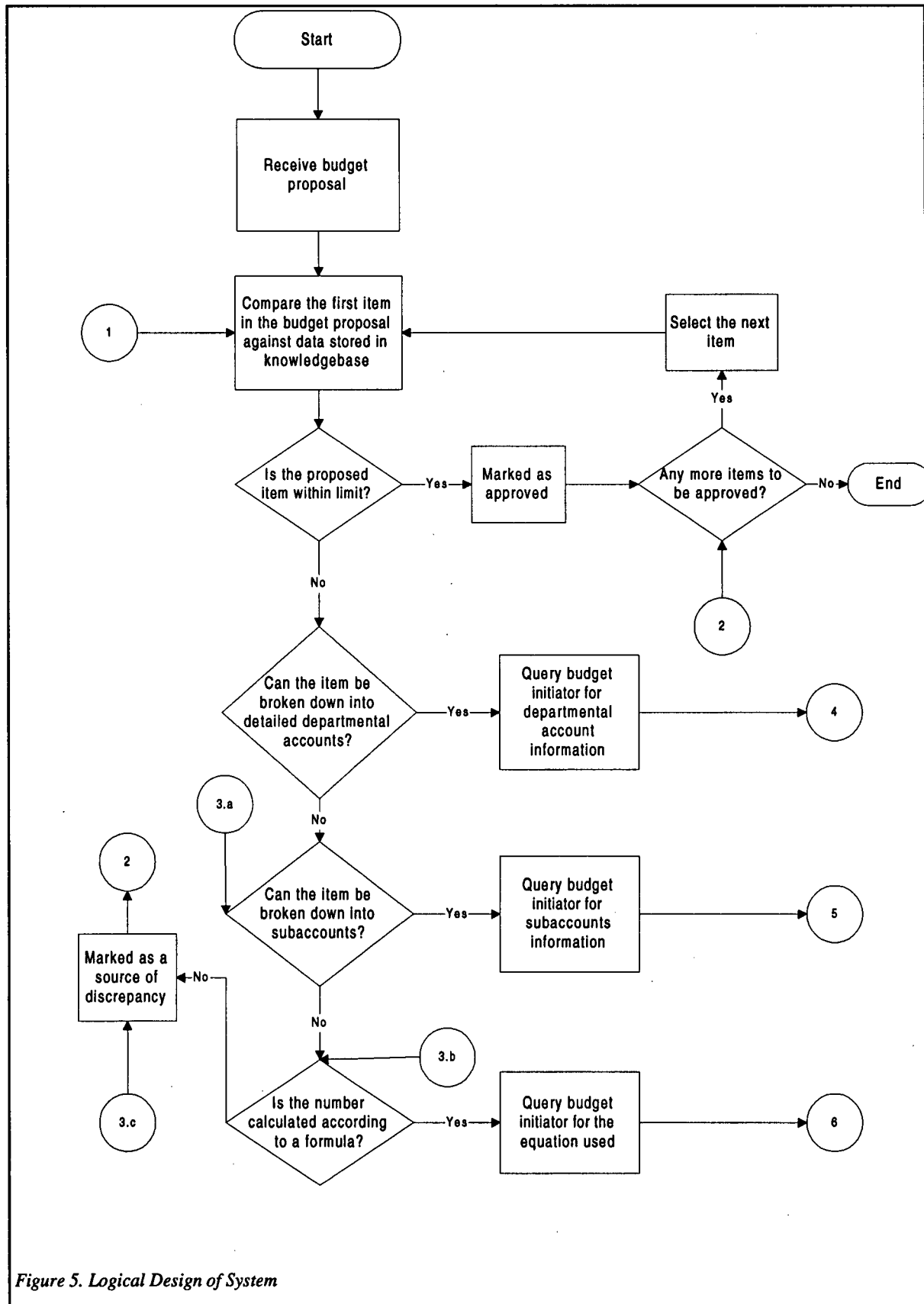


Figure 5. Logical Design of System

This revised algorithm contains detailed step by step logical design of how a budgetary request would be handled, for example the supplementary budget proposal in the National Motors Case. The algorithm explains certain critical steps that must be undertaken before an item can be approved or stamped as a source of discrepancy.

1.1 Assumptions Used in the Algorithm

Certain assumptions are made in the algorithm:

1. The term "item" in the algorithm refers to a particular account submitted by the budget initiator in a pre-defined format³ that is understood by all parties involved.
2. The term "limit" refers to a tolerance level that specifies the percentage of deviation allowed for in a budget request. This piece of knowledge is predefined for and specific to each account and can vary from account to account. For example, a 5% limit for the Total Expense account means the account will be approved as long as the requested amount falls within $\pm 5\%$ of the predefined amount.

1.2 Initial Checking

The system is started after receiving an initial budget request. To begin with, it checks the first requested item against the data stored in its own database.

For example, a request for a budget of \$120,000 is received from an Asian operation at the corporate headquarters. The system compares this amount against the amount of the same account stored in the headquarters' database. If the headquarters' data agrees with the requested item, it would be approved and the next item in the requested item list will be

checked following the sequence of the list. However, if the headquarters' data specifies that the estimated amount should only be \$100,000 for the Asian operation, the system will continue to perform more checking in order to determine what causes the \$20,000 discrepancy. Further checking procedures are being conducted by disassembling the item into smaller sub-accounts according to certain schemes that are understood by all parties. These checking procedures are not conducted in any particular order. However, an account cannot be labeled as a source of discrepancy unless it has failed all the tests in the subsequent checking procedures.

³ The format of the account will be further discussed in the latter part of this chapter.

1.3 Further Analysis - Departmental Accounts

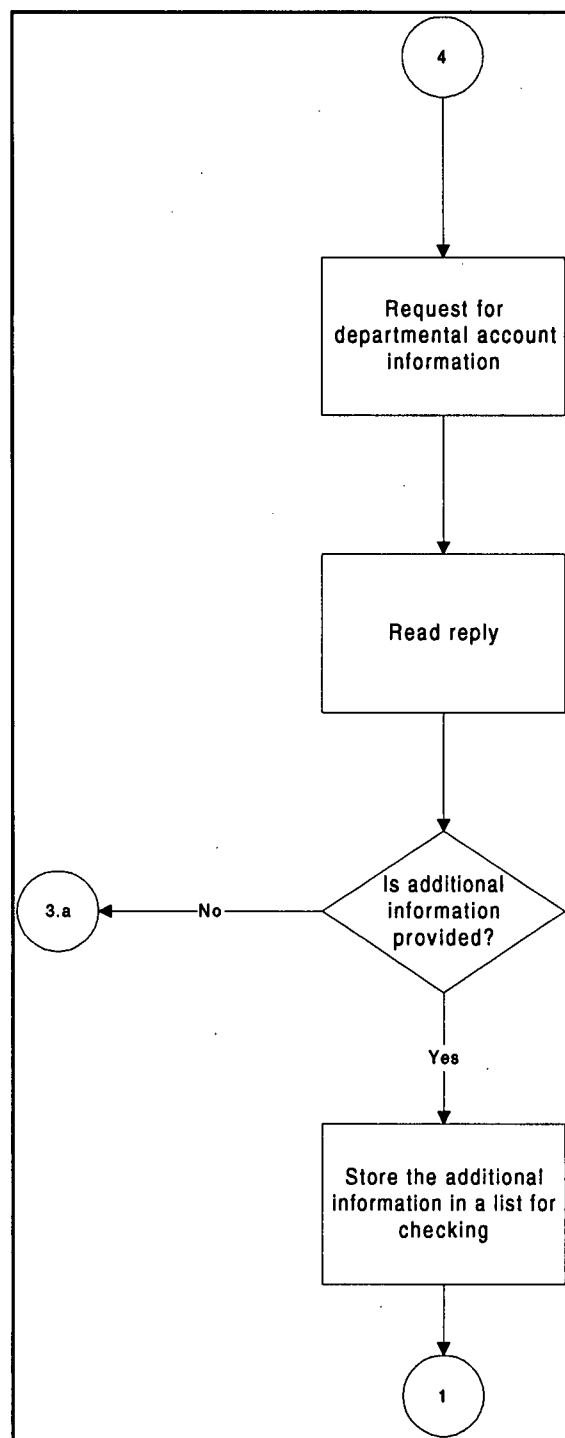


Figure 6. Further Analysis - Checking Divisional Sub-Accounts

Following the failure of initial checking, the system analyzes the budget item by breaking it down into smaller components and requesting the budget initiator to submit information on them. The system then checks these components by going through the returned data.

For example, following the example in the previous section, the system checks the headquarters' database to see if the total budget figures can be broken down into departmental accounts, i.e. Asian division A budget, Asian division B budget. The system then sends a request to the budget initiator for breaking the \$120,000 into subsequent Division A and Division B accounts. If the initiator does not want to submit the data in the

requested format, he/she can simply choose to refuse the request. The system will attempt to perform other tests by asking for information organized in another format.

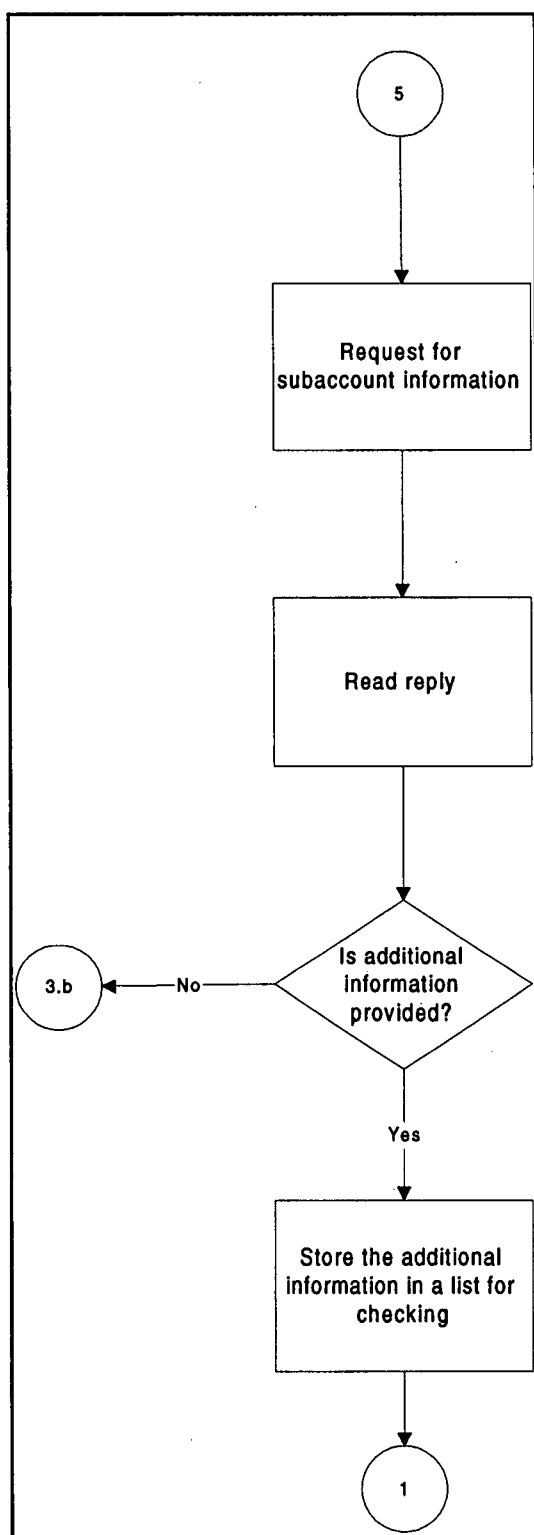


Figure 7. Further Analysis - Categorical Sub-accounts

1.4 Further Analysis - Categorical Sub-accounts

If the total budget cannot be broken down into departmental accounts, the system will attempt to analyze the total budget figures by breaking them down into sub-accounts of types of expense or revenue.

For example, if the information of division A and B is not available from the Asian operation, the system will attempt to check if the total budget can be broken into sub-accounts, i.e. Asian Marketing Expense, Asian Salary and Wages, Asian Office Expenses. The system then sends a request to the budget initiator for breaking the \$120,000 into the following categories:

- Asian Marketing Expense
- Asian Salary and Wages
- Asian Office Expenses.

If no information is returned from the budget initiator, the next test will be carried out.

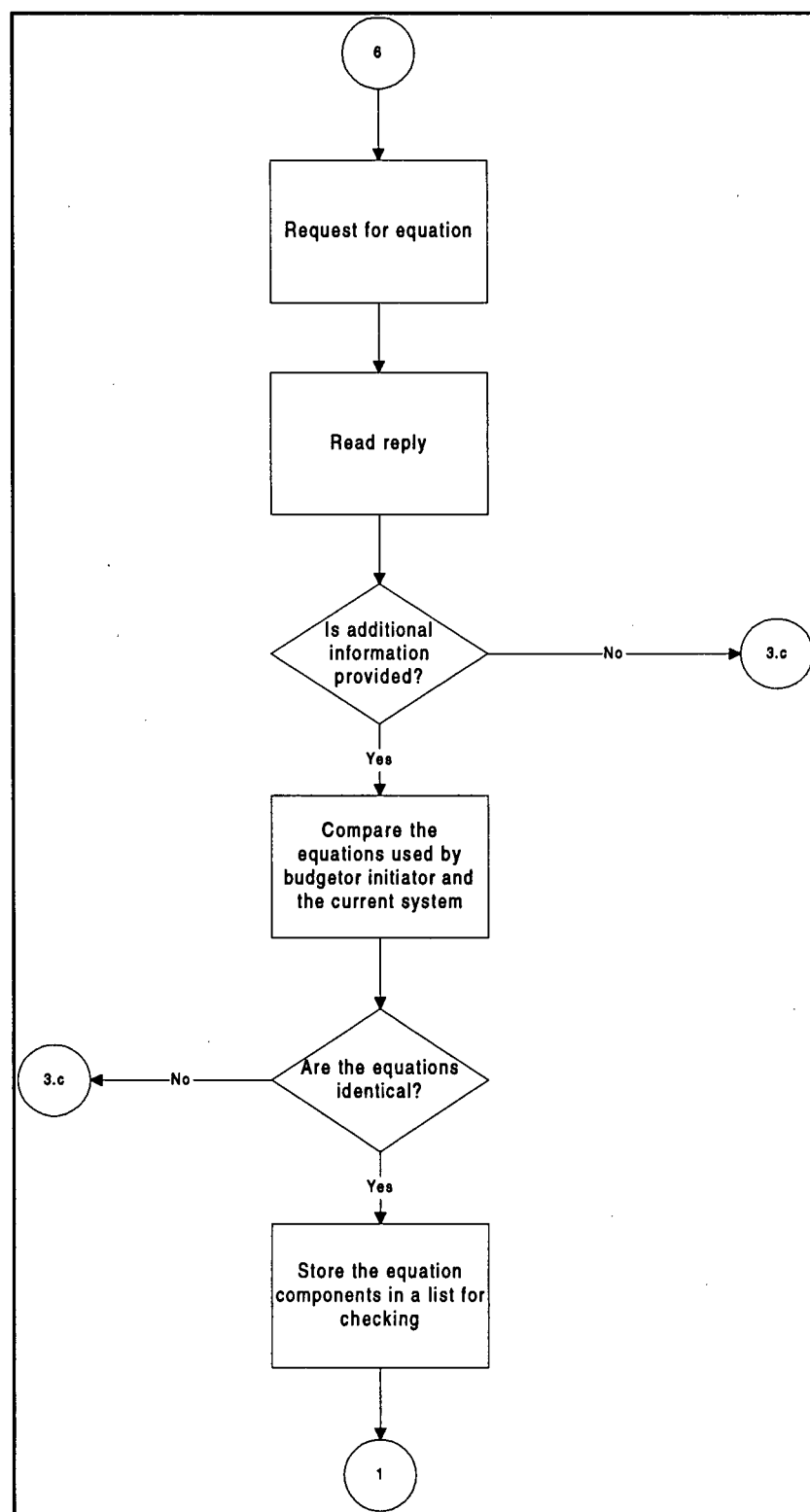


Figure 8. Further Analysis - Equation

1.5 Further Analysis - Equation

If the information of sub-accounts of expenses or revenue is not available, the system checks whether the item is calculated according to some equations stored in the system. The system sends a request to the budget initiator for information on the equation. After receiving a reply from the budget initiator, the system will compare the submitted equation and figures against

the ones stored in its own database.

For example, the system checks whether the \$120,000 is calculated based on an equation, i.e.

$\text{Last Year's Actual Budget} \times \text{Inflation Rate} \times \text{Currency Fluctuation} + \text{Contingency Allowance} = \text{Current Year Budget Proposal}$

If the result is affirmative, the system sends a request to the Asian operation's system for submitting the equation used by the budget initiator and the numbers they used when calculating the total budget.

1.6 Iterative Approach

The above 3 analysis methods are performed on every piece of information submitted by the budget initiator, regardless of whether it is part of the original budget proposal or an answer in reply of a query. An item will be marked as "Source of Discrepancy" when all 3 analyses performed and the item cannot be further divided into smaller components.

2 Structural Design of the System

The structure of a running instance of the final complete system that supports all the activities described in the previous section is represented in the following diagram (Figure 9). The system consists of a main reasoning engine which performs the three analyses outlined in the previous section on submitted items. The Main Reasoning Engine also contains an Input/Output Handler module which provides the functionality of querying other systems for information needed and interpreting any replies from other systems.

There are also a number of modules that are external to the Main Reasoning Engine, namely; the Account Database and its subsequent components, and the Equation Simplification Engine. These modules are explained in the following sections.

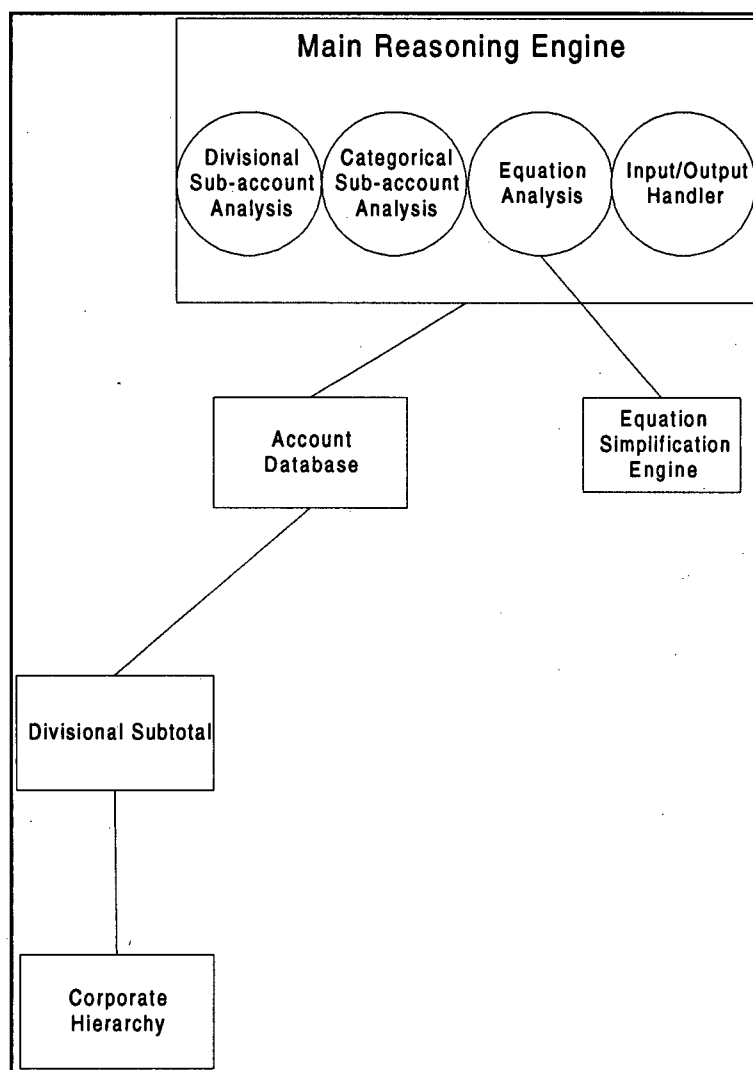


Figure 9. System Structure

2.1 Main Reasoning Engine

This module contains the main reasoning process which goes through the list of submitted items and performs analysis on each of the individual items. It sends a query to other systems for additional information required to pinpoint the source of discrepancy.

2.2 The Equation Simplification Engine

This module supports one of the analyses performed by the

Main Reasoning Engine - the

equation analysis. The equation analysis test to checks whether the equations used by two parties to calculate an account are the same. This particular module simplifies equations in their simplest terms. For example:

$$2 \times (3X + Y) - 2Y = 6X$$

By using this module, we can also check the equality of any two equations. For example, two different equations are used in two departments to calculate the value of an account:

Asian Division: $2 \times (3X + Y)$

Singapore: $6X + 2Y$

The equality of these two equations can be tested by combining them in the following way and feeding the combined equation into the Equation Simplification Engine:

$$2 \times (3X + Y) - 6X - 2Y$$

Since the engine always tries to simplify the equation in its simplest terms, in this case the result will be 0. Therefore, as long as the individual variables of the equations are defined, we can safely conclude that two equations are identical if their difference is 0.

2.3 The Account Database

This is the database that contains all the account records. For each account, the following information is stored in the database; namely, the account number, account owner, measurement, version number, account value. The account representation will be discussed in detail in the following section.

There are two pieces of modules used by the Account Database to record and establish the relationships among accounts. As can be seen in the System Structure diagram (Figure 9), they are the **Divisional Sub-total** module and the **Corporate Hierarchy** module. Using

the information stored in these two modules, the system establishes the relationship that the value of any account is the sum of the corresponding account of all divisions under the control of the owner of this account. For example, for a company with an Asian division established to control operations in Hong Kong, China, Singapore, the system will automatically treat all accounts under the Asian division as the sums of their corresponding accounts in the three regions, i.e. Asian Operational Expenses = Hong Kong Operational Expenses + China Operational Expenses + Singapore Operational Expenses.

3 Knowledge Representation

This section discusses how account information is stored in the database. In order for the system to perform analysis on budgetary items, certain information must be contained in the database such as the account number, account value and account owner. All this knowledge that is specific to this case are represented as facts. A system is designed to represent the knowledge for storing and retrieving the data efficiently. This knowledge representation convention is developed by following the chart of accounts outlined in the Principles of Accounting by H. A. Finney (Finney, Miller and Mitchel, 1965).

3.1 Basic Components of an Account

The facts are stored in a nomenclature system that allows the system to identify the information related to accounts. The facts generally take the following form:

account (KEY, [VALUE], DEVIATION ALLOWED)

where as **KEY** allows the system to uniquely identify the account, **[VALUE]** is a list of numbers containing the account's current and any historical values, and **DEVIATION**

ALLOWED tells the system how much deviation in this account can be tolerated by the reasoning engine when analyzing a budget proposal containing this account. For example,

```
account(KEY1, [99.5, 90, 82], 0.025)
```

represents the fact that the account which can be uniquely identified as **KEY1** has a current value of **99.5** and its historical values are **90** and **82** respectively, and this account can still be approved if it is exceeded by no more or less than **2.5%**.

Currently the deviation allowed is interpreted as plus or minus the percentage of the original value. It is possible to adjust the system to just having an upper level limit or one lower level limit depending on the nature of account.

3.2 Account Key

In order to identify and compare accounts, we need more information than just values and deviation allowance. Using the above example, the knowledge of "account **KEY1** has a value of **99.5**" can be established. However, the system has no knowledge whether this number, "**99.5**", refers to dollar amount or number of personnel. Therefore, the **KEY** component is expanded to contain the following four pieces of data: **TYPE**, **ACCOUNT NUMBER**, **UNIT** and **OWNER**. The complete structure of the account is as follows:

```
account(key(TYPE, [ACCOUNT NUMBER], UNIT, OWNER), [VALUES],
DEVIATION ALLOWED)
```

This is best illustrated by the following example:

```
account(key(exp, [1, 0, 0, 0], dollar, manufacturing),
[2916000], 0.1)
```

This fact tells the system that the account it is dealing with is, in fact, an expense account (represented by the keyword **exp**) which has the account number of [1, 0, 0, 0], is owned by

the manufacturing office (**manufacturing**), has a value of \$2,916,000, and cannot exceed or fall below 10% of its original value.

The following table (Table 9) provides examples of some of the keywords used to represent the information in the National Motors Case for each account component.

ACCOUNT COMPONENT	KEYWORDS	LEGEND
Type	<ul style="list-style-type: none"> • exp • rev • asset 	<ul style="list-style-type: none"> • Expense • Revenue • Asset
Unit	<ul style="list-style-type: none"> • person • dollar • parts 	<ul style="list-style-type: none"> • Number of people • Dollar amount • Parts count
Owner	<ul style="list-style-type: none"> • Manufacturing • Controller 	<ul style="list-style-type: none"> • Manufacturing Office • Controller's Office

Table 9. Examples of Keywords Used in the National Motors Case

3.3 Account Number

The last example shows a list representation of the **account number** ([1, 0, 0, 0]). An important feature of this list representation of the account number is that it allows the representation of a hierarchical order of accounts and sub-accounts (Table 10). This hierarchical structure of account numbering provides a convenient method of representing the relationship between master account and sub-accounts.

ACCOUNT	ACCOUNT NUMBER
Total Expenses	[1, 0, 0, 0]
Total Expenses - Accounting	[1, 1, 0, 0]
Total Expenses - Production	[1, 2, 0, 0]
Production - Personnel Exp	[1, 2, 1, 0]
Production - Machinery Exp	[1, 2, 2, 0]

Table 10. Hierarchical Order of Account Numbers

4 Testing

The system is developed and tested using the data provided in the National Motors Case. The goal of the testing is to solve the case with the assistance of the system. The implementation is deemed successful if the system is able to identify the sources of discrepancy. In other words, the system is successfully implemented if its final result matches the results derived manually in Chapter 3 of this thesis.

In order to simulate communication and exchange of information between the controller's office and the manufacturing office, we assume that both the manufacturing office and the controller's office possess their own copies of the system with separate databases containing data that reflects their knowledge of the situation. The communication between two parties is achieved through file sharing. The following diagram depicts the

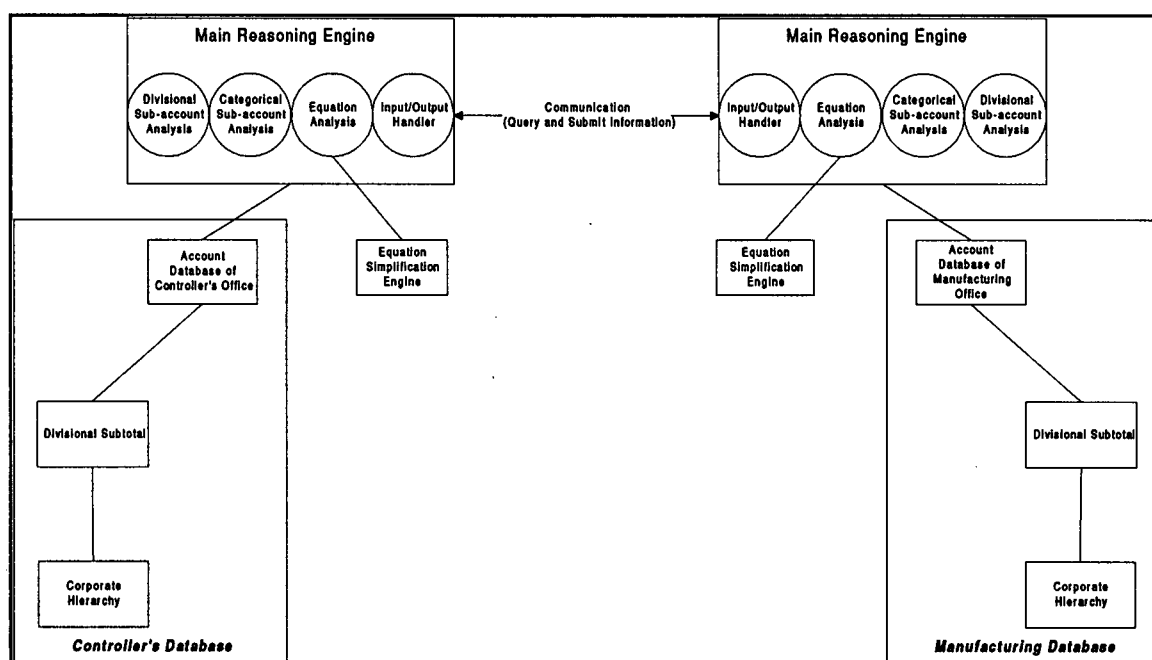


Figure 10. Relationships Between Testing Systems

relationship of the two systems (Figure 10).

As illustrated in the above diagram, the two systems have identical components except for the account data residing in the database. These two versions of databases are necessary since the two parties possess different knowledge of the situation. The different knowledge causes differences in beliefs when they develop and evaluate the initial budget proposal. The purpose of the system is to narrow differences by exchanging and sharing information between the two parties.

4.1 The Screen Capture

The system is tested by using data from the National Motors Case. Two instances of the system are being run under Windows95 environment to simulate the situation of two separate systems used by the budgetor and budgetee. The following screen (Figure 11) shows the initial response of the Controller's system after receiving the initial supplemental budget proposal. It attempts to check the submitted proposal against its own record. Upon realizing that its own record does not seem agree with what has been submitted, it sends a request to the manufacturing office's system for more information.

```

SWI-Prolog (version 2.1.14)
sys / subtotal compiled, 0.05 sec, 956 bytes.
sys / simplify compiled, 0.50 sec, 57,472 bytes.
sys / engine compiled, 0.82 sec, 76,844 bytes.
main compiled, 0.99 sec, 83,812 bytes.

Yes
2 ?- t.

***
Checking the following item: account(key(exp, [1,0,0,0], dollar, manufacturing),
[2916000], 0.100000, Manufacturing total budget)

The requested item: Manufacturing total budget (key(exp, [1,0,0,0], dollar, manu
facturing)) exceeds limit.
Ask the originator to supply information on subsidiary divisions.

Writing command file to request detail of Manufacturing total budget (key(exp, [
1,0,0,0], dollar, manufacturing))

Please start another instance of prolog to generate response.

When response is received, type 1 to continue the current process.

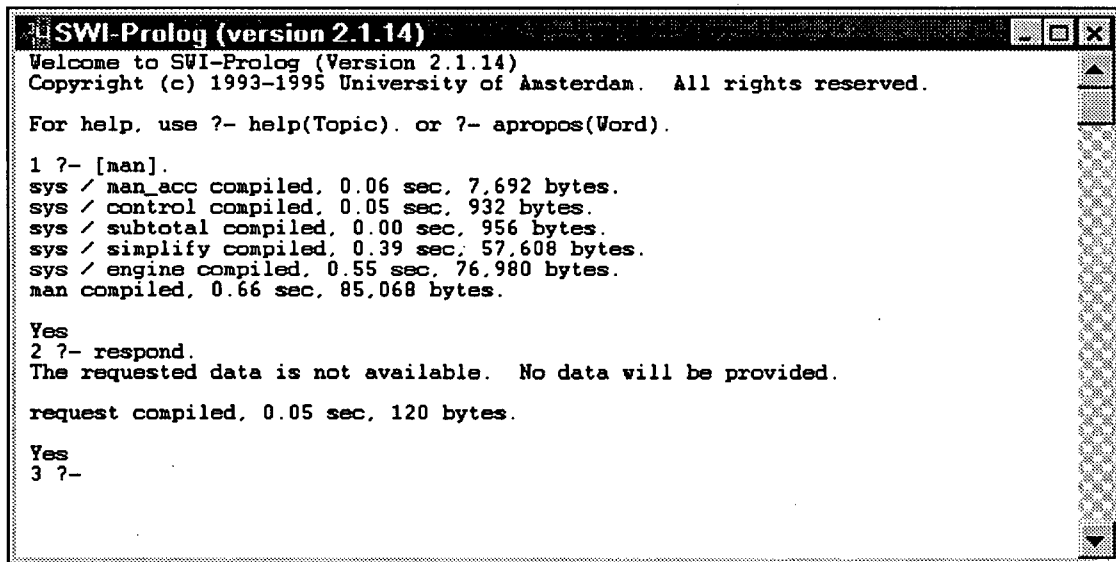
|:

```

Figure 11. Controller's Initial Response

The manufacturing office's system attempts to answer the request by checking its own database for information. It finds out that the requested information is not available, and sends a reply back to the requester (Figure 12).

The Controller's system then applies the second test to the budget proposal by reorganizing the account in the format of categorical sub-accounts and sends a second query



```

SWI-Prolog (version 2.1.14)
Welcome to SWI-Prolog (Version 2.1.14)
Copyright (c) 1993-1995 University of Amsterdam. All rights reserved.

For help, use ?- help(Topic). or ?- apropos(Word).

1 ?- [man].
sys / man_acc compiled, 0.06 sec, 7,692 bytes.
sys / control compiled, 0.05 sec, 932 bytes.
sys / subtotal compiled, 0.00 sec, 956 bytes.
sys / simplify compiled, 0.39 sec, 57,608 bytes.
sys / engine compiled, 0.55 sec, 76,980 bytes.
man compiled, 0.66 sec, 85,068 bytes.

Yes
2 ?- respond.
The requested data is not available. No data will be provided.

request compiled, 0.05 sec, 120 bytes.

Yes
3 ?-

```

Figure 12. Manufacturer Unable to Provide Information

to the manufacturing office's system (Figure 13) to ask for information in this format.

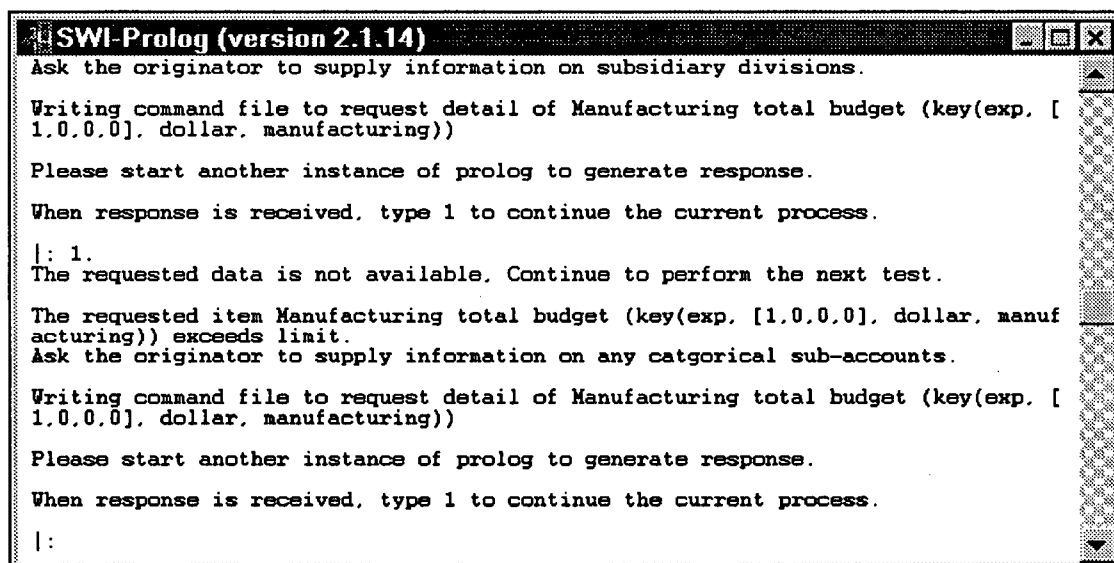


Figure 13. Query Sent After Second Test

This time the manufacturing office finds an answer in its database. It writes the reply to a text file so that the controller's system can read the necessary information from it (Figure 14).

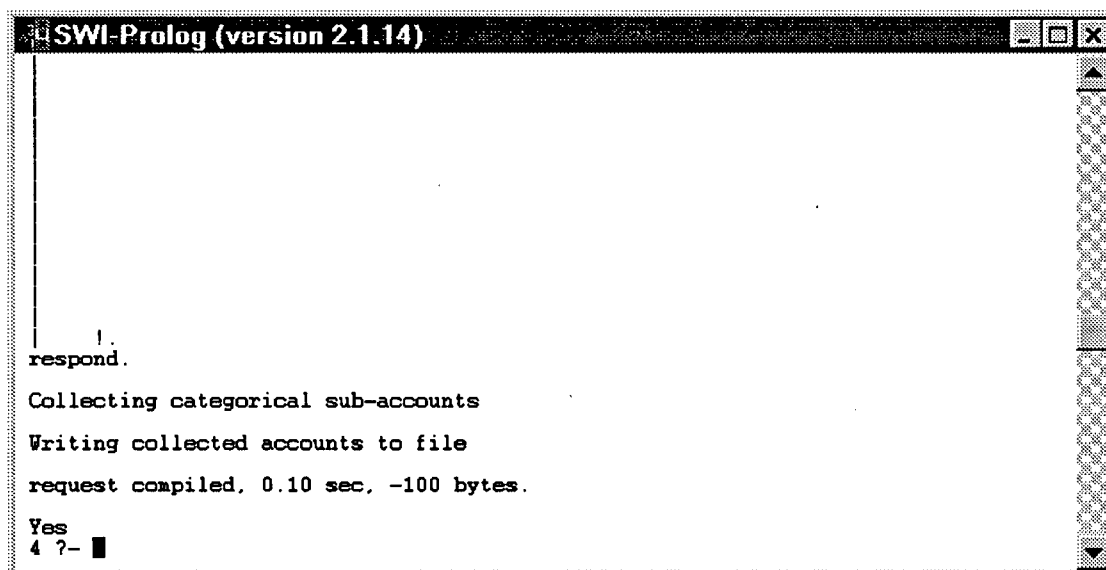


Figure 14. Manufacturing Office's Reply

After reading the reply from the manufacturing office, the controller's system starts to analyze the items included in the reply. It then attempts to either approve or disapprove the items by applying the same type of test (Figure 15).

```

SWI-Prolog (version 2.1.14)
1.0.0.0], dollar, manufacturing))

Please start another instance of prolog to generate response.

When response is received, type 1 to continue the current process.

|: 1.

***
Checking the following item: account(key(exp, [1.1.0.0], dollar, manufacturing),
[2320000], 0.100000, Manufacturing personnel expense)

The requested item Manufacturing personnel expense (key(exp, [1.1.0.0], dollar,
manufacturing)) exceeds limit.
Ask the originator to supply information on any categorical sub-accounts.

Writing command file to request detail of Manufacturing personnel expense (key(e
xp, [1.1.0.0], dollar, manufacturing))

Please start another instance of prolog to generate response.

When response is received, type 1 to continue the current process.

|:

```

Figure 15. A New Round of Examinations

When the system encounters an item calculated according to an equation, such as salary expenses, it asks originator to supply the equation. It then sends the returned equation and its own equation to the equation simplification engine to test if the two equations are indeed identical. If they are identical, the system will continue to compare the equations component by component. Otherwise, the system will label them as a source of discrepancy.

The following section shows a log file which records the whole process of how the system solves the National Motors Case.

4.2 Log

This section shows the result of the system which has successfully identified the sources of discrepancy in the National Motors Case. The complete log file is available in the appendix section which was designed to be as self explanatory as possible for each and every step that the system went through.

The system was able to identify all discrepancies in the dollar amount. As we recall from Chapter 3, the discrepancies are as follows:

<u>Sources of Discrepancy</u>	<u>Amount</u>
One-Time Cost	\$224,610
Salary Mix Change	183,120
Work Load Increase	252,270
Discrepancy (given)	<u>\$660,000</u>

The system identified the discrepancy caused by One-Time Costs in the following operation:

```

Controller: ***

Controller: Checking the following item: account(key(exp,
[1,1,1,0], dollar, manufacturing), [144610], 0.100000, One-
time personnel cost)

Controller: One-time personnel cost = 144610 (key(exp,
[1,1,1,0], dollar, manufacturing)) exceeds limit ---> NOT
APPROVED

.
.
.

Controller: ***

Controller: Checking the following item: account(key(exp,
[1,3,2,0], dollar, manufacturing), [80000], 0.100000, One time
computer cost)

```

Controller: One time computer cost = 80000 (key(exp, [1,3,2,0], dollar, manufacturing)) exceeds limit ---> NOT APPROVED

Salary Mixed change is identified as:

Controller: ***

Controller: Checking the following item: account(key(exp, [1,9,9,9], dollar, manufacturing), [15768,14088], 0.100000, 1983 Average salary)

Controller: 1983 Average salary = 15768 (key(exp, [1,9,9,9], dollar, manufacturing)) exceeds limit ---> NOT APPROVED

And the increase due to workload is identified as:

Controller: ***

Controller: Checking the following item: account(key(exp, [1,1,3,2], dollar, manufacturing), [252270], 0.100000, Workload increase)

Controller: Workload increase = 252270 (key(exp, [1,1,3,2], dollar, manufacturing)) exceeds limit ---> NOT APPROVED

4.3 Personnel Discrepancy Reconciliation

As shown in the previous section, the system successfully identified the discrepancies in terms of dollar amount, it was not able to identify cause of discrepancy in terms of number of personnel. However, this is not a deficiency of the system. Although the manufacturing office's system contains this data, the controller's system does not have enough knowledge to ask the right questions in order to identify the source of discrepancy in number of personnel.

Chapter Six

CONCLUSION AND FUTURE WORK

In this thesis, our interest is in studying the automation of budget negotiation. But since the budget negotiation process is such a broad topic and is also complicated by the many behavioral issues involved in it, we decided to focus our efforts on the mechanistic aspects of budgeting. Traditionally much time and effort have been spent in identifying the real issues and reconciling discrepancies in order to prepare the budgeting parties for negotiation. In other words, although budgeting is a well studied area, not enough effort has been paid to studying the automation of processing. Therefore, we concentrated our research efforts on narrowing the differences in preparing the budgeting parties for negotiation.

This chapter concludes this thesis and research. The corporate budgetary process was first studied and behavioral factors complicating the budgeting process were identified. The

goal of automation is not to completely eliminate human involvement in the budgeting process, rather the goal of automation is to provide human players with more relevant information, and to free up more time and resources for them to concentrate on the relevant issues of budget negotiation.

This study utilizes a case that was created based on a real-life situation. Based on the findings in this case, a set of criteria for automation were identified. A series of existing information systems and tools readily available on the market were evaluated and criticized based on this set of criteria. We have come to the conclusion that as with all fine systems on the market, they fail to meet one or more of the following criteria: information storage, mathematical calculation and logical reasoning capability.

A system was then designed to automate the task of narrowing the differences. The system was implemented on a Windows 95 platform and written in Prolog language. Throughout the implementation, a convention was developed to represent the knowledge of budgeting process and accounting information.

This thesis demonstrates usage of the system by applying it in a "bottom-up" budget preparation scenario. The example shows how the controller can approve or disapprove budgets prepared by a subsidiary. However, the system also works in a "top-down" scenario in which the corporate controller prepares the budget and forces the subsidiary to comply with it. The system has the ability to identify source of discrepancy between any two budgets regardless of the budgeting parties' position in the corporate hierarchy. For example, the subsidiary can use the system to identify any sources of discrepancy between controller's prepared budget and its own understanding of the financial situation.

In order to evaluate the effectiveness of the system, data of the National Motors case were then fed into the system for a test run. Through the implementation and testing of the system the following areas were identified:

1 Limitations of the Study

Due to time and resource constraints, only one case was used to test the usability of the system in this study. To provide more meaningful results, more real life cases can be used for the testing process. It is desirable to test the system, possibly in a real business organization setting.

The system successfully provides a list of source of discrepancy to user. However, it lacks an explanation engine. Users cannot query the system on how it arrives to a particular set of answer.

2 Future Work

The research can be extended in several ways. First, the system and concept of budget automation can be tested in business organizations to study how well the business world will adopt the concept of budget automation.

Second, an explanation engine can be added to the system to allow users to query the reasoning behind each recommendation.

Third, we also recognize there are behavioral factors that inhibit users from disclosing and sharing information. The system can therefore be perfected by adding a function which allows users to attach an access level to each piece of data so that data can be retrieved

selectively based on the discretion of the owner of the data. By doing so, users are provided with the flexibility of disclosing only the information they think is suitable.

Last but not the least, this thesis provide information clarification concept that can be extended to develop a general inquiry tool. Effort could be directed towards studying the integration of this inquiry tool with the existing accounting systems.

3 Conclusion

As stated in the beginning of this thesis, this study attempts to bring automation to the budget negotiation process by employing a computer-based system. As shown by the developed system, this automation is brought to the area of narrowing the scope of differences in the budget negotiation process. Although information clarification is a small step in budget negotiation, it results in a significant amount of saving in time and resources, once this process is automated. More effort could be made towards studying the automation of the budget negotiation process based in this thesis.

BIBLIOGRAPHY

1. Anthony, R. N., Dearden, J., and Govindarajan, V. *Management Control Systems (7th Edition)*, Irwin: Boston, Massachusetts, 1992.
2. Arnold, J., Hope, T., *Accounting for Management Decisions (2nd Edition)*, Prentice Hall: Hertfordshire, 1983.
3. Carruth, P. J., McClendon, T. O., and Ballard M. R., "What Supervisors Don't like about Budget Evaluations," *Management Accounting*, February 1983, p. 42.
4. Chang, M. K. and Woo, C., 1994, *ACM Transactions on Information Systems*. 12, 4, 60-382.
5. Cyert, R. M. and March, J. G., *A Behavioral Theory of the Firm*: 1963: Prentice Hall.
6. Finney, H. A., Miller, H. E., and Mitchell, C. L., *Principles of Accounting - Canadian Edition*, Prentice-Hall: New Jersey, 1965.
7. Garrison, R. H., Chesley, G. R., and Carroll, R. G., *Managerial Accounting - Concepts for Planning, Control, Decision Making*, Irwin: Boston, Massachusetts, 1990.
8. Kersten, G. E., Michalowski, W. Szpakowicz, S., and Koperczak, Z., 1991, Restructurable representations of negotiation, *Management Science*, 37, 10(Oct.), 1269-1290.
9. Loomis, M. E. S., 1995, *Object Databases The Essentials*: Addison-Wesley Publishing Company
10. Reynolds, G. W. "Information Systems for Managers," West Publishing Company, 1992
11. Rob, P and Coronel, C., 1993 *Database Systems: Design, Implementation, and Management*: Wadsworth Publishing Company.
12. Schiff, M. and Lewin, A. Y., 1968 Where traditional budgeting fails. *Financial Executive*, May, pp. 50-62.
13. Siegel, G., Ramanauskas-Marconi, H., 1989, *Behavioral Accounting*: South-Western Publishing Co.
14. Sycara, K. P., 1991. Problem restructuring in negotiation. *Management Science*, 37, 10 (Oct), 1248-1268.

15. Van Le, T., 1993, Techniques of Prolog Programming with Implementations of Logical Negation and Quantified Goals, 1993, John Wiley & Sons, Inc.
16. Versant ODBMS, VERSANT Concepts and Usage Manual, 1995, July.

APPENDIX A

Case 9-1

National Motors, Inc.*

William Franklin, controller of the Panther Automobile Division of National Motors, a manufacturer of numerous products including a wide line of automobiles and trucks, was faced with a difficult decision in 1983. The manufacturing office had submitted a supplemental budget in which it had requested additional funds for increased administrative costs in its operations control department. The controller's office had written a memorandum in reply, explaining why it thought this request was not justified, and the manufacturing office had now answered this memorandum.

Mr. Franklin now had three possible courses of action: (1) to concur with the manufacturing offices' position, in which case the request would undoubtedly receive the necessary approval of the general manager; (2) to continue his opposition, in which case his views and those of the manufacturing office would be placed before the general manager, who would decide the issue; and (3) to reply with a further analysis in the hope that the manufacturing office would become convinced of the soundness of his position.

During the last quarter of 1982, the Panther Automobile Division had absorbed the manufacturing activities of the Starling Automobile Division. Responsibility for product planning and marketing of Starling cars, however, remained with the Starling Division.

A major reason for the consolidation of manufacturing activities was to reduce operating costs. There had been an anticipated annual saving in

the operations control department, for example, of \$368,000 from a reduction in the number of salaried personnel by 24. There also had been an expected saving from computerization of parts control in that department.

Prior to the consolidation, the Starling Division had a computer system of parts control in its operations control department. The Panther Division, on the other hand, had been using a manual system in its corresponding department.

In December 1982, a study was made to determine which system would serve the division best. On the basis of an estimated reduction of 23 salaried people and of \$276,000 in salary and other costs in the operations control department, because of computerization, the decision was made to completely computerize the Panther Division's system of parts control. The results of this study were concurred by the manufacturing office of the Panther Division.

Strong budgetary control was exercised throughout the National Motors organization. In the fall of each year, every department manager developed his or her proposed budget for the next year. Each proposed budget then entered an extensive process of analysis, revision, consolidation, review, and approval by higher levels of management, first within a division and then at the corporate level. At each management level, budgets for subordinate units were consolidated prior to submission to the next higher management level. Controllers at the divisional and corporate levels participated actively in this process.

* This case was prepared by J. Hekimian under the supervision of R. N. Anthony, Harvard Business School. Copyright © by the President and Fellows of Harvard College. Harvard Business School case 161-004.

Once formal approval had been given a budget, it became a firm commitment for the responsible manager. He or she could not exceed this budget except by submitting and obtaining approval of a supplemental budget. A supplemental budget was prepared and processed in essentially the same way as the original budget. Policy prescribed that a supplemental budget be justified on the basis of changes in conditions after the original budget was approved.

The 1983 budgets had been approved prior to the consolidation of the Panther and Starling Divisions, so there was a separate budget for each division and, consequently, for the operations control departments of the two divisions. The approved budgets for 1983 for these two operations control departments are summarized below, together with the estimated savings resulting from consolidation and control computerization.

Exhibit 1 shows the Starling Division's personnel ceiling commitment as of December 31, 1982, the expected saving in numbers resulting from the consolidation, and the proposed ceiling for 1983. Since the manufacturing office believed that the new consolidated system of parts control in the operations control department was going to be about the same as the Starling Division's computerized system, the standards that it used to develop the proposed personnel requirements were based on the Starling Division's work load and authorized personnel levels for 1982.

Specifications Control Section (25 People)

The work load determinant used in this activity was the number of specifications requests to be processed. In the previous year, 20 employees had been approved in the Starling Division's specifications control section: 5 were clerical and

	<i>Panther Division</i>		<i>Starling Division</i>		<i>Total</i>	
	<i>Number</i>	<i>Dollars (000s)</i>	<i>Number</i>	<i>Dollars (000s)</i>	<i>Number</i>	<i>Dollars (000s)</i>
Budget before consolidation	76	\$1,444	55*	\$1,336	131	\$2,780
Savings from consolidation					(24)	(368)
Savings from computerization†					(23)	(276)
Total					84	\$2,136

* After the transfer of five specifications follow-up personnel out of the specifications control section.

† Based on the study completed in December 1982, concurred by the manufacturing office.

Manufacturing Office's Proposed Supplemental Budget

In April 1983, the manufacturing office proposed in a supplemental budget that the Panther operations control department, now servicing both Starling and Panther automobiles, be allotted for 1983 a personnel ceiling of 109 people to handle the combined work load. Its proposal and reasoning are summarized in Exhibit 1 and in the following paragraphs.

supervisory, 10 processed specifications requests, and 5 were involved in specifications follow-up. The specifications control procedure currently used in the Panther Division operations was generally the same as that used by Starling. But in the future, the specifications follow-up procedure would no longer be done in this section or, for that matter, in the operations control department.

In 1982, the 10 analysts in the Starling specifications control section had processed 2,964 spec-

ifications requests, or an average of 296 specifications each.

In the Panther Division, 3,680 specifications requests had been processed during 1982. The manufacturing office believed that a comparison of both Starling and Panther data, as shown in Exhibit 2, indicated that there was a definite relationship between the number of specifications requests processed and the number of unique, new model parts.

On the basis of the above calculations, the manufacturing office estimated the total number of specifications requests for 1983 for both Panther and Starling automobiles, and the personnel required to handle this work load as in Exhibit 3.

Design Parts Section (67 People)

The number of unique parts to be processed was used as the general work load determinant in

this section. In 1982, for 6,584 parts there were 27 specifications coordinators in the Starling Division budget for an average of 242 parts per coordinator. According to Exhibit 4, which was drawn up in the manufacturing office, 58 specifications coordinators would be required to handle the combined work load in 1983 plus 9 supervisors and clerical workers.

Planning and Control Section (15 People)

The requirements for this section were determined by the manufacturing office as shown in Exhibit 5, based on an overall work load indicator of number of parts to be handled.

Manager's Office (2 People)

A personnel ceiling of two was requested: the manager and his secretary.

EXHIBIT 1 Personnel Ceilings for Operations Control Department in 1983
(proposed by manufacturing office)

Positions	Starling Commitment 12/31/82	Starling Savings	Proposed Levels		
			Starling	Panther	Totals
Manager and secretary	2	2	—	2	2
Specifications control	20	12	8	17	25
Design parts control	31	12	19	48	67
Planning and control	7	3	4	11	15
Total	<u>60</u>	<u>29</u>	<u>31</u>	<u>78</u>	<u>109</u>

EXHIBIT 2 Relationship of Number of Parts to Specification Requests

Division	Number of Unique Parts	Number of Specifications Requests	Specifications Requests per Unique Part
Panther	8,810	3,680	0.42
Starling	<u>6,584</u>	<u>2,964</u>	<u>0.45</u>
Total	<u>15,394</u>	<u>6,644</u>	<u>0.43</u>

Assumpt.

EXHIBIT 3 Estimates of Personnel Requirements for Specifications Control Section, 1983**A. Specifications requests and equivalent personnel:**

Division	Estimated Number of Unique Parts*	×	Specifications Requests per Unique Part	÷	Actual Output per Worker	=	Equivalent Personnel
Panther	11,600		0.42		296		16.5
Starling	4,800		0.45		296		7.3
Total	16,400						23.8

B. Salaried personnel requirements:

Division	Equivalent Personnel	Less Planned Efficiency (approx: 10%)†	Less Planned Overtime (approx: 5%)	Salaried Ceiling Required
Panther	16.5	1.7	0.9	14
Starling	7.3	0.8	0.4	6
Total	23.8	2.5	1.3	20

C. Other personnel requirements:

Position	Panther	Starling	Total
Section supervisor and secretary	2	—	2
Unit supervisor	1	1	2
Clerk-typist	—	1	1
Total fixed	3	2	5†
Total salaried and other			25

* The Panther Division had added a new car to its line, and the Starling Division had dropped one.

† "Planned efficiency" reduces the calculated personnel requirements to a level approximately consistent with the lowest work load level anticipated during the coming year. In order to handle periodic work load increases during a year, the department is forced to improve its efficiency and, if necessary, to utilize overtime or temporary clerical help from outside agencies.

‡ Same as Starling commitment of December 31, 1982.

Estimated Dollar Requirements

The manufacturing office estimated that a total of \$2,916,000 would be needed to operate the consolidated operations control department for 1983. This figure was broken down as follows:

Personnel	\$2,320,000
Material and supplies	90,000
Computer services	490,000
Miscellaneous	16,000
Total	<u>\$2,916,000</u>

Personnel expenses. This estimate was based on the figure for actual salaries plus approved fringe benefits, in accordance with the level of requested salaried personnel ceilings.

Materials and supplies. This expense was about \$20,000 higher than the 1982 Starling actual. According to the manufacturing office, the job to be accomplished now was about two-and-one-half times the job accomplished by the Starling Division in 1982, but the expense was only 30 percent greater. This was a result of efficiencies in programming and reporting, which, in

EXHIBIT 4 Estimates of Personnel Requirements for Design Parts Section, 1983

A. Specifications coordinators requested:

<i>Division</i>	<i>1983 Parts Count</i>	<i>Estimated Output per Worker</i>	<i>Equivalent Personnel Required</i>	<i>Less Planned Efficiency (approx: 10%)</i>	<i>Less Planned Overtime (approx: 5%)</i>	<i>Ceiling Required</i>
Panther	11,600	242	48.0	4.7	2.3	41
Starling	4,800	242	19.8	1.9	.9	17
Total	<u>16,400</u>		<u>67.8</u>	<u>6.6</u>	<u>3.2</u>	<u>58</u>

B. Supervisory and clerical workers requested:

<i>Position</i>	<i>Panther</i>	<i>Starling</i>	<i>Total</i>
Section supervisor and secretary	2	—	2
Unit supervisors	2	1	3
Clerk-typists	3	1	4
Total	<u>7</u>	<u>2</u>	<u>9</u>
Unit supervisors to coordinators	1:20	1:17	1:19
Clerk-typists to coordinators	1:13	1:17	1:15
Total for the section			<u>67</u>

turn, would result in savings in materials and supplies.

Computer services. Starling had spent \$380,000 in 1982 to accomplish a job that was about 40 percent as great as the combined Starling-Panther job. Included in the proposed amount was \$68,000 for start-up cost associated with the conversion of the manual Panther system to a computerized system. Therefore, the real cost was \$422,000, or only about 10 percent more than the 1982 Starling actual. The manufacturing office was proposing to do a job 150 percent greater than done at Starling for only 10 percent more money. This was said to be the result of efficiencies in programming and reporting.

Analysis by the Controller's Office

The controller's office did not concur with the manufacturing office's proposal. It summarized

both the 1983 Panther Division's budget and the Starling Division's budget as approved prior to the consolidation, and compared these figures with those proposed by the manufacturing office. This summary is shown in Exhibit 6 and is explained in the following paragraphs.

Although the proposed combined Panther and Starling budgets for 1983 showed a decrease of 22 salaried personnel, there was an increase in cost of \$136,000.

The manufacturing office had referred to a saving of 24 people and \$368,000 in the Starling Division. This reduction, according to the controller, was the result of (a) a reduction in the 1983 parts count and (b) a reduction of supervisory and clerical personnel. This saving of 24 people, therefore, had nothing to do with computerization and would have occurred under either a computerized or a manual system.

Although the main reason for computerizing the Panther Division's system of parts control had

National Motors, Inc. 461

EXHIBIT 5 Estimate of Personnel Requirements for Planning and Control Section, 1983

Position	Number of Unique Parts for Starling, 1982		Parts per Person
	Starling Personnel		
Programming computer	3	6,584	2,195
Programming timing and coordination	2	6,584	3,292

	Number of Unique Parts, 1983		Estimated Output per Person	Equivalent Personnel		Personnel Ceiling Requested		
	Panther	Starling		Panther	Starling	Panther	Starling	Total
Program timing and coordination	11,600	4,800	3,292	3.5	1.5	4	2	6
Programming	11,600	4,800	2,195	5.3	2.2	5	2	7
Total				8.8	3.7	9	4	13
Section supervisor and secretary								2
Total for the section								15

EXHIBIT 6 Budget Comparison for Salaried Personnel Prepared by Controller

Budget Status	Panther Division		Starling Division		Total	
	Number	Dollars (000s)	Number	Dollars (000s)	Number	Dollars (000s)
Budget before consolidation	76	\$1,444	55*	1,336	131	\$2,780
Proposed	78	1,948	31	968	109	2,916
Net change	(2)	(504)	24	368	22	(136)
Explanation of changes:						
Savings from computerization of Panther system	23 †	276 †	—	—	23	276
Savings from consolidation	—	—	24	368	24	368
Proposed increase to Panther budget	(25)	(780)	—	—	(25)	(780)
Net change	(2)	\$ (504)	24	\$ 368	22	\$ (136)

() = Adverse effect on profit.

* Reflects the transfer of five specifications follow-up personnel out of the specifications control system.

† Based on study of December 1982, concurred in by manufacturing office.

EXHIBIT 7 Controller's Revised Budget Comparison for Salaried Personnel, 1983

System	Panther Division		Starling Division		Total	
	Number	Dollars (000s)	Number	Dollars (000s)	Number	Dollars (000s)
Combined manual systems	76	\$1,444	35	\$812	111	\$2,256
Proposed computerized systems	78	1,948	31	960	109	2,916
Difference between cost of computerized system and manual system	(2)	(504)	4	(156)	2	(660)

EXHIBIT 8 Controller's Proposed Budget, 1983

Salaried Personnel	Panther	Starling	Total
Number	53	31	84
Budget dollars (000s)	\$1,168	\$968	\$2,136

been financial savings, the controller calculated what the combined budget would have been if, in fact, the Starling Division's system had been changed to a manual one comparable to the one in use by the Panther Division prior to the consolidation. The budget requirement for the Panther Division, of course, would not change. However, 35 people and \$812,000 would be required for the Starling Division, on the basis of Panther Division's standards as developed in the manufacturing office's analysis. Thus, a comparison between the manual system and the computerized system was as shown in Exhibit 7.

According to Exhibit 7, the effect of the computerization and the consolidation on the 1983 Panther budget, which was based on a manual system, was to increase the 1983 salaried personnel level by two people and to increase costs by \$504,000. The controller was at a loss to know why these increases should result from computerization. Moreover, the manufacturing office had committed itself to a saving of 23 people and \$276,000 in the Panther Division, whereas the

current proposal was 25 people and \$780,000 over the levels committed.

The controller believed that budget figures under a *combined computerized system*, instead, should be as shown in Exhibit 8.

In this calculation, the Panther Division's number of salaried personnel was based on the pre-computerization figure (76) minus the saving agreed to by the manufacturing office as a result of computerization (23). The Panther Division's budget dollars were based on the same sort of analysis—\$1,444,000 minus \$276,000. Starling Division's figures were those used in Exhibit 7, based on a reduced parts count, supervisory savings, and the functional transfer of personnel. The budget figures for the new division should be 84 people and \$2,136,000.

On the basis of its analysis, the controller's office recommended that the manufacturing office at least not increase its 1983 costs for the operations control department over the level that would have occurred under a combined manual system. This meant a dollar budget of \$2,256,000. Per-

National Motors, Inc. 463

EXHIBIT 9 Detail of Controller's Recommended Budget, 1983

<i>Proposals</i>	<i>Panther</i>		<i>Starling</i>		<i>Total</i>	
	<i>Number</i>	<i>Dollars (000s)</i>	<i>Number</i>	<i>Dollars (000s)</i>	<i>Number</i>	<i>Dollars (000s)</i>
Manufacturing office's request	78	\$1,948	31	\$968	109	\$2,916
Controller's recommended reductions:						
Salary mix	—	170	—	—	—	170
Overtime	—	54	—	42	—	96
Required personnel (to meet financial objective)	25	394	—	—	25	394
Total recommended reductions	25	618	—	42	25	660
Total recommended level	53	1,330	31	926	84	2,256

EXHIBIT 10 Revised Estimates of Number of Unique Parts

<i>Division</i>	<i>1983 Original Budget Estimates</i>	<i>Current Known Conditions</i>
Panther	10,200	11,600
Starling	—	4,800
Total	10,200	16,400

sonnel reductions would be required to contain costs within recommended levels; these were set forth in Exhibit 9.

Protest from the Manufacturing Office

The manufacturing office did not accept the controller's recommendation of a reduction of 25 salaried people and \$660,000, though it agreed that, generally speaking, a computerized operations control system should not be any more costly than the previously used manual system.

Work-Load Content and Volume Adjustments

One of the arguments of the manufacturing office was that its proposed Starling-Panther budget included additional people to handle actual

work-load volume increases over the estimated levels used in developing the 1983 Panther budget for a manual system. The parts counts estimates used in developing the 1983 annual budget and the proposed consolidated computerized budget were as shown in Exhibit 10.

According to the manufacturing office, in addition to increased work as a result of the added work load of the Starling Division, there had been an increase of 1,400 parts in the Panther Division as a result of understated original estimates. This increased parts count would have resulted in a requirement for at least 10 more people under the manual system, at a cost of about \$180,000, plus an estimated \$8,000 for operating expenses.

EXHIBIT 11 Budget Increase Due to Salary Mix

<i>Salary Base</i>	<i>Proposed Ceiling</i>	<i>×</i>	<i>Average Annual Salary</i>	<i>=</i>	<i>Total Annual Salaries</i>
At approved budget rates	109		\$14,088		\$1,535,592
At proposed budget rates	<u>109</u>		<u>15,768</u>		<u>1,718,712</u>
Total:					<u>(\$ 183,120)</u>

Unavoidable Increases in Salary Mix

As a result of Starling-Panther consolidation and the consequent personnel changes, the average salary per employee retained in the operations control department had increased significantly. This resulted from the retention of employees on the basis of seniority. The approved budget provided for an average salary of \$14,088. The Starling-Panther budget, proposed by the manufacturing office for 1983 based on actual salaries, provided for an average annual salary in excess of \$15,600. Therefore, if average salaries had remained unchanged after the consolidation, the manufacturing office's budget proposal would have been \$183,120 less, as shown in Exhibit 11.

Association with Integrated Data Processing Plan

By implementing the computerized operations control system, the manufacturing office contended that it had taken an inevitable step included in the company's integrated data processing plan, which provided for eventual establishment of a completely computerized master parts control system. This step would make it possible to reduce significantly the original expense estimates associated with setting up this master system.

The original proposal, submitted prior to the consolidation of the two divisions, contained cost estimates of \$189,774 during 1983 and \$209,344

EXHIBIT 12 Effective Cost Decrease Due to Mechanization

<i>Revised Cost Factors</i>	<i>1983</i>	<i>1984 Going Level</i>
Original cost estimates	\$189,774	\$209,344
Cost of consolidation and revised assumptions based on manual system	<u>33,970</u>	<u>109,138</u>
Total cost estimates to include effect of consolidation based on a manual system	233,744	318,482
Reduction in cost estimates to give effect to consolidation	17,400	122,020
Savings directly associated with a computerized versus manual system	<u>203,344</u>	<u>196,462</u>

each year thereafter for providing a master parts control system to preproduction control. According to the manufacturing office, these cost estimates would have been increased to \$223,744 and \$318,482, respectively, as a result of the consolidation if a manual system were used. As a direct benefit of implementing a computerized operations control system, however, the manufacturing office believed that it could show a saving of about \$206,000 during 1983 and \$196,000 for each year thereafter. See Exhibit 12.

Nonrecurring Cost Penalties

The manufacturing office's proposed budget included a nonrecurring cost penalty of \$224,610, resulting from the change in organization and procedure. This was comprised of \$144,610 in salaries and wages and \$80,000 in computer expense. If work volume remained at the same levels in future years, the manufacturing office felt that its budget could be revised as shown in Exhibit 13.

Functional Improvements and Advantages

The manufacturing office contended, furthermore, that a computerized operations control sys-

tem offered certain other advantages over a manual system.

1. It provided a single and better integrated program progress report that reflected the status of engineering, manufacturing, and purchasing actions against schedules on a more timely basis than did a manual system.

2. It provided a master file that, once stored in the computer, could be used to produce other useful information.

3. It was compatible with the objective to mechanize the issuance of specifications and would result in a more efficient method of handling this activity. The manufacturing office said that it could not put a dollar value on these advantages, but that it was reasonable to expect them to yield cost savings.

Summary

A cost comparison for a manual versus a computerized operations control system, based on the above adjustments, was as shown in Exhibit 14.

The manufacturing office concluded its arguments by pointing out that the computerized system cost only \$82,000 a year more than a manual system, as shown in the preceding table, rather than \$660,000 more, as stated by the controller.

EXHIBIT 13 Future Savings of Nonrecurring Costs

<i>Budget Items</i>	<i>1983</i>	<i>Future Years</i>	<i>Reductions</i>
Average personnel ceiling	117	109	9
Personnel costs	\$2,317,752	\$2,173,142	\$144,610
Computer expense	490,000	410,000	80,000
Other operating costs	108,272	108,272	—
Total	<u>\$2,916,024</u>	<u>\$2,691,414</u>	<u>224,610</u>

APPENDIX B

National Motors Case Solution

<u>Sources of Discrepancy</u>		<u>One-Time Cost</u> (Stated in the Case, p. 465)	
One-Time Cost		Amount	
Salary Mix Change (Given on p. 464)		\$224,610	
Work Load Increase (plugged)		183,120	
Discrepancy (given)		252,270	
		<u>\$660,000</u>	
<u>Salary Mix Change</u>		<u># Personnel</u> <u>Total Salary</u>	
Average Salary 83		\$15,768	109 \$1,718,712
Average Salary 82		\$14,088	109 1,535,592
			<u>\$183,120</u>
<u>Work Load Increase</u>		<u>Personnel</u> <u>In Dollar Term</u>	
Difference in Estimation		25	
One-Time (\$144,610/15,600)		9	
Personnel Increase for Work Load		16	\$15,768 \$249,590

APPENDIX C

1 Main Reasoning Engine

```

:- consult(sys/control).
:- consult(sys/subtotal).
:- consult(sys/simplify).

%The request has been successfully processed if there is no more items in
the list
request([]):-
    write('no more items in this list'), nl, nl,
    append('log.txt'),
    write('Controller:      no more items in this list'), nl, nl,
    told.

request(-99999):-
    nl, write('****'), nl,
    write('The requested data is not available, Continue to perform the
next test. '), nl,
    append('log.txt'),
    write('Controller:      The requested data is not available,
Continue to perform the next test. '), nl,
    told,
    fail.

%Handling the request by starting from the first and process recursively
request([Head|Tail]):-
    nl, write('****'), nl,
    write('Checking the following item: '), write(Head), nl,
    append('log.txt'),
    nl, write('Controller:  ****'), nl,
    write('Controller:      Checking the following item: '),
write(Head), nl,
    told,
    approve(Head),
    request(Tail).

%The requested item is approved if it falls within the deviation limit
approve(account(Key, [Value|_], _, Desc)):-
    read_Val_Dev(Key, [MyVal], MyDev, _),
    Max is MyVal*(MyDev+1),
    Min is MyVal*(1-MyDev),
    Value<Max,
    Value>Min,
    write(Desc), write(' ( '), write(Key), write(')'), write(' = '),
write(Value), write(' --->'),
    write(' APPROVED'), nl, nl,
    append('log.txt'),

```

```

        write('Controller:      '), write(Desc), write(' ('), write(Key),
write(')'), write(' = '), write(Value), write(' --->'),
        write(' APPROVED'), nl, nl,
        told.

%reads in the value and deviation of an account stored in our database
read_Val_Dev(Key, [Value], Deviation, _):-
    account(Key, [Value|_], Deviation, _).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%If the requested item can be broken down in details, query the request
originator
%for detailed information by breaking the item down according to the
control hierarchy.
%Check the subsidiary accounts.
approve(Item):-
    subsidiary(Item),
    display_msg1(Item),
    request_subsidary(Item, Subsidiary_List),
    request(Subsidiary_List).

%Check to see if our database contains detailed subsidiary breakdowns
subsidiary(account(key(T, A, M, O), _, _, _)):-
    control(O, S),
    account(key(T, A, M, S), _, _, _),!.

%writing the subsidiary query command to a file
request_subsidary(Item, Return_List):-
    write_request(Item, Return_List, 'subsidiary_account').

%output request to file
write_request(account(Key, _, _, Desc), List, Function):-
    write('Writing command file to request detail of '), write(Desc),
write(' ('), write(Key), write(')'), nl,
    tell('request.pl'),
    write(':-'), write(Function), write('('), write(Key), write(')'),
nl,
    told,
    write('Please start another instance of prolog to generate
response. '), nl,
    write('When response is received, type 1 to continue the current
process. '), nl, nl,
    append('log.txt'),
    write('Controller:      Writing command file to request detail of
'), write('('), write(Key), write(')'), nl,
    write('Controller:      Please start another instance of prolog to
generate response. '), nl,
    write('Controller:      When response is received, type 1 to
continue the current process. '), nl, nl,
    told,
    read(Command),
    read_response(Command, List).

%get the response by reading the returned data file
read_response(1, Return_List):-
    see('response.pl'),
    read(Return_List),
    seen.

```

[illegible]

```

%If the requested item can be broken down into sub-accounts, query the
originator
%for sub-account information. Check the sub-accounts
approve(Item):-
    subaccount(Item),
    display_msg2(Item),
    request_subaccount(Item, Subaccount_List),
    request(Subaccount_List).

%Check to see if our database contains sub-accounts of the current item
subaccount(account(key(T, [A1, A2, A3, 0], M, 0), _, _, _)):-
    account(key(T, [A1, A2, A3, A4], M, 0), _, _, _),
    A4 > 0, !.
subaccount(account(key(T, [A1, A2, 0, 0], M, 0), _, _, _)):-
    account(key(T, [A1, A2, A3, 0], M, 0), _, _, _),
    A3 > 0, !.
subaccount(account(key(T, [A1, 0, 0, 0], M, 0), _, _, _)):-
    account(key(T, [A1, A2, 0, 0], M, 0), _, _, _),
    A2 > 0, !.

%Inform user about sending a query
display_msg2(account(Key, _, _, Desc)):-
    write('The requested item '),
    write(Desc), write(' ('), write(Key), write(')'),
    write(' exceeds limit. '), nl,
    write('Ask the originator to supply information on any categorical
sub-accounts. '), nl, nl,
    append('log.txt'),
    write('Controller:      The requested item '),
    write(Desc), write(' ('), write(Key), write(')'),
    write(' exceeds limit. '), nl,
    write('Controller:      Ask the originator to supply information on
any categorical sub-accounts. '), nl, nl,
    told.

%writing the subaccounts query command to a file
request_subaccount(Item, Return_List):-
    write_request(Item, Return_List, 'get_subaccount').

%collect subaccounts and store in a list
get_subaccount(Key):-
    subaccount(account(Key, _, _, _)),
    nl, write('Collecting categorical sub-accounts'), nl,
    write('Writing collected accounts to file'), nl, nl,
    append('log1.txt'),
    nl, write('Manufacturing:      Collecting categorical sub-
accounts'), nl,
    write('Manufacturing:      Writing collected accounts to file'),
nl, nl,
    told,
    bagof(Subaccount, find_subaccount(Key, Subaccount),
Subaccount_List),
    tell('response.pl'),
    writeq(Subaccount_List), write('.'), nl,
    told.

get_subaccount(_):-
    write('The requested data is not available. No data will be
provided. '), nl, nl,

```

```

tell('response.pl'),
write(-99999),
write('.'),
nl,
told,
append('log1.txt'),
write('Manufacturing:           The requested data is not available.
No data will be provided. '), nl, nl,
told.

%find sub-accounts
find_subaccount(key(T, [A, 0, 0, 0], M, O), account(key(T, [A, B, 0, 0],
M, O), [Head], Dev, Desc)):-
    account(key(T, [A, B, 0, 0], M, O), [Head|_], Dev, Desc),
    B > 0.

find_subaccount(key(T, [A, B, 0, 0], M, O), account(key(T, [A, B, C, 0],
M, O), [Head], Dev, Desc)):-
    account(key(T, [A, B, C, 0], M, O), [Head|_], Dev, Desc),
    C > 0.

find_subaccount(key(T, [A, B, C, 0], M, O), account(key(T, [A, B, C, D],
M, O), [Head], Dev, Desc)):-
    account(key(T, [A, B, C, D], M, O), [Head|_], Dev, Desc),
    D > 0.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% If the requested item is calculated according to some equation, ask the
% originator for the
% equation and compare it against our own
approve(Item):-
    equation_exist(Item),
    display_msg3(Item),
    request_equation(Item, Returned_Equation),
    check_equation(Item, Returned_Equation).

%Check to see if our database contains formula of the current item
equation_exist(account(key(T, A, M, O), _, _, _)):-
    equation(key(T, A, M, O), _).

%writing the formula query command to a file
request_equation(Item, Equation):-
    write_request(Item, Equation, 'get_equation').

get_equation(Key):-
    equation(Key, Equation),
    write('The requested equation is found. '), nl,
    write('Writing the requested equation to text file. '), nl, nl, nl,
    tell('response.pl'),
    write(Equation), write('.'), nl,
    told,
    append('log1.txt'),
    write('Manufacturing:           The requested equation is found. '),
nl,
    write('Manufacturing:           Writing the requested equation to text
file. '), nl, nl, nl,
    told.

```

```

get_equation(_):-
    write('The requested equation is not available.  No data will be
provided. '), nl, nl,
    tell('response.pl'),
    write(-99999),
    write('.'),
    nl,
    told,
    append('log1.txt'),
    write('Manufacturing:          The requested equation is not
available.  No data will be provided. '), nl, nl,
    told.

%Inform user about sending a query
display_msg3(account(Key, _, _, Desc)):-
    write('The requested item '),
    write(Desc), write(' ('), write(Key), write(')'),
    write(' exceeds limit. '), nl,
    write('Ask the originator to supply equation. '), nl, nl,
    append('log.txt'),
    write('Controller:          The requested item '),
    write(Desc), write(' ('), write(Key), write(')'),
    write(' exceeds limit. '), nl,
    write('Controller:          Ask the originator to supply equation. '),
nl, nl,
    told.

%check the returned formula against ours, if the formulae are the same,
check the components
check_equation(Item, Returned_Equation):-
    same_equation(Item, Returned_Equation, List),
    write('Equation checked, proceed to check the value of individual
component. '), nl,
    append('log.txt'),
    write('Controller:          ***'), nl,
    write('Controller:          Equation checked, proceed to check the value
of individual component. '), nl,
    told,
    get_c(List).

get_c([]).
get_c([Head|Tail]):-
    write_request(account(Head, _, _, _), ReturnItem, 'get_component'),
    request([ReturnItem]),
    get_c(Tail).

get_component(Key):-
    account(Key, Val, Dev, Desc),
    write('Manufacturing:          Writing the components of the
requested equation to file. '), nl, nl,
    append('log1.txt'),
    write('Manufacturing:          Writing the components of the
requested equation to file. '), nl, nl,
    told,
    tell('response.pl'),
    writeq(account(Key, Val, Dev, Desc)), write('.'), nl,
    told.

```



```

get_component(_):-
    write('The requested equation (component) is not available. No
data will be provided. '), nl, nl,
    append('log1.txt'),
    write('Manufacturing:          The requested equation (component) is
not available. No data will be provided. '), nl, nl,
    told,
    tell('response.pl'),
    write(-99999),
    write('.'),
    nl,
    told.

%if the equation used are not the same, inform user
check_equation(account(Key, _, _, _), Returned_Equation):-
    write('Different equations are used for this account ---> '), nl,
    write('Their equation: '), write(Returned_Equation), nl,
    write('Our equation: '), equation(Key, Equation), write(Equation),
nl,
    append('log.txt'),
    write('Controller:          Different equations are used for this
account ---> '), nl,
    write('Controller:          Their equation: '),
write(Returned_Equation), nl,
    write('Controller:          Our equation: '), equation(Key, Equation),
write(Equation), nl,
    told.

%Compare equations
same_equation(account(Key, _, _, _), Equation, List):-
    equation(Key, Our_Equation),
    simplify1(Our_Equation-Equation, 0),
    extract(Our_Equation, List).

% extract(+Expr, -AccList).
%extract(Alias, Key):-
%    alias(Alias, Key).
%extract(Alias, Key).

extract(A+B, List):- !,
    extract(A, List1),
    extract(B, List2),
    append(List1,List2,List).
extract(A-B, List):- !,
    extract(A, List1),
    extract(B, List2),
    append(List1,List2,List).
extract(A*B, List):- !,
    extract(A, List1),
    extract(B, List2),
    append(List1,List2,List).

extract(A/B, List):- !,
    extract(A, List1),
    extract(B, List2),
    append(List1,List2,List).
extract(-B, List):- !,
    extract(B, List).

```

```

extract(Expr, List):-
    atom(Expr), !,
    alias(Expr, Key),
    bagof(Key, alias(Expr, Key), List).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%if all the above tests failed, the item is not approved
%
approve(account(Key, [Val|_], _, Desc)):-
    write(Desc), write(' = '), write(Val), write(' ('), write(Key),
write(')'),
    write(' exceeds limit --->'),
    write(' NOT APPROVED'), nl, nl,
    append('log.txt'),
    write('Controller:      '),
    write(Desc), write(' = '), write(Val), write(' ('), write(Key),
write(')'),
    write(' exceeds limit --->'),
    write(' NOT APPROVED'), nl, nl,
    told.

```

2 Equation Simplification Engine

```

/*
ALGEBRA SIMPLIFIER

This file encloses a code SIM v5.5 for algebra calculation.
Please send me your comments and suggestions.

Note: The code is developed under SWI-Prolog 1.9.5, need to change
power/3 for
BIN-Prolog.

Copyright:
1, Using this file for commercial purpose without my permission
   is not allowed;
2. the file may be distributed freely.

Example:
*/

go:-
    expression(E), nl, pp(E),
    simplify(E, NewE), pp('=', NewE),
    fail.

expression(E):-
E = (a^2-b^2)/(a+b).
expression(E):-
E = 3*x*y+2*a*b-2*y*x-b*a*2.
expression(E):-
E = 0+1*a/1-b^0.
expression(E):-
E = (a-b)^5/(b-a)^3.
expression(E):-
E = (2*a+3*a*b)/a.
expression(E):-
E = 6/(a^2*b+6).
expression(E):-
E = (a*a-b*(2*a-b))/(a-b).
expression(E):-
E = a^3/(c*(b/c*a^(-2))).

gol:-
expression1(E1,E2),
simplify1(E1*E2,E3),
simplify(E3/E1, Enew),
    nl, pp('E1 =', E1), pp('E2 =', E2), pp('E3 = E1*E2 ='), pp(E3),
    pp('E3/E1 =', Enew),
    fail.

expression1(E1,E2):-
E1 = 3*a^3+b*c+4*a*b*c-2*c,
E2 = 5*a*a-b*c.
expression1(E1,E2):-
E1 = a*b*c*d-b*c^3+a^4*2-c^3*3,
E2 = 2*a*b^2+3*d^3*b*c.

```

```

/*-----*/
/*      SIM v5.5      */
/*      /*      made by      */
/*      /*      zhuhail@vax.sbu.ac.uk */
/*      /*      18/7/96      */
/*      /*      any comment welcome */
%-----for SWI-Prolog

power(A, B, AB):-
  AB is A^B.

sort_all(List, Lsor):-
  msort(List, Lsor).

rev_sort(List, Lsor):-
  sort(List, L1), reverse(L1, Lsor).
%-----
/*
%-----for BIN-Prolog
power(A, B, AB):- integer(A), integer(B), B > 0, !,
  pow(A, B, C), integer(C, AB).
power(A, B, AB):-
  pow(A, B, AB).

sort_all(List, Lsor):-
  prolog:merge_sort(<, List, Lsor).

rev_sort(List, Lsor):-
  prolog:merge_sort(>, List, Lsor).
%-----
*/

any_f(sin(_)).
any_f(cos(_)).
any_f(exp(_)).
any_f(log(_)).

simplify(E, NewE):-
  check(E),
  sim(E, NewE).

sim(E, NewE):- numberf(E, Num), !,
  NewE = Num.
sim(E, NewE):- atom_n_exp(E), !,
  NewE = E.
sim(E, NewE):-
  e2l(E, List),
  del_list(List, Ldel),
  l2e(Ldel, E1),
  minus_inv_b(Esigninv, E1),
  sim_again(E, Esigninv, NewE).

sim_again(E, E, NewE):- !,
  NewE = E.
sim_again(_, E, NewE):-
  sim(E, NewE).

simplify1(E, NewE):-
  check(E),

```

```

siml(E, NewE).

siml(E, NewE):-
  expand(E, E1),
  sim(E1, E2),
  siml_again(E, E2, NewE).
siml_again(E, E, NewE):- !,
  NewE = E.
siml_again(_, E, NewE):-
  siml(E, NewE).

e2l(E, Ldiv):-
  removep(E, Erem),
  minus_inv(Erem, Einv),
  e2ll(Einv, List),
  sum_plus_list(List, Lsum),
  factorise(Lsum, Lfac),
  sort_list(Lfac, Lsor),
  collect_like(Lsor, Lcol),
  simplify_division(Lcol, Ldiv).

factorise(List, Lfac):-
  prime(Prime),
  factor_list(List, Prime, Lfac).

prime(Prime):-
  Prime = [2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,
    83,89,97,101,103,107,109,113,127,131,137,139,149,151,157,163,
    167,173,179,181,191,193,197,199].

factor_list([], _, []).
factor_list([H|T], Prime, [H1|T1]):-
  f_list(H, Prime, H1),
  factor_list(T, Prime, T1).

f_list([Sum|T], Prime, [NewS|NewT]):-
  f_li(T, Sum, Prime, NewS, NewT).

f_li([], Sum, _, Sum, []).
f_li([H^M|T], Sum, Prime, NewS, [H1^M|T1]):- int_list(H), !,
  list_div_prime(Prime, H, H1, 1, S1),
  power(S1, M, SM),
  S2 is Sum*SM,
  f_li(T, S2, Prime, NewS, T1).
f_li([H|T], Sum, Prime, NewS, [H1|T1]):- int_list(H), !,
  list_div_prime(Prime, H, H1, Sum, S1),
  f_li(T, S1, Prime, NewS, T1).
f_li([H|T], Sum, Prime, NewS, [H|T1]):-
  f_li(T, Sum, Prime, NewS, T1).

int_list([]).
int_list([H|_]|T):-
  integer(H),
  int_list(T).

list_div_prime([], List, List, Sum, Sum).
list_div_prime([H|_], List, List, Sum, Sum):-
  big_than_list(List, H), !.
list_div_prime([H|T], List, NewL, Sum, NewS):-

```

```

list_d_p(List, H, L1, Sum, SumH),
list_div_prime(T, L1, NewL, SumH, NewS).

list_d_p(List, H, NewL, Sum, SumH):-
list_div_int(List, H, L1), !,
S1 is Sum*H,
list_d_p(L1, H, NewL, S1, SumH).
list_d_p(List, _, List, Sum, Sum).

big_than_list([[0|_] | T], P):- !,
big_than_list(T, P).
big_than_list([[H|_] | _], P):- H > 0, P > H, !.
big_than_list([[H|_] | _], P):- H < 0, P < H, !.
big_than_list([_|T], P):-
big_than_list(T, P).

list_div_int([], _, []).
list_div_int([[H|T] | TT], N, [[H1|T] | TT1]):-
real_int_div(H, N, H1),
list_div_int(TT, N, TT1).

real_int_div(A, B, C):-
D is A mod B,
D = 0,
C is A//B.

collect_like(List, Lcol):-
collect_like_term(List, Lcot),
collect_like_factor(Lcot, Lcof),
collect_again(List, Lcof, Lcol).

collect_again(L, L, Lcol):- !,
Lcol = L.
collect_again(_, L, Lcol):-
collect_like(L, Lcol).

sort_list([H|T], [H|Tsor]):-
sort_time_list(T, Tsot),
h2t(Tsot, T1),
sort_all(T1, T2),
h2t(Tsor, T2).

h2t([], []).
h2t([[H|Tt] | Tp], [Tt-H | Tnew]):-
h2t(Tp, Tnew).

sort_time_list([], []).
sort_time_list([H|T], [Hsot | Tsot]):-
sort_time(H, Hsot),
sort_time_list(T, Tsot).

sort_time([H|T], [H|Tsor]):-
sort_plus_list(T, Tsop),
sort_all(Tsop, Tsor).

sort_plus_list([], []).
sort_plus_list([H^M | T], [Hsol^M | Tsop]):- H = [_|_], !,
sort_list(H, Hsol),
sort_plus_list(T, Tsop).

```

```

sort_plus_list([H|T], [Hsol|Tsop]):- H = [_|_], !,
    sort_list(H, Hsol),
    sort_plus_list(T, Tsop).
sort_plus_list([H|T], [H|Tsop]):-
    sort_plus_list(T, Tsop).

minus_inv(A, B):- numberf(A, N), !,
    B = N.
minus_inv(A, B):- atom(A), !,
    B = A^1.
minus_inv(A, B):- min_inv(A, B), !.
minus_inv(A, B):- any_f(A), !,
    B = A^1.
minus_inv(A, A).

min_inv(A+B, AB):-
    minus_inv(A, A1),
    minus_inv(B, B1),
    AB = A1+B1.
min_inv(A-B, AB):- !,
    minus_inv(A, A1),
    minus_inv(B, B1),
    AB = A1+B1*(-1).
min_inv(-A, B):-
    minus_inv(A, A1),
    B = A1*(-1).
min_inv(A*B, AB):-
    minus_inv(A, A1),
    minus_inv(B, B1),
    AB = A1*B1.
min_inv(A/B^N, AB):- !,
    Nneg is -1*N,
    minus_inv(A, A1),
    minus_inv(B, B1),
    AB = A1*B1^Nneg.
min_inv(A/B, AB):-
    minus_inv(A, A1),
    minus_inv(B, B1),
    AB = A1*B1^(-1).
min_inv(A^M, B):- atom_num(A), !,
    B = A^M.
min_inv(A^M, B):- any_f(A), !,
    B = A^M.
min_inv(A^M, B):-
    minus_inv(A, A1),
    B = A1^M.

minus_inv_b(B, A):-
    min_inv_b(B, A) -> true
    ; B = A.

min_inv_b(AB, A+B):-
    minus_inv_b(A1, A),
    minus_inv_b(B1, B),
    AB = A1+B1.
min_inv_b(AB, A-B):- !,
    minus_inv_b(A1, A),
    minus_inv_b(B1, B),
    AB = A1-B1.

```

```

min_inv_b(B, -A):-
  minus_inv_b(A1, A),
  B = -A1.
min_inv_b(AB, A*B^(-1)):- !,
  minus_inv_b(A1, A),
  minus_inv_b(B1, B),
  AB = A1/B1.
min_inv_b(AB, B^(-1)*A):- !,
  minus_inv_b(A1, A),
  minus_inv_b(B1, B),
  AB = A1/B1.
min_inv_b(AB, A*B):-
  minus_inv_b(A1, A),
  minus_inv_b(B1, B),
  AB = A1*B1.
min_inv_b(B, A^(-1)):- !,
  minus_inv_b(A1, A),
  B = 1/A1.
min_inv_b(B, A^M):-
  minus_inv_b(A1, A),
  B = A1^M.

pp(A):-
  write(A),nl.
pp(A, B):-
  write(A),write(' '),write(B),nl.
pp(A, B, C):-
  write(A),write(' '),write(B),write(' '),write(C),nl.
pp(A, B, C, D):-
  write(A),write(' '),write(B),write(' '),write(C),write(' '),
  write(D),nl.

atom_num(A):- atom(A), !.
atom_num(A):- number(A).

zero(A):- number(A), A < 0.0000001, A > -0.0000001.

unit(A):- number(A), A < 1.0000001, A > 0.9999999.

not_deal(A):-
  atom_n_exp(A).
not_deal(A):-
  any_f(A).
not_deal(A^_):-
  any_f(A).

atom_n_exp(A^_):- atom_num(A), !.
atom_n_exp(A):- atom_num(A).

collect_like_factor(A, [Sum|Tnew]):- A = [Sum|T], !,
  collect_factor(T, Tnew).
collect_like_factor(A, A).

collect_factor([], []).
collect_factor([H|T], [Hnew|Tnew]):-
  collect_fact(H, Hnew),
  collect_factor(T, Tnew).

collect_fact([Sum|T], [Sum|Tnew]):-

```



```

collect_fac(T, Tnew).

collect_fac([], []).
collect_fac([H|T], [Hnew|Tnew]):-
    collect_like_factor(H, HH),
    collect_f(T, HH, Hnew, T1),
    collect_fac(T1, Tnew).

collect_f([], H, H, []).
collect_f([H1|Tt], H^M, Hnew, Tnew):-
    collect_like_factor(H1, HH),
    HH=H^N, !,
    MN is M+N,
    collect_f(Tt, H^MN, Hnew, Tnew).
collect_f([Th|Tt], H, Hnew, [Th|Tt1]):-
    collect_f(Tt, H, Hnew, Tt1).

check(X):- atom_num(X), !.
check(X):- var(X), !,
    pp('Encounter a variable!'),
    pp('(Begin with a lower case letter for elements in expression.)'),
    fail.
check(A+B):- !,
    check(A),
    check(B).
check(A-B):- !,
    check(A),
    check(B).
check(A*B):- !,
    check(A),
    check(B).
check(A/B):- !,
    check(A),
    check(B).
check(-B):- !,
    check(B).
check(A^B):- !,
    check(A),
    check_num1(A^B).
check(F):- any_f(F), !,
    arg(1, F, A),
    check(A).
check(X):-
    pp('Either a wrong expression or I do not like it:',X),
    fail.

check_num1(_^B):- numberf(B, _), !.
check_num1(A^B):-
    pp(B, 'must be a number for me to deal with', A^B),
    fail.

removep(X, T):- atom_n_exp(X), !,
    T = X.
removep(A+(B+C), T):- !,
    removep(A+B+C, T).
removep(A+(B-C), T):- !,
    removep(A+B-C, T).
removep(A-(B+C), T):- !,
    removep(A-B-C, T).

```

```

removep(A-(B-C),T):-!,
    removep(A-B+C,T).
removep(A+B,T):-!,
    removep(A,NewA),
    removep(B,NewB),
    T = NewA+NewB.
removep(A-B,T):-!,
    removep(A,NewA),
    removep(B,NewB),
    T = NewA-NewB.
removep(-B,T):-!,
    removep(B,NewB),
    T = -NewB.
removep(A/(B/C),T):-!,
    removep(A/B*C,T).
removep(A/(B*C),T):-!,
    removep(A/B/C,T).
removep(A/(-B),T):-!,
    removep(-A/B,T).
removep(A*(B/C),T):-!,
    removep(A*B/C,T).
removep(A*(B*C),T):-!,
    removep(A*B*C,T).
removep(A*(-B),T):-!,
    removep(-A*B,T).
removep(A/B,T):-!,
    removep(A,NewA),
    removep(B,NewB),
    T = NewA/NewB.
removep(A*B,T):-!,
    removep(A,NewA),
    removep(B,NewB),
    T = NewA*NewB.
removep((A^B)^C,T):-!,
    removep(A,NewA),
    removep(B,NewB),
    removep(C,NewC),
    T = NewA^(NewB*NewC).
removep(A^B,T):-!,
    removep(A,NewA),
    T = NewA^B.
removep(T,T).

simd([], []).
simd([H|T], Lsimd):-atom_num(H),!,
    Lsimd = [H|NewT],
    simd(T, NewT).
simd([H|T], [Hsimd|Tsimd]):-
    sim_div(H, Hsimd),
    simd(T, Tsimd).

sim_div([H|T], [H|NewT]):-
    simdivision(T, NewT).

simdivision([], []).
simdivision([H], NewL):-!,
    NewL=[H].
simdivision([H|T], NewL):-simdiv(H,T,ReduceL),!,
    simdivision(ReduceL, NewL).

```

```

simdivision([H|T],[H|NewT]):-
    simdivision(T,NewT).

simdiv(L1^M,[L2^N|T],New):- M > 0, N < 0,
    divide_list(L1,L2,Ldiv),
    Ldiv = [[A]],
    numberf(A,Num), !,
    MN is M+N,
    N1 is -1*N,
    New = [[[L1^MN,Num^N1],[0]]|T].
simdiv(L1^M,[L2^N|T],New):- M < 0, N > 0,
    divide_list(L1,L2,Ldiv),
    Ldiv = [[Num]],
    number(Num), !,
    MN is M+N,
    New = [[[L1^MN,Num^N],[0]]|T].
simdiv(A^(-1),[H|T],New):- divide_list(H,A,DivL), !,
    New = [DivL|T].
simdiv(A,[H^(-1)|T],New):- divide_list(A,H,DivL), !,
    New = [DivL|T].
simdiv(A^(-1),[H|T],New):- divide_list(A,H,DivL), !,
    New = [DivL^(-1)|T].
simdiv(A,[H^(-1)|T],New):- divide_list(H,A,DivL), !,
    New = [DivL^(-1)|T].
simdiv(A,[H|T],[H|NewT]):-
    simdiv(A,T,NewT).

simplify_division(List,Lsim):- List = [[_]], !,
    Lsim = List.
simplify_division(List,Lsim):- List = [_,[_,_]], !,
    Lsim = List.
simplify_division(List,Lsim):-
    simd1(List,List1),
    simd(List1,Lsim).

simd1([],[]).
simd1([H|T],[Hc|Tc]):-
    simd2(H,Hc),
    simd1(T,Tc).

simd2([],[]).
simd2([H|T],[H2|T2]):- not_deal(H), !,
    H2 = H,
    simd2(T,T2).
simd2([H^M|T],[H2|T2]):- number(M), !,
    simplify_division(H,H1),
    H2 = H1^M,
    simd2(T,T2).
simd2([H|T],[H2|T2]):-
    simplify_division(H,H2), !,
    simd2(T,T2).
simd2([H|T],[H2|T2]):-
    H2 = H,
    simd2(T,T2).

collect_like_term([H|T],[H|Tcol]):-
    collect1(T,List1),
    collect(List1,Tcol).

```

```

collect1([], []).
collect1([H|T], [Hc|Tc]):-
    collect2(H, Hc),
    collect1(T, Tc).

collect2([], []).
collect2([H|T], [H2|T2]):- not_deal(H), !,
    H2 = H,
    collect2(T, T2).
collect2([H^M|T], [H2|T2]):- !,
    collect_like_term(H, H1),
    H2 = H1^M,
    collect2(T, T2).
collect2([H|T], [H2|T2]):-
    collect_like_term(H, H2), !,
    collect2(T, T2).
collect2([H|T], [H2|T2]):-
    H2 = H,
    collect2(T, T2).

collect([], []).
collect([H0|List1], [NewH|NewT]):-
    H0=[Para0|List0],
    coll(List1, Para0, List0, List2, NewPara),
    NewH=[NewPara|List0],
    collect(List2, NewT).

coll([], Para0, _, [], Para0).
coll([H1|T1], Para0, List0, T2, NewPara):- H1=[Para1|List1], List0 = List1,
    !,
    NewP is Para0+Para1,
    coll(T1, NewP, List0, T2, NewPara).
coll([H1|T1], Para0, List0, [H1|T2], NewPara):-
    coll(T1, Para0, List0, T2, NewPara).

sum_plus_list(A, B):- any_f(A), !,
    B = A.
sum_plus_list(A, B):- atom_num(A), !,
    B = A.
sum_plus_list(List, Lnew):-
    sum_plus(List, Sum, Rest),
    L1 = [[Sum]|Rest],
    sum_again(List, L1, Lnew).

sum_again(L, L, Lnew):- !,
    Lnew = L.
sum_again(_, L, Lnew):-
    sum_plus_list(L, Lnew).

sum_plus(List, Sum, Rest):- sum_p(List, 0, Sum, Rest).

sum_p([], SoFar, SoFar, []).
sum_p([[H]|T], SoFar, Sum, Rest):- numberf(H, NewH), !,
    New is SoFar+NewH,
    sum_p(T, New, Sum, Rest).
sum_p([H|T], SoFar, Sum, Rest):-
    sumt(H, Htime),
    sump_or_not(Htime, T, SoFar, Sum, Rest).

```

```

sump_or_not(Htime,T,SoFar,Sum,Rest):- numberf(Htime, NewH), !,
    New is SoFar+NewH,
    sum_p(T,New,Sum,Rest).
sump_or_not(Htime,T,SoFar,Sum,Rest):-
    Rest = [Htime|RestT],
    sum_p(T,SoFar,Sum,RestT).

sumt(List,NewList):-
    sum_time(List,Sum,Rest),
    NewList = [Sum|Rest].

sum_time(List,Sum,Rest):-
    sum_t(List,1,Sum,Rest).

sum_t([],SoFar,SoFar,[]).
sum_t([H|T],SoFar,Sum,Rest):- numberf(H, NewH), !,
    New is SoFar*NewH,
    sum_t(T,New,Sum,Rest).
sum_t([H|T],SoFar,Sum,Rest):- not_deal(H), !,
    Rest=[H|RestT],
    sum_t(T,SoFar,Sum,RestT).
sum_t([H^M|T],SoFar,Sum,Rest):- !,
    sum_plus_list(H,Hplus),
    sumt_or_not(Hplus^M,T,SoFar,Sum,Rest).
sum_t([H|T],SoFar,Sum,Rest):-
    sum_plus_list(H,Hplus),
    sumt_or_not(Hplus,T,SoFar,Sum,Rest).

sumt_or_not(Hplus,T,SoFar,Sum,Rest):- numberf(Hplus, NewH), !,
    New is SoFar+NewH,
    sum_t(T,New,Sum,Rest).
sumt_or_not(Hplus,T,SoFar,Sum,Rest):-
    Rest = [Hplus|RestT],
    sum_t(T,SoFar,Sum,RestT).

expand(E, Eexp):-
    timeinto(E, Etime),
    expand_again(E, Etime, Eexp).

expand_again(E, E, Eexp):- !,
    Eexp = E.
expand_again(_, E, Eexp):-
    expand(E, Eexp).

timeinto(A * (B + C), ABC):- !,
    ABC = A*B+A*C.
timeinto((B + C) * A, ABC):- !,
    ABC = B*A+C*A.
timeinto(A * (B - C), ABC):- !,
    ABC = A*B-A*C.
timeinto((B - C) * A, ABC):- !,
    ABC = B*A-C*A.
timeinto(A + B, AB):- !,
    timeinto(A, A1),
    timeinto(B, B1),
    AB = A1 + B1.
timeinto(A - B, AB):- !,
    timeinto(A, A1),
    timeinto(B, B1),

```

```

AB = A1 - B1.
timeinto(- B, AB):- !,
  timeinto(B, B1),
  AB = - B1.
timeinto(A * B, AB):- !,
  timeinto(A, A1),
  timeinto(B, B1),
  AB = A1 * B1.
timeinto(A / B, AB):- !,
  timeinto(A, A1),
  timeinto(B, B1),
  AB = A1 / B1.
timeinto(F, NewF):- any_f(F), !,
  arg(1, F, A),
  timeinto(A, NewA),
  replace(NewA, F, NewF).
timeinto(A, A).

numberf(A, N):- number(A), !,
  N is A.
numberf(F, N):- any_f(F), !,
  arg(1, F, A),
  numberf(A, NA),
  replace(NA, F, NewF),
  N is NewF.
numberf(A, N):-
  numf(A, N).

numf(A+B, N):-
  numberf(A, NA),
  numberf(B, NB),
  N is NA+NB.
numf(A-B, N):- !,
  numberf(A, NA),
  numberf(B, NB),
  N is NA-NB.
numf(-B, N):-
  numberf(B, NB),
  N is -NB.
numf(A*B, N):-
  numberf(A, NA),
  numberf(B, NB),
  N is NA*NB.
numf(A/B, N):-
  numberf(A, NA),
  numberf(B, NB),
  div_to_int(NA, NB, N).
numf(A^B, N):-
  numberf(A, NA),
  numberf(B, NB),
  power(NA, NB, N).

divide_list(List, List, Div):- !,
  Div = 1.
divide_list(List1, List2, Div):-
  deal_list(List1, Ldeal1),
  deal_list(List2, Ldeal2),
  (divide(Ldeal1, Ldeal2, L) -> true
   ; list_time_into(Ldeal1, Ltimel),

```

```

        list_time_into(Ldeal2, Ltime2),
        divide(Ltime1,Ltime2,L)),
    sum_plus_list(L,Lsum),
    sort_list(Lsum, Div).

deal_list(A, Ldeal):- atom(A), !,
    Ldeal = [[0],[1,A^1]].
deal_list(List, List).

list_time_into(List, Ltime):-
    l2e(List, E),
    sim1(E, Esim1),
    e2l(Esim1, Ltime).

rev_sort_list(List, Lnew):-
    rev_time(List, L1),
    rev_sort(L1, L2),
    t2h(L2, Lnew).

rev_time([], []).
rev_time([H|T], [Hrev|Trev]):-
    rev_sort(H, Hrev),
    rev_time(T, Trev).

t2h([], []).
t2h([H|T], [Ht2h|Tt2h]):-
    t2h1(H, Ht2h),
    t2h(T, Tt2h).

t2h1(List, [Last|Rest]):- t2h1(List, Last, Rest).

t2h1([Last], L, R):- !, L = Last, R = [].
t2h1([H|T], L, [H|Tr]):-
    t2h1(T, L, Tr).

divide(List1,List2,L):-
    rev_sort_list(List1,Lsor1),
    rev_sort_list(List2,Lsor2),
    div(Lsor1,Lsor2,L).

div([[A]],_,T):- zero(A), !, T = [].
div(List1,List2,[H3|T3]):-
    div1(List1,List2,H3),
    H3=[H3H|H3T],
    H3negH is -1*H3H,
    H3neg=[H3negH|H3T],
    time_list2(List2,H3neg,L),
    append(List1,L,AppL),
    sum_plus_list(AppL,SumpL),
    sort_list(SumpL, Lsort),
    collect_like(Lsort,Lcoll),
    zero_term(Lcoll, Lrem),
    rev_sort_list(Lrem, Lsorti),
    div(Lsorti,List2,T3).

div1([H1|_],[H2|_],H3):-
    dd(H1,H2,H3).

dd([H1|T1],[H2|T2],[H3|T3]):-

```

```

div_to_int(H1, H2, H3),
d(T2,T1,T3).

d([],List1,List1).
d([H2|T2],List1,Ldivision):-
  d1(List1,H2,NewL1),
  NewL1 \= List1,
  d(T2,NewL1,Ldivision).

d1([],_,[]).
d1([A^M|T],A^N,Lnew):- N >= 0, M > N, !,
  MN is M-N,
  Lnew = [A^MN|Tnew],
  d1(T,A^N,Tnew).
d1([A|T],A,Lnew):- !,
  d1(T,A,Lnew).
d1([H|T],A,[H|NewT]):-
  d1(T,A,NewT).

int_to_int(A, B, C):-
  integer(A),
  integer(B),
  real_int_div(A, B, C).

div_to_int(A, B, C):-
  int_to_int(A, B, C), !.
div_to_int(A, B, C):-
  C is A/B.

time_list2([],_,[]).
time_list2([H2|T2],H3,[H|T]):- number(H2), !,
  find_h(H2,H3,H),
  time_list2(T2,H3,T).
time_list2([H2|T2],H3,[H|T]):-
  append(H3,H2,AppH),
  sumt(AppH,H),
  time_list2(T2,H3,T).

find_h(H2,[H3|[]],H):- !,
  H is H2*H3.
find_h(H2,[H3|T3],[H|T3]):-
  H is H2*H3.

zero_term([H|T],[H|NewT]):-
  zero_term1(T,NewT).

zero_term1([],[]).
zero_term1([A|_]T,NewL):- zero(A), !,
  zero_term1(T,NewL).
zero_term1([H|T],[H|NewT]):-
  zero_term1(T,NewT).

del_list(List, Ldel):-
  del_plus_list(List, Ldelp),
  reduce_plus_list(Ldelp, Ldel).

del_plus_list(L1, L2):-
  del_plus(L1, L3),
  (L3 = [] -> L2 = [[0]]; L2 = L3).

```



```

del_plus([], L):- !, L = [].
del_plus([[A|_] | T], Ldelp):- zero(A), !,
    del_plus(T, Ldelp).
del_plus([[A] | T], Ldelp):- number(A), !,
    Ldelp = [[A] | NewT],
    del_plus(T, NewT).
del_plus([H | T], [NewH | NewT]):- !,
    del_time_list(H, NewH),
    del_plus(T, NewT).
del_plus(A, A).

del_time_list(List, Lnew):-
    del_time(List, Ldel),
    (Ldel = [] -> Lnew = [1]; Lnew = Ldel).

del_time([], []).
del_time([H | T], Ldelt):- unit(H), !,
    del_time(T, Ldelt).
del_time([_ ^ M | T], Ldelt):- zero(M), !,
    del_time(T, Ldelt).
del_time([H ^ M | T], [NewH | NewT]):- unit(M), not_deal(H), !,
    NewH = H,
    del_time(T, NewT).
del_time([H ^ M | T], [NewH | NewT]):- !,
    del_plus_list(H, H1),
    NewH = H1 ^ M,
    del_time(T, NewT).
del_time([H | T], [NewH | NewT]):-
    del_plus_list(H, NewH),
    del_time(T, NewT).

reduce_plus_list([], L):- !, L = [].
reduce_plus_list([[ -1, A] | T], Lred):- A = [_ | _], !,
    insert_neg(A, Lred, NewT),
    reduce_plus_list(T, NewT).
reduce_plus_list([[A] | T], Lred):- A = [_ | _], !,
    append(A, NewT, Lred),
    reduce_plus_list(T, NewT).
reduce_plus_list([H | T], [NewH | NewT]):- !,
    reduce_time_list(H, NewH),
    reduce_plus_list(T, NewT).
reduce_plus_list(A, A).

reduce_time_list([], []).
reduce_time_list([[A] | T], Lredt):- !,
    append(A, NewT, Lredt),
    reduce_time_list(T, NewT).
reduce_time_list([H | T], [NewH | NewT]):- not_deal(H), !,
    NewH = H,
    reduce_time_list(T, NewT).
reduce_time_list([H | T], [NewH | NewT]):-
    reduce_plus_list(H, NewH),
    reduce_time_list(T, NewT).

insert_neg([], Tneg, Tneg).
insert_neg([H | T], [Hinv | Tinv], NewT):-
    neg(H, Hinv),
    insert_neg(T, Tinv, NewT).

```

```

neg([-1, A], B):- !,
    B = [A].
neg([-1|A], B):- !,
    B = A.
neg(A, [-1|A]).

e2l1(E, List):-
    e2_plus_list(E, [], List).

e2_plus_list(A+B, Sofar, List):- !,
    e2_plus_list(B, Sofar, ListB),
    e2_plus_list(A, ListB, List).
e2_plus_list(A, Sofar, List):-
    e2_time_list(A, ListA),
    List = [ListA|Sofar].

e2_time_list(E, List):- e2_time(E, [], List).

e2_time(A*B, Sofar, List):- !,
    e2_time(B, Sofar, ListB),
    e2_time(A, ListB, List).
e2_time(A, Sofar, List):- atom_num(A), !,
    List = [A|Sofar].
e2_time(A, Sofar, List):- any_f(A, A1), !,
    List = [A1|Sofar].
e2_time(A^M, Sofar, List):- any_f(A, A1), !,
    List = [A1^M|Sofar].
e2_time(A^M, Sofar, List):- !,
    e2l1_or_not(A, ListA),
    List = [ListA^M|Sofar].
e2_time(A, Sofar, List):-
    e2l1_or_not(A, ListA),
    List = [ListA|Sofar].

any_f(F, F1):-
    any_f(F),
    arg(1, F, A),
    sim(A, A1),
    replace(A1, F, F1).

replace(New, T1, T2):-
    functor(T1, Name, 1),
    functor(T2, Name, 1),
    arg(1, T2, New).

e2l1_or_not(E, LN):- atom_num(E), !,
    LN = E.
e2l1_or_not(E, LN):- any_f(E), !,
    LN = E.
e2l1_or_not(E, LN):-
    e2l1(E, LN).

plus_or_minus(Sofar, ET, T, E, EThead):- numberf(EThead, Num), Num < 0, !,
    l2el(T, Sofar-ET, E).
plus_or_minus(Sofar, ET, T, E, _):-
    l2el(T, Sofar+ET, E).

positive_or_negative(ET, T, E, EThead):-

```

```

numberf(EThead,Num), Num < 0, numberf(ET,En), !,
ETneg is -En,
l2e1(T, ETneg, E).
positive_or_negative(ET, T, E, EThead):-
numberf(EThead,Num), Num < 0, !,
ETneg = -ET,
l2e1(T, ETneg, E).
positive_or_negative(ET, T, E, _):-
l2e1(T, ET, E).

l2e(List^M, E):- List = [_|_], !, E = E1^M,
l2e(List, E1).
l2e([H|T], E):- !,
l2eT(H,ET,EThead),
positive_or_negative(ET, T, E, EThead).
l2e(A, A).

l2e1([], Sofar, Sofar).
l2e1([H|T], Sofar, E):-
l2eT(H,ET,EThead),
plus_or_minus(Sofar,ET,T,E,EThead).

l2eT([H|T], E, H):- numberf(H, Hn), Hn < 0, !,
Hpositive is -1*Hn,
l2eT1(T, Hpositive, E).
l2eT([H|T], E, H):- l2e(H, He),
l2eT1(T, He, E).

l2eT1([], Sofar, Sofar).
l2eT1([H|T], Sofar, E):-
l2e(H, He),
(unit(Sofar) -> l2eT1(T, He, E)
; l2eT1(T, Sofar*He, E)).

```

3 Divisional Subtotal

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%           Divisional Subtotal           %
%                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
find_Val(key(T, A, M, O), Val):-
    control(O, S),
    account(key(T, A, M, S), [Val|_], _, _).

sum_list(0, [ ]).
sum_list(Total, [H|T]):-
    sum_list(NTotal, T),
    Total is NTotal+H.

```

4 Corporate Hierarchy

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%           Superior and Subordinate Relationship           %
%                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% CONTROL(SUPERIOR, SUBSIDIARY)

direct_control(manufacturing, panther).
direct_control(manufacturing, starling).

control(Superior, Subsidiary):-
    direct_control(Superior, Subsidiary).

control(Superior, Subsidiary):-
    direct_control(Superior, Intermediate),
    control(Intermediate, Subsidiary).

```

5 Controller's Database

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%      Controller's Database      %
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% ACCOUNT(KEY(AACCOUNT#, MEASUREMENT, TYPE, OWNER), VALUE,
DEVIATION_ALLOWANCE)

%ESTIMATED MANUFACTURING BUDGET IN PERSONNEL
account(key(exp, [1, 1, 0, 0], person, panther), [53, 76], 0.10, 'Panther
personnel'). %panther personnel

account(key(exp, [1, 1, 0, 0], person, starling), [31, 55], 0.10,
'Starling personnel'). %Starling personnel

account(key(exp, [1, 1, 0, 0], person, manufacturing), [84, 131], 0.10,
'Manufacturing personnel'). %manufacturing personnel - horizontal sum

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%ESTIMATED MANUFACTURING BUDGET IN DOLLAR
account(key(exp, [1, 0, 0, 0], dollar, panther), [1330000, 1168000,
1444000], 0.10, 'Panther budget'). %panther

account(key(exp, [1, 0, 0, 0], dollar, starling), [926000, 968000,
1336000], 0.10, 'Starling budget'). %Starling
account(key(exp, [1, 0, 0, 0], dollar, manufacturing), [2256000, 2136000,
2780000], 0.10, 'Total, manufacturing budget'). %manufacturing -
horizontal sum

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%TOTAL ESTIMATED BUDGET IN DOLLAR
account(key(exp, [1, 1, 0, 0], dollar, manufacturing), [1740000], 0.10,
'Manufacturing personnel expenses'). %manufacturing personnel

account(key(exp, [1, 2, 0, 0], dollar, manufacturing), [90000], 0.10,
'Manufacturing material and supplies expense'). %manufacturing material
and supplies
account(key(exp, [1, 3, 0, 0], dollar, manufacturing), [410000], 0.10,
'Manufacturing computer services expenses'). %manufacturing computer
services
account(key(exp, [1, 3, 1, 0], dollar, manufacturing), [410000], 0.10,
'Regular services'). %regular services
account(key(exp, [1, 4, 0, 0], dollar, manufacturing), [16000], 0.10,
'Manufacturing miscellaneous expenses'). %manufacturing miscellaneous

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%TOTAL ESTIMATED PERSONNEL EXPENSES
account(key(exp, [1, 1, 2, 0], dollar, manufacturing), [1535592], 0.10,
'Salary based on 1982 average'). %salary based on average
account(key(exp, [1, 1, 3, 0], dollar, manufacturing), [204408], 0.10,
'Other expenses'). %other

```

```

account(key(exp, [1, 1, 3, 1], dollar, manufacturing), [204408], 0.10,
'Misc. expenses '). %other

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%FORMULA TO CALCULATE AVERAGE SALARY
equation(key(exp, [1, 1, 2, 0], dollar, manufacturing),
ave_salary*man_personnel).
alias(man_personnel, key(exp, [1, 9, 8, 8], person, manufacturing)).
alias(ave_salary, key(exp, [1, 9, 9, 9], dollar, manufacturing)).
account(key(exp, [1, 9, 9, 9], dollar, manufacturing), [14088], 0.10,
'1982 average salary'). %average salary

account(key(exp, [1, 9, 8, 8], person, manufacturing), [109], 0.10,
'Number of personnel'). %Number of personnel for sensitivity analysis

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%MANUFACTURING PARTS COUNT
account(key(asset, [2, 0, 0, 0], parts, panther), [10200], 0.10, 'Panther
parts count'). %panther parts count

account(key(asset, [2, 0, 0, 0], parts, starling), [0], 0.10, 'Starling
parts count'). %starling parts count

% TOTAL OF THE ABOVE ITEMS IS USED TO CALCULATE TOTAL PERSONNEL EXPENSE
account(key(asset, [2, 0, 0, 0], parts, manufacturing), [10200], 0.10,
'Manufacturing parts count'). %manufacturing parts count

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Department Subtotal%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%sum of same account at a higher level of the organizational hierarchy
account(key(T, A, M, O), Total, _, _):-
    bagof(Val, find_Val(key(T, A, M, O), Val), L),
    sum_list(Total, L).

```

6 Manufacturing Office's Database

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%           Manufacturing's Database           %
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% ACCOUNT(KEY(TYPE, ACCOUNT#, MEASUREMENT, OWNER), VALUE,
DEVIATION_ALLOWANCE, DESCRIPTION)

%STARLING PERSONNEL
account(key(exp, [1, 1, 2, 0], person, starling), [8], 0.10, 'Starling
spec control personnel'). %Starling spec control
account(key(exp, [1, 1, 3, 0], person, starling), [19], 0.10, 'Starling
design parts control personnel'). %Starling design parts control
account(key(exp, [1, 1, 4, 0], person, starling, 'Starling planning and
control personnnel'), [4], 0.10). %Starling planning and control

%SUBTOTAL OF THE ABOVE STARLING ACCOUNTS
account(key(exp, [1, 1, 0, 0], person, starling), [31], 0.10, 'Starling
total personnel'). %Starling total personnel

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%PANTHER PERSONNEL
account(key(exp, [1, 1, 1, 0], person, panther), [2], 0.10, 'Panther
manager and secretary'). %panther manager and secretary

account(key(exp, [1, 1, 2, 0], person, panther), [17], 0.10, 'Panther
Spec Control personnel'). %panther spec control

account(key(exp, [1, 1, 3, 0], person, panther), [48], 0.10, 'Pather
design personnel'). %panther design parts control

account(key(exp, [1, 1, 4, 0], person, panther), [11], 0.10, 'Panther
planning and control'). %panther planning and control

%SUBTOTAL OF THE ABOVE PANTHER ACCOUNTS
account(key(exp, [1, 1, 0, 0], person, panther), [78], 0.10, 'Panther
total personnel'). %Panther total personnel

account(key(exp, [1, 1, 0, 0], person, manufacturing), [109], 0.10,
'Manufacturing total personnel'). %Manufacturing total personnel
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%MANUFACTURING TOTAL BUDGET IN DOLLAR AMOUNT
account(key(exp, [1, 1, 0, 0], dollar, manufacturing), [2320000], 0.10,
'Manufacturing personnel expense'). %manufacturing personnel

account(key(exp, [1, 2, 0, 0], dollar, manufacturing), [90000], 0.10,
'Manufacturing material and supplies'). %manufacturing material and
supplies

account(key(exp, [1, 3, 0, 0], dollar, manufacturing), [490000], 0.10,
'Manufacturing computer services'). %manufacturing computer services

```

```

account(key(exp, [1, 4, 0, 0], dollar, manufacturing), [16000], 0.10,
'Manufacturing miscellaneous expense'). %manufacturing miscellaneous

%TOTAL OF THE ABOVE MANUFACTURING ACCOUNTS - Total Dollar Budget
account(key(exp, [1, 0, 0, 0], dollar, manufacturing), [2916000], 0.10,
'Manufacturing total budget'). %manufacturing total budget

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%MANUFACTURING TOTAL PERSONNEL IN DOLLAR AMOUNT
account(key(exp, [1, 1, 1, 0], dollar, manufacturing), [144610], 0.10,
'One-time personnel cost'). %one-time personnel cost

account(key(exp, [1, 1, 2, 0], dollar, manufacturing), [1718712,
1535592], 0.10, 'Salary based on average salary 1983'). %salary based on
average

account(key(exp, [1, 1, 3, 0], dollar, manufacturing), [456678], 0.10,
'Other expenses'). %other
account(key(exp, [1, 1, 3, 1], dollar, manufacturing), [204408], 0.10,
'Misc. expenses'). %other
account(key(exp, [1, 1, 3, 2], dollar, manufacturing), [252270], 0.10,
'Workload increase'). %other

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%FORMULA TO CALCULATE AVERAGE SALARY
equation(key(exp, [1, 1, 2, 0], dollar, manufacturing),
man_personnel*ave_salary).
%alias(man_personnel, key(exp, [1, 1, 0, 0], person, manufacturing)).
alias(man_personnel, key(exp, [1, 9, 8, 8], person, manufacturing)).
alias(ave_salary, key(exp, [1, 9, 9, 9], dollar, manufacturing)).

account(key(exp, [1, 9, 9, 9], dollar, manufacturing), [15768, 14088],
0.10, '1983 Average salary'). %average salary

account(key(exp, [1, 9, 8, 8], person, manufacturing), [109], 0.10,
'Number of personnel'). %Number of personnel for sensitivity analysis

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%MANUFACTURING COMPUTER SERVICES IN DOLLAR AMOUNT
account(key(exp, [1, 3, 1, 0], dollar, manufacturing), [410000], 0.10,
'Regular services'). %regular services
account(key(exp, [1, 3, 2, 0], dollar, manufacturing), [80000], 0.10,
'One time computer cost'). %ONE-TIME CHARGE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%MANUFACTURING PARTS COUNT
account(key(exp, [2, 0, 0, 0], parts, panther), [11600, 10200], 0.10,
'Panther parts count'). %panther parts count
account(key(exp, [2, 0, 0, 0], parts, starling), [4800], 0.10, 'Starling
parts count'). %starling parts count

% TOTAL OF THE ABOVE ITEMS IS USED TO CALCULATE TOTAL PARTS
account(key(exp, [2, 0, 0, 0], parts, manufacturing), [16400, 10200],
0.10, 'Manufacturing parts count'). %manufacturing parts count

```



```
%%%%%%%%Department Subtotal%%%%%%%%
%%%sum of same account at a higher level of the organizational hierarchy
account(key(T, A, M, O), Total, _, _):-
    bagof(Val, find_Val(key(T, A, M, O), Val), L),
    sum_list(Total, L).
```

APPENDIX D

1 Test Result

Controller: ***

Controller: Checking the following item: account(key(exp, [1,0,0,0], dollar, manufacturing), [2916000], 0.100000, Manufacturing total budget)

Controller: The requested item Manufacturing total budget (key(exp, [1,0,0,0], dollar, manufacturing)) exceeds limit.

Controller: Ask the originator to supply information on subsidiary divisions.

Controller: Writing command file to request detail of (key(exp, [1,0,0,0], dollar, manufacturing)).

Controller: Please start another instance of prolog to generate response.

Controller: When response is received, type 1 to continue the current process.

Manufacturing: The requested data is not available.
No data will be provided.

Controller: The requested data is not available, Continue to perform the next test.

Controller: The requested item Manufacturing total budget (key(exp, [1,0,0,0], dollar, manufacturing)) exceeds limit.

Controller: Ask the originator to supply information on any categorical sub-accounts.

Controller: Writing command file to request detail of
(key(exp, [1,0,0,0], dollar, manufacturing)).

Controller: Please start another instance of prolog to
generate response.

Controller: When response is received, type 1 to continue
the current process.

Manufacturing: Collecting categorical sub-accounts

Manufacturing: Writing collected accounts to file

Controller: ***

Controller: Checking the following item: account(key(exp,
[1,1,0,0], dollar, manufacturing), [2320000], 0.100000,
Manufacturing personnel expense)

Controller: The requested item Manufacturing personnel
expense (key(exp, [1,1,0,0], dollar, manufacturing)) exceeds
limit.

Controller: Ask the originator to supply information on any
categorical sub-accounts.

Controller: Writing command file to request detail of
(key(exp, [1,1,0,0], dollar, manufacturing)).

Controller: Please start another instance of prolog to
generate response.

Controller: When response is received, type 1 to continue
the current process.

Manufacturing: Collecting categorical sub-accounts

Manufacturing: Writing collected accounts to file

Controller: ***

Controller: Checking the following item: account(key(exp, [1,1,1,0], dollar, manufacturing), [144610], 0.100000, One-time personnel cost)

Controller: One-time personnel cost = 144610 (key(exp, [1,1,1,0], dollar, manufacturing)) exceeds limit ---> NOT APPROVED

Controller: ***

Controller: Checking the following item: account(key(exp, [1,1,2,0], dollar, manufacturing), [1718712], 0.100000, Salary based on avarage salary 1983)

Controller: The requested item Salary based on avarage salary 1983 (key(exp, [1,1,2,0], dollar, manufacturing)) exceeds limit.

Controller: Ask the originator to supply equation.

Controller: Writing command file to request detail of (key(exp, [1,1,2,0], dollar, manufacturing)).

Controller: Please start another instance of prolog to generate response.

Controller: When response is received, type 1 to continue the current process.

Manufacturing: The requested equation is found.

Manufacturing: Writing the requested equation to text file.

Controller: ***

Controller: Equation checked, proceed to check the value of individual component.

Controller: Writing command file to request detail of (key(exp, [1,9,9,9], dollar, manufacturing)).

Controller: Please start another instance of prolog to generate response.

Controller: When response is received, type 1 to continue the current process.

Manufacturing: Writing the components of the requested equation to file.

Controller: ***

Controller: Checking the following item: account(key(exp, [1,9,9,9], dollar, manufacturing), [15768,14088], 0.100000, 1983 Average salary)

Controller: 1983 Average salary = 15768 (key(exp, [1,9,9,9], dollar, manufacturing)) exceeds limit ---> NOT APPROVED

Controller: Writing command file to request detail of (key(exp, [1,9,8,8], person, manufacturing)).

Controller: Please start another instance of prolog to generate response.

Controller: When response is received, type 1 to continue the current process.

Manufacturing: Writing the components of the requested equation to file.

Controller: ***

Controller: Checking the following item: account(key(exp, [1,9,8,8], person, manufacturing), [109], 0.100000, Number of personnel)

Controller: Number of personnel (key(exp, [1,9,8,8], person, manufacturing)) = 109 ---> APPROVED

Controller: ***

Controller: Checking the following item: account(key(exp, [1,1,3,0], dollar, manufacturing), [456678], 0.100000, Other expenses)

Controller: The requested item Other expenses (key(exp, [1,1,3,0], dollar, manufacturing)) exceeds limit.

Controller: Ask the originator to supply information on any categorical sub-accounts.

Controller: Writing command file to request detail of (key(exp, [1,1,3,0], dollar, manufacturing)).

Controller: Please start another instance of prolog to generate response.

Controller: When response is received, type 1 to continue the current process.

Manufacturing:	Collecting categorical sub-accounts
----------------	-------------------------------------

Manufacturing:	Writing collected accounts to file
----------------	------------------------------------

Controller: ***

Controller: Checking the following item: account(key(exp, [1,1,3,1], dollar, manufacturing), [204408], 0.100000, Misc. expenses)

Controller: Misc. expenses (key(exp, [1,1,3,1], dollar, manufacturing)) = 204408 ---> APPROVED

Controller: ***

Controller: Checking the following item: account(key(exp, [1,1,3,2], dollar, manufacturing), [252270], 0.100000, Workload increase)

Controller: Workload increase = 252270 (key(exp, [1,1,3,2], dollar, manufacturing)) exceeds limit ---> NOT APPROVED

Controller: ***

Controller: Checking the following item: account(key(exp, [1,2,0,0], dollar, manufacturing), [90000], 0.100000, Manufacturing material and supplies)

Controller: Manufacturing material and supplies (key(exp, [1,2,0,0], dollar, manufacturing)) = 90000 ---> APPROVED

Controller: ***

Controller: Checking the following item: account(key(exp, [1,3,0,0], dollar, manufacturing), [490000], 0.100000, Manufacturing computer services)

Controller: The requested item Manufacturing computer services (key(exp, [1,3,0,0], dollar, manufacturing)) exceeds limit.

Controller: Ask the originator to supply information on any categorical sub-accounts.

Controller: Writing command file to request detail of (key(exp, [1,3,0,0], dollar, manufacturing)).

Controller: Please start another instance of prolog to generate response.

Controller: When response is received, type 1 to continue the current process.

Manufacturing:	Collecting categorical sub-accounts
Manufacturing:	Writing collected accounts to file

Controller: ***

Controller: Checking the following item: account(key(exp, [1,3,1,0], dollar, manufacturing), [410000], 0.100000, Regular services)

Controller: Regular services (key(exp, [1,3,1,0], dollar, manufacturing)) = 410000 ---> APPROVED

Controller: ***

Controller: Checking the following item: account(key(exp, [1,3,2,0], dollar, manufacturing), [80000], 0.100000, One time computer cost)

Controller: One time computer cost = 80000 (key(exp, [1,3,2,0], dollar, manufacturing)) exceeds limit ---> NOT APPROVED

Controller: ***

Controller: Checking the following item: account(key(exp, [1,4,0,0], dollar, manufacturing), [16000], 0.100000, Manufacturing miscellaneous expense)

Controller: Manufacturing miscellaneous expense (key(exp, [1,4,0,0], dollar, manufacturing)) = 16000 ---> APPROVED

Controller: no more items in this list