INTEGRATING SYSTEMS
FOR FINANCIAL INSTITUTIONS SERVICES USING
COMPOSITE INFORMATION SYSTEMS

by

MARIA de las NIEVES RINCON

B.S. in Computing Science,
University of Essex (England), 1978

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF

MASTER OF SCIENCE IN MANAGEMENT

at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June, 1987

© Maria de las Nieves Rincón

The author hereby grants to M.I.T. permission to reproduce and to distribute copies of this thesis document in whole or in part.

Signature of Author: ___________________________ Alfred P. Sloan School of Management May 15, 1987

Certified by: ___________________________________ Stuart E. Madnick
Thesis Supervisor

Accepted by: ____________________________________ Jeffrey A. Barks
Associate Dean, Master's and Bachelor's Programs
INTEGRATING SYSTEMS
FOR FINANCIAL INSTITUTION SERVICES
USING COMPOSITE INFORMATION SYSTEMS

by

MARIA de las NIEVES RINCON

Submitted to the Alfred P. Sloan School of Management on May 15, 1987 in partial fulfillment of the requirements for the degree of Master of Science in Management.

ABSTRACT

In this thesis, the case of a very large, multinational bank is analyzed. The focus of the thesis is on information technology, and in particular, on the implementation of systems for Financial Institutions. These systems will be analyzed and evaluated using the Composite Information Systems or CIS concept which is a methodology to map strategic applications to the appropriate technical and organizational solutions.

In the case of the bank, three CIS principles have been identified as critical for the implementation of FI systems. These are, integration, autonomy, and evolution. This work focuses mainly on the integration capabilities of these systems, and on which ways the technology available can enhance these capabilities.

This thesis addresses the following issues. 1) Analysis and evaluation of the current Financial Institution systems. 2) What technologies are being used?. 3) How do the systems conform to the goals of CIS?. And, 4) What are the constraints imposed by the current implementation to achieve integration?

Thesis Supervisor: Stuart E. Madnick
Title: Associate Professor of Management Science
Acknowledgements

I would like to give special thanks to Professor Stuart Madnick for his inspiration, guidance, and support not only throughout this thesis but also in the past few years.

I want to thank the people at Citibank, in particular Carlos Salvatory for his support to carry out this project, and Jim Calderella, Bill Frank and Les Kirschner for their time and invaluable contributions towards the completion of this thesis. I would also like to extend my appreciation to Professor Y. Richard Wang, Ted Hennessy, and Stephanie McCarthy for their contributions on this research project. Finally, I want to express my gratitude to "Fundación Gran Mariscal de Ayacucho" for their financial support during my second year at MIT.

Most of all, I would like to thank my parents for their continuous support and encouragement during all these years.
# TABLE OF CONTENTS

Abstract.................................................................................................................. 2

Acknowledgments..................................................................................................... 3

Introduction................................................................................................................ 6

Chapter 1: Composite Information Systems......................................................... 11
  1.1 The CIS concept......................................................................................... 11
  1.2 Potential CIS applications...................................................................... 13

Chapter 2: CIS principles: Autonomy, Integration, Evolution.......................... 16
  2.1 Autonomy............................................................................................... 16
  2.2 Integration............................................................................................... 22
  2.3 Evolution................................................................................................. 25

Chapter 3: The Financial Institution Products and Service................................................................................................. 27

Chapter 4: A historic overview of Financial Institution Systems............................ 32

Chapter 5. The CIS Conceptual Model................................................................. 40
  5.1 Conceptual Model Components................................................................. 40
  5.2 Autonomy, Integration, and Evolution aspects of the model......................... 45

Chapter 6. Analysis of the Financial Institution Systems.................................. 50
  6.1 The Funds Transfer System.................................................................... 52
  6.2 The Demand Deposit Account system.................................................... 67
  6.3 The Transaction Investigation system...................................................... 74
  6.4 Evaluation of the FI systems using the CIS conceptual model.................. 75
Chapter 7: Analyzing the Financial Institution Systems using the CIS model

7.1 Strategic goals of Financial Institutions. 82
7.2 CIS - The Financial Institution Systems... 86
7.3 Technical obstacles......................... 86
7.4 Organizational obstacles.................... 89

Chapter 8: Evaluation of the current implementation of FI systems

8.1 Integration characteristics of FI systems. 91
8.2 Problems posed by current implementation of FI systems.......................... 92
8.3 Implications of current implementation of FI systems.............................. 97
8.4 Costs of operation and maintenance of redundant databases................. 103

Chapter 9: Future directions

9.1 Consolidation of the dynamic database..... 110
9.2 Sharing the dynamic database............. 118
9.3 New technologies - Distributed databases.. 126

Conclusions................................................ 133

References................................................. 137

Bibliography............................................... 139
Introduction

In this thesis, the case of a very large multinational bank is analyzed. The focus and scope of the analysis is on information systems technology, and in particular, on the implementation of Financial Institution systems. These systems support the products and services provided by the bank to its Financial Institution customers in US Dollars. These customers are banks themselves both, domestic and international. Corporations and individuals are served by other organizations within the bank. The bank has over 8,000 Financial Institution customers. The volume of transactions processed per day in real time by some of the FI systems can be as high as 85,000. These transactions involve billions of dollars. For instance, the total amount of money transferred through the bank's Funds Transfer System can be up to $250 billion per day. Needless to say, the accuracy, reliability, risk control, and ability to provide a continuous service are of primary importance for the customers, and the bank.

The analysis and evaluation of the integration capabilities of Financial Institution Systems is done using the Composite Information Systems model [1]. The case of this bank is of particular interest since, first information systems technology plays an increasingly important role in the ability of a bank to deploy its products and services. Second, the
bank has a very decentralized organizational structure (see figure 1) which has had an important influence in the development and implementation of information systems. These systems have been developed independently following the decentralization trend, and the result has been in some cases, a collection of disparate systems. Third, the financial industry is undertaking continuous external changes, and the Financial Institution systems need to respond quickly to the changes imposed by external forces to keep its competitive advantage.

The issue of connectivity and integration among these disparate systems has also become increasingly important. Obsolete practices such as, tape hands-off to communicate among systems are still used in the bank. Tape hands-off require manual intervention, and slow down the communication across systems. Disparate information systems are usually accompanied by disperse and redundant databases. The dispersion of data has an impact on performance, operational costs, duplicate maintenance, inconsistency of data, and real time checks against the bank books. All of these are of critical importance for the transactions processed by the Financial Institution systems group. However, the integration of the systems must be achieved while still maintaining some of the flexibility that was obtained by the decentralization of systems development and operations.
Figure 1

The bank's organizational structure
Therefore, three attributes have been identified as critical in the development and implementation of Financial Institution systems: integration, autonomy, and evolution. **Integration**, refers to the ability to connect disparate systems, which in most cases have disperse and redundant databases, called "shadow databases." **Autonomy** refers to the independence of systems, which provides flexibility in their development and operation. And **evolution** refers to the ability of the systems to evolve to accommodate either internal or external changes.

A **conceptual model**, which is described and analyzed in this thesis, was developed in 1982 as a framework for the implementation of Financial Institution systems. This model attempts to mediate the conflicts of trying to meet the three attributes mentioned above. This paper evaluates the current implementation of the Financial Institution Systems which have followed the conceptual model guidelines. The integration and connectivity capabilities of the Financial Institution systems are further evaluated identifying strengths and weaknesses from the technical perspective, and potential future problems that may arise from their lack of integration. Organizational issues are also considered since they are an important factor in the successful implementation of technology, particularly in the bank.

The thesis focuses mainly on technological issues, and the following questions are addressed throughout the thesis. 1)
What systems are in place to serve Financial Institutions?, 2) What technologies are being used?. 3) How do the systems conform to the goal of CIS?. 4) What are the constraints imposed by the current systems to achieve integration?.

After evaluating the current implementation of Financial Institution Systems, alternative solutions are considered in order to achieve the goal of integration.
Chapter 1
Composite Information Systems

1.1 The Composite Information Systems concept.

The Composite Information Systems (CIS) concept is presented as a methodology to map strategic applications to appropriate solutions using information technology and organizational theories [1]. A Composite Information System is defined as a system which integrates independent systems which reside within or across organizational boundaries. CIS thus provides a top down process by which the strategic goals are identified and are linked to technological innovation and organizational structure (see figure 2). The CIS model recognizes that there are both technical and organizational obstacles that constrain the ability to connect systems that have been developed independently. Some of the technical problems encountered are, heterogeneous hardware configurations that do not communicate with each other, different software environments, independent databases, and different database management systems. Some of the organizational problems that can be identified are, lack of standards, great degree of divisional or departmental autonomy, and lack of communication among groups. These obstacles jeopardize, to a greater or lesser degree, the feasibility to connect these systems. The purpose
Figure 2
Composite Information System model
of the CIS concept is to help find technical and organizational solutions to achieve the interconnection of disparate systems. The interconnection of these systems can reduce costs by, exploiting economies of scale through the sharing of resources, eliminating human intervention and tape hands-off, reducing the number of errors, and eliminating duplication of functions, thus reducing operational costs. On the other hand, interconnection can make available distinct or complementary products not offered by the competition.

1.2 POTENTIAL CIS APPLICATIONS.

Madnick and Wang [2] have identified four categories of potential CIS applications: 1) Inter-organizational applications. These applications involve systems that go across organizational boundaries, such as linking customers and suppliers. Examples such as, American Hospital Supply system [3] and the Airlines Reservation Systems [4] are found in the literature and belong to this category. 2) Inter-functional applications. These applications involve systems that interconnect applications from different functional areas of the organization. For example, both the purchasing/inventory control system and the accounting system of a corporation need information regarding the materials that are delivered to the warehouse. The former needs the information to keep inventory control, the latter for invoicing purposes. However, it is not unusual to gather and
enter this information independently in each of the two systems. In this case, the failure to link these two systems leads to the duplication of the collection of information and data entry functions, and to potential discrepancies on inventory information. 3) Inter-product applications. The interconnection of applications that support independent products have become increasingly important to provide a single product that consolidates existing products or services. The Merrill Lynch's Cash Management Account example found in the literature [3] belongs to this category. 4) Inter-model applications. These applications involve the linkage of different models, policy models, econometrics models, etc., which may have been developed under very different environments.

All these potential applications require integration among intra-organizational or inter-organizational systems. The integration will ultimately accomplish cost reduction or product differentiation, as it was illustrated in the examples given above.

The advances in data base technology and data communications help to accomplish the goal of systems integration. However, integration may be jeopardized by historical and/or organizational reasons. Some systems were designed years ago, as independent systems and with old technology; thus, many of these systems have combined the data and the application into
a single package. On the other hand, some organizations have broken down the business of the firm into autonomous units that can be managed as viable and isolated business and where the goals of the management are confined to the unit. Thus, in many cases, they have distributed the processing power. As a result, independent systems, computing centers, and databases exist. Another problem is that the management of the different units may not be willing to relinquish the power and control they have gained through decentralization. CIS recognizes that these obstacles exist and that the goal of integration for these systems involves both technical and organizational solutions to solve the obstacles that interfere with integration.
Chapter 2

CIS principles: autonomy, integration, and evolution

The case of the Financial Institution systems being analyzed is of special interest since, first, most of the systems have been developed independently. Second, the organizational structure of the bank is highly decentralized, and the flexibility provided by local autonomy is very important for the management of the bank. Finally, the products and services offered to Financial Institutions vary with time and the systems need to conform to these changes.

Thus, CIS recognizes that the principles of autonomy, integration, and evolution need to be mediated to keep the advantages of each one of them. A discussion of these principles in the context of the Financial Institution systems is presented in this chapter.

2.1 AUTONOMY.

Autonomy refers to the capability of treating each system as a discrete entity that can be developed, operated, and managed independently of the other systems and where each system must maintain a level of processing integrity to complete its functional responsibility as if it were totally independent of other systems. Autonomy is a very important attribute that
matches the organizational structure of the bank which is highly decentralized, and where control, responsibility and accountability have been distributed to the lowest management level of the organization. From the systems perspective, autonomy refers to the ability of individual managers to control the resources they need to support their business needs.

The autonomy discussion can be related to the centralization vs. decentralization dilemma. Different authors have discussed this issue. In particular, Rockart [5] presents a framework for the analysis of centralization vs decentralization. The framework identifies three dimensions where decentralization can be implemented namely, systems operation, systems development, and systems management (Figure 3). Although this framework is very useful, it is too general in the context of the bank. Therefore, based on Rockart's framework and taking into account the characteristics of the bank, autonomy has been identified in six categories or dimensions which are illustrated in figure 4. Autonomy can be implemented in none, some, or all of these categories. The six categories are:

1. **Hardware control.** This category refers to the centralization or decentralization of processing capacity. The degree of centralization can be so much as having a large central hub that serves the whole organization to the
The Financial Institution systems as of 1986
other extreme where each single unit has its own processing power capacity.

2. **Operational control** refers to controlling and running the operations of the data center. This category covers a range of functions, from hardware and operating systems maintenance to capacity planning. Although operational control is usually tied to hardware control (first dimension,) this need not be the case. For example, even if hardware control (i.e., capacity power) has been distributed to sub-organizations, the operations can still be run and controlled by centralized systems staff.

3. **Transaction control** refers to the management of the unit's transactions. This category includes functions such as, data entry, verification, and repair of transactions.

4. **Software development** refers to the design, implementation, and maintenance of software for applications. Centralized software development is done by central personnel staff. On the other hand, software development can be moved to the units to be closer to the users where it will be done either in-house or by contracting outside firms.

5. **Data control** refers to the development, control, and maintenance of the database(s). Centralized data control does not mean that all data should be kept in a single file
Figure 4

Centralization vs. Decentralization dimensions

The Financial Institution systems as of 1986
but that the control of the data should be centralized, e.g. propagate updates, set standards to get access to data, etc. Many organizations are developing a data dictionary in which information about the data is stored and the data is standardized to facilitate communication among applications. Other organizations are moving towards distributed databases which allows the access to data contained at different locations.

6. **Management control.** Within this category, the functions performed are: the setting of standards, planning for data processing, hardware evaluation and acquisition, and decision making regarding the projects to be implemented.

A high degree of autonomy simplifies the task of managing each division. However, it presents some disadvantages: duplication of functions, inefficient use of resources, reduction of economies of scale, independent and redundant databases, and lack or poor communication among systems, such as tape hands-off.

As it will be discussed later, the systems in the bank have been developed independently having as a result independent databases. However, a lot of the information contained in the different databases is common or redundant across databases, creating "shadow databases". As a result of the shadow databases, updates need to be propagated to avoid
inconsistencies in the data contained in them. In some cases, consistency of the information is of critical importance, as is the case with a customer's account balances. This information is used by several applications, each of which has its independent database. So, procedures need to be set up to perform the updates propagation either in real-time or batch. If it does not need to be real time, a file with all the updates can be sent daily. In many cases, these procedures require human intervention, either "tape hands-off" or by entering manually the changes to the database. Otherwise, updates can be propagated automatically, exchanging messages across applications. If the updates need to be done in real time and there is a large number of those, this approach can result in an overload of messages. In all these cases, software needs to be in place to take care of the propagation of updates.

2.2 INTEGRATION.

Integration refers to the coordination and consistency across systems. Some general characteristics of integrated information systems are, to share common data, to enable communication among systems, and share resources. Integration provides several advantages over independent systems. It allows reduction in the duplication of functions, more efficient utilization of resources, more consistency across systems, and transparency in the use of different systems.
However, it also requires a higher degree of complexity to coordinate the development and operation of all these systems. Integration can vary in degree, one extreme would be to have a single processor where all the systems are hosted. In this case, other factors such as, response time, ease of migration to a larger system, and lack of flexibility to run some applications arise. The bank has an example of a such a system running in its overseas branches [6]. However, other organizations have followed the trend towards distributed processing given the better price/performance ratios for hardware, and the cost reductions and improvements in data communication and database technology. These organizations are finding a need for the integration of their systems. This last case holds true for the bank being evaluated.

The reasons for achieving system integration in the bank are several: to provide more consolidated services to the customer, to improve the internal operations of the bank, and to reduce operational costs. To keep its competitive advantage, the bank has an increased need to offer consolidated products that cut across different products and services. Systems integration is especially important to the bank since the systems are highly independent of each other as a consequence of its organizational structure.

Integration is also important for the internal operations of the bank, to better monitor transactions and to reduce risk
exposure. An example where globally integrated information is critical for the exposure of the bank is the monitoring of high risk economic situations. For example, it is necessary for the bank to know the exposure of Mexican pesos not only in their Mexican branch but its exposure for all branches worldwide. Integration is also important to check overdraft and balances against the actual bank books.

Standardization of some of the dimensions, hardware, software, programming languages, data, communications, and external interfaces is a way to help accomplishing the integration goal. Standardization of hardware, software, and data provides the basis for easier integration even if connection of current applications may not be a foreseeable need. Standard hardware and software will make it easier to interface applications when that need arrives, and standardization of data, (e.g. data definitions,) will ease the interchange of data among different applications.

There is a trade-off between the cost in developing vs. maintaining systems. Although integration usually requires a higher cost up-front, it will reduce future maintenance costs. For example, if a shared database is developed for all applications that need common data, higher costs and longer time will be spent for initial development (i.e., complexity, coordination) but there will be future gains in terms of maintaining the databases (e.g. consistency of data).
Maintenance of disperse databases usually involves, tape hands-off, human intervention, and flow of messages across applications. Another example is provided by separating the data from the application by using data base management system technology. As applications will be separated from the data, future changes to the data will not require to change all the applications that have access to the database.

2.3 EVOLUTION.

The last CIS principle is evolution. This term means that the systems should be capable to accommodate change. The processing requirements within the bank are constantly changing by the introduction of new products, changes in the existing products, and changes in the external business environment, such as deregulation, improvements in processing methods, and changes in technology. Therefore, components within the system must be modular to adjust to change without disrupting other systems, and to minimize complexity in the evolutionary process.

Evolution also refers to the ability of the systems to accommodate growth. The processing capacity for the systems should conform to increases in the demand of the products offered to the Financial Institutions. Therefore, flexibility in the architecture for augmenting capacity should be taken into account. This implies flexibility in both dimensions,
hardware, and software.

Evolution is sometimes constrained by hardware and software limitations. For example, even if a DBMS would be desirable to ease the evolutionary process, it may not be feasible to implement due to performance issues. The complexity of systems in place are also a constraint on the ease of evolution. Adding functionality may take several months to be implemented. Other considerations such as, ease of migration should also be taken into account, e.g., if hardware runs out of capacity, the ability to migrate towards more powerful hardware (i.e., processor or peripheral devices) is a necessity. These issues need to be considered in the initial development to avoid future costly maintenance to the systems.

An example where evolution is important is the ability to provide the customer with integrated information. For example, a customer may want to have one consolidated statement with all his/her business transactions, such as loans, checking activity, funds transfers, etc. Although this product may not be demanded today, the systems in place should be able to evolve in order to provide this integrated service, whenever needed.
Chapter 3
The Financial Institution products and services

This chapter presents a description of the systems developed by the Financial Institution Systems Group (FISG) to support the products and services that are offered to the Financial Institutions customers [7]. These customers are defined as those which are themselves banks, including both, domestic and international banks. All the transactions handled by FISG are in US dollars. No multicurrency transactions are handled by this group. A summary of the FI products is presented in figure 5. As mentioned previously, the scope of this thesis will be limited to analyze and evaluate the systems developed by FISG.

The Financial Institution Systems Group is based in New York and gives support to the business divisions which offer products and services to Financial Institutions for all the transactions processed in US Dollars. The functions served by the systems group can be classified as either business banking functions, such as Funds Transfer, Letter of Credit, Cash Management, etc; or supporting management functions, such as accounting analysis, and reporting.

A brief explanation of the products and services that are offered to the Financial Institutions customers follows:
<table>
<thead>
<tr>
<th>Product</th>
<th>Description</th>
<th>Average # transactions per day</th>
<th># of customers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Funds Transfer</td>
<td>Move funds electronically in US Dollars to domestic or international beneficiaries</td>
<td>60,000</td>
<td>8,000+</td>
</tr>
<tr>
<td>Cash Management</td>
<td>Customers are connected electronically to the bank and can perform funds transfers, balance inquiry, and transaction history inquiry</td>
<td>10,000</td>
<td>900+</td>
</tr>
<tr>
<td>Demand Deposit Accounts</td>
<td>Automated accounting and reporting system for customers</td>
<td>50,000</td>
<td>8,000+</td>
</tr>
<tr>
<td>Trade Services</td>
<td>Letter of credit operations and collections</td>
<td>1000+</td>
<td>8,000+</td>
</tr>
<tr>
<td>Inquiry/Investigations</td>
<td>Find causes of discrepancies between customer's and bank's records for funds transfer operations</td>
<td>1,500</td>
<td>8,000+</td>
</tr>
</tbody>
</table>

**Figure 5**

Financial Institution's products
1. **Funds transfer.** The Funds Transfer service allows customers to electronically move funds through the bank to domestic or international beneficiaries. The bank has direct, on-line access to Fed wire, S.W.I.F.T., C.H.I.P.S., and Bankwire/Cashwire. It also accepts input from the Cash Management Account, structured or unstructured format telex, mail, phone or facsimile. The funds transfer operation of this bank is one of the largest in the world. Besides having 8000+ Financial Institution customers, the bank serves as an intermediary for other Financial Institutions and therefore needs to keep information of about 20,000 direct and indirect customers. It handles an average of 60,000 transfers operations per day. However, in peak days over 80,000 transfers operations can be generated, which represent about $250 billion in funds transfers processed in one business day. The demand for funds transfer services has experienced a continuous growth of 25% per year.

2. **Cash Management Account.** The Cash Management Account is an online, interactive electronic reporting and payment system for US Dollar accounts in New York. Customers are electronically connected to the bank and can obtain balance information, access a 45-day transaction history, send funds transfers, track a letter of credit and initiate letter of credit reimbursement authorizations and amendments. The average number of transactions for the Cash Management Account is 10,000, approximately 4,000 of which are funds
transfer operations, and 6,000 are inquiries. There are 900+ customers for this product, and this number is expected to grow to 1,700 customers by 1990.

3. Demand Deposit Accounts Statements. DDA statement is the bank's automated accounting and reporting system for customers holding demand deposit accounts, depending on the customer's needs. Account statements are generated monthly, bi-weekly, weekly, or daily. This statement can either be descriptive, showing the detail of each item; or non-descriptive, summarizing the transactions. An account history statement is also generated for customers. This statement shows all overdraft charges and refunds of interest due to overdrawn accounts and backvalue adjustments. The Demand Deposit Account system has 8000+ customers (accounts) and handles 50,000 postings per day.

4. Trade services. This includes letter of credit operations, and collection. Letters of credit issued by a bank on behalf of its client, give a designated beneficiary the right to draw funds in accordance with the stipulated terms and conditions in the Letter of Credit. Trade services handle 1000+ transactions per day.

5. Inquiry/Investigation support. In financial transactions, a large number of discrepancies occur between customer records and bank records. Customers inquiry about the
source of these discrepancies and the bank is responsible to find their causes using the bank's records, and to fix the discrepancy if the bank is in error. This kind of inquiry is called an investigation. At the moment all the investigations involve funds transfer transactions. The inquiry/investigation division performs an average of 1,500 investigations per day.

The systems that support these products have been developed independently and there is a one to one match between the products described above and the systems that support them. The technical, organizational, and historical reasons for these independent systems, and their technical implications and connectivity capabilities will be analyzed in the following chapters.
Chapter 4

A historic overview of Financial Institution Systems

The Financial Institution Systems (FIS) have evolved through different stages. From a very centralized systems structure in the early 1970's, they changed to an extremely decentralized one in the middle 1970's. In the 1980's, they are implementing a new architecture (explained in the next chapter), that contains elements of both centralization and decentralization. This chapter discusses these three stages in the evolution of the Financial Institution's systems.

In the early 1970's, the Financial Institution systems followed a very centralized structure. All systems were big, complex, and hosted in one IBM computer. The control, maintenance, software development, operations, and management of the systems were all centralized and handled by the Information Systems Group.

In the 1970s, the organizational structure of the bank was very decentralized, with autonomy and control given to the lowest management level in the organization. The objective of this structure was to allow for rapid change and encourage creativity. However, the systems in place, which were highly centralized, did not comply with this decentralized structure. In order to fit the information systems structure to the
organizational structure, the systems in-place needed to be broken down into small pieces that were more manageable from the organizational, operational, and technological point of view.

To achieve this, each business division was given the responsibility to run their own data center, having its own processing capabilities, database, and operational staff. Thus, systems were decentralized and autonomy was granted to the business divisions in all the dimensions, hardware, operational control, software development, data control, and management control. As a result, over 90 computers and more than 20 applications were put in place. (Figure 6 shows the proliferation of computers at that time.) The environment included five processor vendors (among them were, DEC, Prime, and Data General,) six distinct processor families, ten distinct operating systems, eight distinct programming languages and independently developed software modules. This variety of hardware and software had significant implications for the cost-effectiveness and flexibility of the systems.

The information systems people have expressed that "the decentralization of the systems was driven by the organizational structure and a price on technology had to be paid." A price on technology was paid because the trend in computer technology in the 1970's was towards centralization. IBM had very few minicomputers to offer. Instead, IBM offered
Figure 6

1970's Financial Institution systems
large computers with software off-the-shelf, all utilities incorporated into the system, and communication facilities. On the other hand, DEC provided better minicomputer systems but with almost no software available for their systems. Since the management goal was to give a computer to each business group, minicomputer technology was chosen. So, even though in the 1970's the best choice would have probably been IBM since it provided systems which have more software available off-the-shelf, they could not afford to place 20 IBM computers for the Financial Institution systems. Therefore, in the hardware evaluation process, the technology issue was outweighed by organizational priorities.

However, some people within the bank have expressed that "even if a high price on technology was paid, it was the right thing to do because technology ought to respond to the business instead of driving the business." In retrospect, the bank by following this philosophy of autonomy and decentralization was relatively successful at completing systems to meet business objectives. In contrast with this bank, other banks that followed a centralized approach were not so successful at completing the same kind of systems. The reason for this was the degree of complexity of the systems they were developing.

The main criticism of this stage is precisely its high degree of decentralization. The systems implemented were too small, too heterogeneous, and too numerous. Most business divisions
did not have the expertise to either acquire hardware for their business needs or to develop the necessary software in-house. Thus, different consulting firms were hired for the development of the software. This was costly and the resulting systems had very little commonality. Furthermore, the hardware configurations were different and in many cases, incompatible to each other. There was a lot of duplication in the development of the systems, and a lot of money and time had to be spent in building interfaces among these heterogeneous systems. Finally, each business divisions had its own data center having the responsibility to run and operate them. The degree of centralization vs. decentralization at this time is shown in the following graph:
It did not take very long before management realized that in order to give autonomy to the business divisions, they did not have to relinquish the control over the hardware, software development, and operational control of the data centers. The lack of expertise from the business divisions to run and operate the data centers became a major problem, and the operational control of the data centers was given back to the systems group. Thus, each business group had its own data center but the systems group had the responsibility to operate them. On the other hand, the business divisions kept the transaction control, i.e. data entry, repair and verification of their transactions. The degree of recentralization in the 1970's is shown in the following graph:
The diversity of hardware and software, and the high degree of decentralization of the 1970's had two implications for the Financial Institution systems: 1) Integration. There was an increasing need for integration, but the proliferation of heterogeneous systems made it difficult to integrate the systems. 2) Evolution. The systems in place were difficult to evolve to meet new business requirements and to accommodate growth. Thus, in 1982, a new architecture was developed to mediate the conflicts between autonomy and integration. This new architecture which is called the conceptual model would provide supervisory functions, terminal grouping, and security functions as if every business division had its own computer but without having to give one computer to each single division.

So, even though in the 1980's the economics of computer technology tend to favor the distribution of processing power, there are other problems that may work against this trend. Distributed computing implies, operation of the computers, duplication of functions, building of interfaces to communicate them, and creation of new bottlenecks, such as a communication interface. Thus, from the systems perspective, it would be more cost effective to centralize some of the functions. Following this, the bank started a move to change from a physical distribution to a logical distribution. The degree of centralization vs. decentralization in the 1980's is
shown in the following graph:

![Graph showing the degree of decentralization for Hardware Operational Transaction Software Data Management control control control development control control control]

- Hardware control
- Operational control
- Transaction control
- Software development control
- Data control
- Management control
Chapter 5
The CIS Conceptual Model

The CIS conceptual model was developed as a baseline architecture for the Financial Institution systems seeking technical solutions that do not conflict with the organizational characteristics of the bank. This model is evaluated against the principles of autonomy, integration, and evolution. The model is composed of six components and it is shown in Figure 7 [8].

5.1 CONCEPTUAL MODEL COMPONENTS

1. **External Interface**
The external interface component is responsible for the interface between the bank's systems and external entities which communicate electronically with the bank. This component is responsible for the communication to:

- Payment networks, such as Fedwire and CHIPS.
- Communication networks, such as SWIFT.
- Other systems within the bank.
- Customer terminals.
- Workstations professionals.
Figure 7

The CIS conceptual model
There are three modes of communication that the External Interface component should support:

1. **Message exchange.** The exchange of communication is a discrete element which contains the information required to satisfy a service request.

2. **Interactive form.** The exchange of communication is established by a dialogue between systems. In a dialogue usually more than one message is transmitted.

3. **Bulk form.** This is similar to the message exchange mode since it is the transmission of complete and discrete information packages. However, the size of bulk packages is generally larger than a message.

2. **Message control**

The message control component is responsible for the transmission of information between different processing components, i.e. inter-computer communication, terminal control, etc. This component is responsible for performing the following functions: 1) Routing which accepts a request and determines the physical address to which the message should be delivered to. 2) Translation which maps a limited number of protocols from one standard to another. 3) Sequencing which determines the order to deliver messages. And 4) Monitoring which determines the integrity of messages as they are transmitted through the system at any given time.
3. Transaction Processing

The transaction processing component is composed of all the applications which result in changes to the financial condition of a customer. The ultimate result will be updates to the database. This component performs the following functions:

- Risk management functions, such as checking that a transaction does not violate any condition imposed by the bank, customer, or regulatory agency.
- Validation functions to ensure that all information required for processing is provided, and if provided that it is correct. The validation and repair of transactions will specifically be designed for the application.
- Accounting. The TP component should record all transactions being performed for accounting purposes.
- Management Information Recording. The TP should also record the necessary information to support the decision-making processes of the bank.

4. Information Processing

The information processing component is composed of all the applications that provide information to support the decision making of both customers and internal management; no changes occur in the financial condition of a customer. This component will communicate with databases which are static and historic in nature. This component should perform the
following functions:

- Data analysis to provide the capability to present conclusions based on the data analyzed. The source of this data can be both internal and external to the bank.
- Ad Hoc reporting to provide the capability of obtaining reports as they are needed. The information is based on transaction processing functions or external sources.
- Static reporting. The production of reports which have been defined in advanced. The data to generate these reports is usually obtained from the Transaction Processing component or from external sources.

5) **Data control**

This component controls the access from the processing components to the information that is common to several applications. The following functions are performed by data control: 1) Concurrency control which ensures that multiple requests do not change the same piece of data at the same time, thus preventing inconsistency of data. 2) Update propagation which makes sure that redundant data is updated appropriately. 3) Security which prevents unauthorized access to the database, and controls the view of the data permitted to different users. 4) Routing which determines the segments of the database to be accessed in response to a request, and returns its result to the appropriate processing element. And 5) Sequencing which determines the order in which requests
should have access to the database.

6) **Shared Data Resource**

The shared data resource component is responsible to maintain all the information that is common to different components of the systems. This component does not have to be a single database but it could be distributed according to the needs of information that different processing components have.

This model allows for the inter-organization and intra-organizational potential applications that Madnick and Wang have identified [2]. For example, the Message control component allows for inter-computer communication, so even if different applications are hosted in different computers, transactions can be sent through the centralized message control. Furthermore, it provides a single gateway entrance to all applications which permits communication to any application from any terminal. For example, data can be collected and entered from a single point and then distributed to all the applications requiring that particular information.

5.2 **AUTONOMY, INTEGRATION, AND EVOLUTION ASPECTS OF THE MODEL**

An ideal implementation of the conceptual model for the Financial Institution systems is presented in figure 8.
5.2.1 **AUTONOMY**.

The autonomy principle is reflected in the conceptual model by providing the ability to develop systems independently, i.e., the development of one application is not dependent on the development of any other application. In fact, each one of these applications can be hosted in a different computer. In the CIS conceptual model, the components are independent of each other with the exceptions of, the data control, and the message control components. The applications contained within each component can be as autonomous as desired. In fact, they can have their own hardware, software, and database.

The autonomy in the development of systems can lead to independent systems having autonomous databases which is in conflict with the shared data resource component of the CIS conceptual model. If the independent applications have common data but they do not share the database, the result will be to have "shadow databases" of the original database (i.e., redundancy). One of the reasons for different systems to have independent databases is that managers may not be willing to wait upon the completion of a global database that will be shared by several applications in order to start the development of their own systems.

The infrastructure outlined in the conceptual model does not interfere with the autonomy of the divisions to have their own
Figure 8
Ideal Implementation of conceptual model
operators for data entry, repair, and verification. Thus they still have control over their own transactions. Even though all the transactions may reside in the same computer, and they share the same transaction queue, the system will behave as if they were independent systems.

5.2.2 INTEGRATION.

The conceptual model also conforms to the integration goal. There are two components that are critical for integration, the message control, and the data control. The message control component enables inter-computer communications. Thus, even if applications are hosted in different computers, the protocols to communicate among them have been standardized facilitating their interfacing. Moreover, this component provides a single entrance to all applications for Financial Institutions Thus, any terminal will have the capability to access any application. This makes it possible, the single customer interface for all the products and services offered.

The data control allows for the control and sharing of information that is common to more than one application. Different applications can have different views of the same share data resource. Therefore, functions such as, risk management control that require the communication and integration of several application can now be performed.
5.2.3 EVOLUTION.

Finally, the conceptual model provides a much better environment for evolution, in terms of volume growth and creating new products and services since it provides for better integration among applications, and its modular characteristics allows the extension of hardware to process more transactions.
In this chapter, the current implementation of the Financial Institution systems is analyzed. This implementation follows the guidelines of the conceptual model described in chapter 5. In particular, this chapter contains an analysis of the following components of the Financial Institution Systems, the Funds Transfer system (FTS), the Demand Deposit Account system (DDA), and the Transaction Investigation system. No analysis is provided for the Cash Management System (CM) since it was only rehosted from a Data General to a DEC VAX with no further changes.

TECHNOLOGY CHOSEN

The technology used has been standardized to DEC hardware, and to the MVS operating system.

1. **VENDOR**

The vendor chosen was DEC. The main reasons for this selection were that the information systems group has had extensive expertise with DEC systems; and that most of the Financial Institution systems were already running on DEC systems (PDP 11). This reduced the complexities and risks of transition to the current implementation.
2. **PROCESSOR**

The specific processors chosen were the VAX 785 and the VAX 8600. The reasons for this selection were:

- **Performance:** The VAX/785 has been rated at 1.5 MIPS and the VAX/8600 has been rated at 4-5 MIPS. The VAX/8600 represents an improvement of over 500% in price performance over the previously used PDP 11. MIPS are an important measure since most processors were running up to capacity and some of the systems are CPU intensive.

- **Compatibility:** The VAX 785 and VAX 8600 are totally compatible with the entire family of DEC processors and peripherals.

- The VAX 785 and VAX 8600 can enter a VAX cluster. DEC VAX cluster architecture allows several VAX computers to share multiple sets of data files.

3. **DATA COMMUNICATIONS**

The communications technology chosen was Digital Network Architecture using Decnet/Ethernet which are established products for communication between DEC computers. The physical capacity of Ethernet is 10 million bits per second which exceeds the current needs of FI systems.
4. SOFTWARE

The software chosen was DEC's VMS operating systems. VMS contains all the basic systems services and utilities required to develop the different applications. However, different database technology and programming languages were still chosen for the different applications.

6.1 ANALYSIS OF THE FUNDS TRANSFER SYSTEM.

To analyze the Funds Transfer System in perspective, a brief evaluation of the system under the previous architecture is discussed. Then, a description and analysis of the current implementation is done.

6.1.1 THE FUNDS TRANSFER SYSTEM UNDER THE 1970s ARCHITECTURE

The Funds Transfer System set up in the 1970s is illustrated in figure 9 and presents the following characteristics [9]:

1. Customers were classified by banking groups (e.g. North America, International, etc.) Each banking group had an autonomous processing environment, with its own processor capability to host the application program namely, Customer Information Manager (CIM). Each group also had its own front-end and external interface hardware and software, and its own independent database. There were a total of five of these environments. One vendor was contracted to develop
the CIMs software, whereas two other vendors developed the front-end, and external interfaces.

2. The hardware chosen was minicomputers. Each CIM was hosted in a minicomputer, and each external interface and front-end were hosted in another minicomputer. Most minicomputers used were PDP 11.

3. The CIMs and front-end applications were mainly developed using low level languages (assembly language), in contrast to using high level languages that would be compatible among different hardware configurations.

4. Communications among CIMs was necessary since most funds transfer transactions generated secondary transactions to be processed in other CIMs. A Communications Controller was in place to communicate transactions/messages between the CIMs.

The problems this architecture posed were numerous. For example:

1. When customer groups were reorganized as the organization of the bank changed, this architecture made it difficult to accommodate needed organizational changes since each banking group had its own processor and database.

2. On average, each FTS transaction spawned 2 1/2 secondary
FUNDS TRANSFER SYSTEM
OLD ARCHITECTURE

Bankswitch

To all XPs

SWIFT

Front-end

SWMH

To other applications

CHIPS

CMH

CM

XP

CIM

Communication Controller

CM

XP

CIM

Switch-transfer

FED

CM

FMH

XP

CIM

FMH

CM

XP

CIM

Figure 9
FTS systems - 1970s architecture
transactions to be processed in other CIMs [9]. The fact that CIMs had independent databases increased dramatically the number of transactions that needed to be processed. For example, if 20,000 funds transfer operations were generated daily, an average of 50,000 transactions would be processed.

3. A new bottleneck was created in the system at the Communications Controller since the transmission of transactions among CIMs increased dramatically as volume increased.

4. Each banking group had its own processor for the application (CIM), and for the external interfaces. This created a proliferation of minicomputers, heterogeneous software, duplication of functions, and difficulty in integrating the Funds Transfer System application to other Financial Institution Systems.

5. Given the increasing demand for funds transfer operations, the Funds Transfer System was nearing its maximum capacity, thus degrading to unacceptable limits for customer service the processing capabilities of FTS.

6. The PDP 11, introduced in 1970, was reaching the end of its product life cycle. New hardware was providing much more capacity in terms of MIPS, (processing capacity), and better interfaces to newer peripheral devices.
The main technical problems preventing the easy integration of Funds Transfer applications in the old implementation were, the one to one relation between minicomputers and applications, heterogenous software, the saturation of processing and network capacity, and the independent databases per CIMs. The autonomous and numerous data centers resulted in duplicity of functions which had as a consequence high operational costs.

6.1.2 THE FUNDS TRANSFER SYSTEM UNDER THE NEW ARCHITECTURE

As a result of the limitations posed by the previous architecture, a new funds transfer system was designed in 1984 following the conceptual model guidelines to achieve the required level of integration, processing efficiency, and control. Five main decisions were made:

- A homogeneous operating environment was to be created.
- A unified database was to be used using the DEC VAX cluster technology.
- A centralized and high performance network for inter-computer communications and terminal support was implemented.
- Centralization of message and payment interfaces support was implemented.
In the new Funds Transfer system, there were fundamental changes in the architecture of the system but no reinvention or redesign of the applications was made. The new system (FTS) was divided into four main components, shown in figure 10. These four components are, the Message Processor (MP), the Transaction Processor (TP), the Exception Processor (XP), and a unified FT database.

The specific processor chosen to host the applications was the VAX 8600. In particular, 4 VAXs 8600 were used, one for the Transaction Processor, one for the Exception Processor, and two for the Message Processor. Furthermore, FTS was implemented in a VAX cluster, so that the processors for TP, XP, and MP could share multiple sets of data files. The VMS operating system and the programming language C were chosen to develop the applications. The programming language C replaced the low level languages that were used previously to develop applications.

A description of the four components of FTS follows:

1. **Message Processor (MP).**

   The Message Processor has responsibility for receiving and transmitting messages between the Funds Transfer applications and all the external entities. The external entities are among others; the New York CHIPS, the Federal Reserve Bank, SWIFT, and an Electronic Banking Micro located in the
Figure 10
Funds Transfer System - New architecture
customer's offices. One VAX processor is used to host all the payment network interfaces (e.g. FED, CHIPS). Another VAX processor is used to host all the message network interfaces (e.g. SWIFT). Thus, the message processor consolidates all the external interfaces that were previously integrated into the application and running in different hardware as illustrated in figure 9. The functions performed by the MP for all the messages types include, test/authentication, addressing, editing of message content, and reformating of messages. If the message received is unstructured, the information provided is incomplete, or any errors are found, the message will be routed to the Exception Processor.

2. Transaction Processor (TP).

The Transaction Processor replaces and performs all the functions previously assigned to the CIMs and the Network Controller. Its functions include, balance and amount controls, debit authority controls, and posting. The message is routed to the Exception Processor if any errors are found, or any decision needs to be made.

3. The Exception Processor (XP).

This application handles all the exceptions: 1) Errors discovered during the Message Processing or Transaction Processing. 2) Unformatted messages or messages with incomplete information. 3) Transactions that require decision making capabilities. In these cases, the message is routed to
the Exception Processor. The XP supports the following functions: data entry, repair of messages, and database maintenance and inquiry functions. An expert system has been developed to support some of the functions performed by the Exception Processor. This expert system attempts to interpret unstructured or incomplete messages. If the expert system can perform the interpretation, an operator only has to check the expert system output against the original message. If the expert system cannot interpret the message, the message is then manually interpreted and reformatted by an operator.

4. A Unified Database.
A unified database was designed to be shared by the Transaction Processor, the Message Processor, and the Exception Processor. The unified database substitutes the segregated CIM databases in the previous architecture. The Funds Transfer system manages over 30 files which are classified into transaction files, static files, and dynamic files. An average of 40 to 50 physical accesses to disk are made by each funds transfer transaction. The physical accesses include, get customer information, message repair, get/change customer balances, communication among systems, and recording of messages for restore and recover capabilities. Considering that an average of 60,000 funds transfer transactions are processed daily, the number of accesses to disk are significant.
The software used to manage the FTS files are, TMX-32 software, and the DEC RMS file system. The TMX-32 software is a transaction manager which is used to manage the transaction queues. The main reason for using TMX-32 is the need for high performance, recoverability, and ease of maintenance. On the other hand, the VAX RMS file system is used to manage the dynamic files. These files contain information such as, account balances, and overdraft authorizations. Finally, the static information files which contain information such as, customer names and account numbers are read directly from within the memory for performance reasons.

**TRANSITION PLAN**

The entire plan would take three years to be implemented. A transition plan was devised to rehost all the applications in the new hardware. The plan was divided into eight independent projects with their own project manager and staff. The eight projects included the rehosting and design of the new Transaction Processor, the Exception Processor, and the Message Processor. Although there were fundamental changes in the architecture of the system, no reinvention or redesign of the applications was considered except for integrating some of the segregated applications. Finally, functionality has been added incrementally to the expert system to support the reformating of unstructured messages coming from the FED, CHIPS, SWIFT, and all the other message handlers.
6.1.3 A FUNDS TRANSFER TRANSACTION EXAMPLE

A typical example of a request for a funds transfer from say, Company A in Boston to Company B in London going through the Funds Transfer system is presented below. Here, company A in Boston wants to make a payment of $35 million to Company B in London which has a dollar account with the bank's London branch. A flowchart showing the sequence of steps followed by the transaction is shown in figure 11 and explained below:

1. Company A has an account in a bank in Boston and calls its account representative to perform the payment. The bank in Boston transfers the amount from Company A's account to its own account.

2. The bank in Boston performs a payment to the FED.

3. The FED in NY performs a payment to our bank in NY.

4. At this point, the bank's Funds Transfer system receives a payment message from the FED. Figure 12 shows how this funds transfer transaction flows through the bank's Funds Transfer system. The payment message is received by the Message Processor and is read by the FED Message Handler (FMH) that is hosted in the Message Processor. The FMH acknowledges the reception of a message by recording a copy of the message in the database. After the FMH has read the
Figure 11
Funds transfer operation
message, it attempts to interpret it. Two different paths are followed depending on whether the message is structured or unstructured:

A. **Structured message.** FMH interprets the message by extracting information such as, the originator, the destination, the amount of the payment, and some other reference information. A Financial Transaction record (FTR) is created with this information. Other information such as, the name and address of the destination entity, and the day and time at which the message was received is also included in the FTR. The FTR is then recorded in the database. An average of 75% of the messages are structured and go through the system with no human intervention.

B. **Unstructured message.** FMH can not interpret the message because some information is missing, the information in the message cannot be extracted, or some decision making is needed. Thus, this message is sent to the Exception Processor through the Local Area Network for repair. The Exception Processor is supported by an expert system, which interprets the message, repairs it, and creates a Financial Transaction Record. If the expert system can interpret the message, an operator will verify that the expert system has created the FTR correctly. If the expert system cannot interpret the message, it will send
it to the Exception Processor to be repaired manually by an operator.

5. Once the message has been interpreted and the Financial Transaction Record has been created, the FTR is routed to the Transaction Processor for processing. The destination, the London branch of the bank, has an account in US dollars in NY, and is credited with the $35 million. Before crediting an account, the Transaction Processor performs other operations such as, check for availability of funds, check authority to debit account, etc. This checking is done by the Transaction Processor against the database that contains all the information. When the Transaction Processor is done, it creates an accounting record that is written in the database.

6. After the transaction has been processed by TP, (i.e. the money has been credited to the London account), a message is sent through SWIFT to the London branch of the bank indicating that its account in NY has been credited with $35 million to be paid to Company B. The SWIFT interface application which belongs to MP, acknowledges the reception of the message by recording it in the database.

7. The bank in London receives the message advising for the credit in its account and notifies Company B about the payment.
Figure 12
Funds Transfer transaction through FTS system
6.2 ANALYSIS OF THE DEMAND DEPOSIT ACCOUNT SYSTEM

An analysis of another component of the Financial Institution Systems, the Demand Deposit Account System (DDA) follows [10]. Before evaluating the current implementation of DDA, a brief evaluation of the system under the previous architecture is made.

6.2.1 THE DEMAND DEPOSIT ACCOUNT SYSTEM UNDER THE 1970s ARCHITECTURE

The Demand Deposit Account System in the 1970's is illustrated in figure 13 and presents the following characteristics [10]:

1. The DDA system is divided into three applications, STAAR II, STAAR DFI, and STAAR. STAAR handles the overseas branches, STAAR II handles the International Financial Institutions, and STAAR DFI handles the Domestic Correspondent Banks. STAAR II and STAAR DFI are functionally the same. However, STAAR DFI was added to DDA because STAAR II did not have the capacity to handle the additional volume.

2. The hardware chosen were minicomputers. Most minicomputers were PDP 11s. Each DDA application was hosted in a minicomputer making a total of seven minicomputers to host all the DDA applications.
3. The DDA applications were developed using the operating environment RSTS/E.

4. Applications used non-standard protocols for communications.

The problems this architecture posed were numerous:

1. The DDA systems were often unable to finish the processing on time causing delays in delivering information to customers (e.g., to provide daily balance updates), and to other systems, (e.g., to Funds Transfer System). The activities that posed the greatest problem were: 1) Delays in tape hands-off to Cash Management, and Funds Transfer; 2) Delays in electronic transmissions to SWIFT and the bank switch; and 3) Inability to provide on-line availability for more than 12 hours since the processor was used for intense batch processing.

2. The system also presented functional bottlenecks. The system did not allow for: 1) Adding additional on-line links which would have brought transactions into the system at an earlier time. 2) Adding additional on-line functions which would have allowed more efficient inquiries. 3) Adding additional off-line functions which would have improved the report generation. These limitations were caused by the hardware and software environment. For example, the PDP 11 limited the maximum allowable program size making it
Figure 13
DDA system - 1970s architecture

69
difficult to add functionality to individual on-line and off-line programs. Furthermore, the limited processor capacity of the PDP 11 (0.5 MIPS), made it very difficult to add functionality since the processors were used up to capacity. Finally, the limitations of the operating environment RSTS/E, made it difficult to add functions to the on-line systems.

3. The DDA applications ran in three sites and involved seven machines. These sites were in different floors causing the duplication of functions which translated into inefficient use of resources, increasing operational costs, and increasing costs for maintenance of hardware and software.

4. It was very difficult to integrate the DDA application with the Funds Transfer system in real time since neither the PDP/11 nor RSTS/E could enter a VAX cluster and thus no data sharing could occur.

6.2.2 THE NEW ARCHITECTURE FOR DEMAND DEPOSIT ACCOUNT SYSTEM

In the new DDA system [10], there were changes in the architecture of the system but no reinvention or redesign of the applications was made. The new Demand Deposit Account System development was divided into five stages. In the first stage, the STAAR II and STAAR DFI were integrated into one
system and converted to the VAX architecture. The on-line processing capability of DDA was transported to a VAX/785, and the off-line processing capability was transported to the VAX 8600 (see figure 14). These two processors are more powerful than the PDP/11 in terms of MIPS. This is an important measure since most DDA applications are batch, and CPU intensive. Moreover, although DDA programs have reached their maximum allocable size in the PDP/11, they can grow as large as needed in the VAX processors. The software chosen was DEC's VMS operating systems. The services offered by the VMS operating system are more powerful than those offered by RSTS/E making it easier to add more functionality when needed. The applications were to migrate using the same programming language, BASIC. In the second stage, optimization of the off-line part of the system was to be done if necessary.

The last three stages are long term. Stage 3 will provide integration of the STAAR system to the new DDA architecture. Stage 4 will provide integration with the Funds Transfer System by integrating the on-line segment of the DDA system with the Funds Transfer System. This implies moving posting and account balances to the Transaction Processing function of FTS; and moving the communication interfaces to other banking groups to the FTS Message Processor. Finally, stage 5 will consist on moving the historical inquiries and batch processing to the Information Processing environment.
The DDA system under the new architecture

First two stages
The operational and functional advantages of the DDA as a result of implementing the first two stages (i.e., integrating and rehosting the STAAR II and STAAR DFI applications into VAX processors) are:

1. Batch processing runs 30\% to 60\% faster.
2. Applications are able to handle substantially larger volumes.
3. It is easier to add programs and functions to both the on-line and off-line systems.
4. Two VAXes replace 7 PDP/11s.

Therefore, the following advantages are obtained:

1. Electronic and tape hands-off communication to other systems will be delivered earlier.
2. On-line inquiries will be available 20 hours per day.
3. The monthly statements will be available earlier.
4. Reduction in hardware maintenance costs, physical plant requirements, and operations support.

These functional and operational advantages are derived from using more powerful processors and a more flexible operating environment, and from integrating several applications into less hardware.
The new implementation of the Transaction Investigation System involved three major tasks [8]. First, the system was rehosted into a VAX cluster using the VMS operating system. A VAX 8600 was used to support the database, and two VAXes 11/785 were used for the application programs. Second, the ORACLE Data Base Management System was used to manage the database namely, the historical database (HDB). The HDB which is a 20 gigabyte database, contains a history of the funds transfer transactions for the previous 90 days and holds approximately 40 million records. Moreover, the investigation itself also generates a history that is also kept in the historical database. Third, the communication facilities for this system were standardized to using DECNET/ETHERNET. The system was designed to support 160 simultaneous users having read-only access to the historical database with an expected response time of 5 to 7 seconds.

The study for evaluating the database management system to be used for implementing the historical database was done in 2 months. This is a typical approach within the bank given its philosophy for obtaining immediate results.

Some problems were found during the development of the system which could have been predicted if an up front study had been carried out. For instance, it was discovered at the system
development stage that the ORACLE "money" data type was not big enough to hold the amounts needed by the bank. Also, that the recovery and re-indexing of the database, which was very large, was taking too long. Therefore, tables had to be partitioned by dates, and applications had to be written to access the database across ranges of dates.

6.4 EVALUATION OF THE FINANCIAL INSTITUTION SYSTEMS USING THE CIS CONCEPTUAL MODEL

The Funds Transfer, Demand Deposit Account, and Investigation systems are evaluated from the integration, autonomy, and evolution perspective. The integration capabilities of these systems (i.e., interaction among them) will be evaluated further in chapter 8.

A. INTEGRATION

In terms of integration, the current implementation of FI systems presents several concurrences with the conceptual model. Some of these represent big improvements compared to the previous architecture:

1. The hardware and operating environment across all systems was standardized. That is, the use of DEC VAXes and VMS operating system.
2. The Communication network (DECNET/ETHERNET) match the Message Control component of the conceptual model, and it allows inter-computer communications. The communication network enables transmission of transactions among the different Funds Transfer components, namely the Message Processor, Transaction Processor, and Exception processor. It also allows communications among the different Financial Institution systems. Furthermore, the communication network represents a single gateway for all applications, thus providing the capability to connect to any application from any terminal.

3. The Message Processor of the Funds Transfer system matches the External Interface component of the conceptual model. All the interfaces to the payment networks are hosted in one processor and the interfaces to message networks are hosted in another processor. This provides several advantages. There is no duplication of interfaces. The external interface is independent of the applications. And, it provides a single entrance for all external entities to the Funds Transfer systems.

4. The Exception Processor component of the Funds Transfer system that used to be segregated by applications is now shared by all the banking groups. It also has access to the unified funds transfer database. An expert system that supports the Exception Processor functions has functionality
to format messages received from the different external entities. The exception processor is consolidated on one machine which allows the sharing of functions that are common for different FTS applications.

5. The integration into one system of the two STAAR applications provides operational advantages. Since two VAX's will replace 7 PDP/11, there will be reductions in hardware maintenance costs, operation support, and physical plant requirements.

The variances from the conceptual model are:

1. A unified funds transfer database has been implemented which contains all the data that used to be segregated in the CIMs databases and that is now shared by the message processor, the transaction processor and the exception processor. However, this database is independent of all the other Financial Institution systems, and no sharing of data occurs between them. For example, the Funds Transfer and Letter of Credit systems are both part of the Transaction Processor. Even though in the conceptual model they are shown as sharing a database, no sharing is occurring at the present time, and each of them has its independent database.

2. There is no match with the Data Control component of the conceptual model. The Funds Transfer and Demand Deposit
Account systems have been implemented without Database Management System (DBMS) technology. Even though it is argued that a DBMS could not provide the high performance and recoverability needed by the Funds Transfer System to manage the transaction queues, DBMS technology could still be used to manage the dynamic files, (e.g. account balances) which would provide further advantages, especially for its integration with the Demand Deposit Account system. The Investigation system is the only one among the ones evaluated here, that uses DBMS technology, but this database is not shared by any other system.

3. The DDA system still communicates with other FI systems via tape hands-off. For example, a daily update of balances is sent to the Funds Transfer, Cash Management, and Letter of Credit systems.

Thus, even if there have been some gains in terms of integration, the main obstacle preventing the integration in the current implementation, is that the different systems have independent databases, and no DBMS technology is used.

B. AUTONOMY

The Funds Transfer and Demand Deposit Account systems have been implemented independently of each other and of the other FI systems. Furthermore, the Funds Transfer system and Demand Deposit Account implementation have been divided into
independent projects to permit, ease of implementation, obtaining immediate benefits, and accountability to individual managers. This matches the culture of the bank.

Even though in the implementation of the funds transfer system there has been a certain degree of centralization, (e.g., one processor for all CIMs, centralization of external interfaces, etc.), the banking groups still have logical control. Each of the banking groups have their own group of operators to perform data entry, repair, and verification of messages. Thus, even though all of the banking groups share the FTS Exception Processor, and there is a single queue containing all the messages to be repaired, each message will be repaired and verified only by the appropriate operator. From the point of view of the operators, their functions are performed as if each banking group had its own independent Exception Processor. Moreover, the terminals and operators are located in the banking group offices, and do not belong to the Financial Institution systems group.

The funds transfer system is an example where the hardware, the operation of the data center, the software development and maintenance, and the data control have been centralized but the business divisions still retain the management and control of the business operations. Thus, accomplishing both physical centralization and logical decentralization.
Applications have been hosted in independent hardware which follows the principle of autonomy. However, by "clustering" the applications in a VAX cluster, some integration is achieved, as it is the case with the Funds Transfer System.

Finally, even if the conceptual model calls for a shared database, the FTS and DDA have independent databases. This particular case illustrates that the bank's autonomy principle which emphasized small projects outweighed the need for integration.

C. EVOLUTION
The new Funds Transfer system is better prepared to accommodate evolution. The single Transaction Processor and unified database provide flexibility if organizational changes need to be made in the bank which have an effect on the way customers are distributed by banking groups. The communication facility for inter-computer communication, DECNET/ETHERNET, has much more unused capacity allowing substantial communication growth with current technology.

The implemention of the Funds Transfer System in a DEC VAX cluster enables the capacity of the system to be increased by adding new hardware (e.g. processor, disks) to the VAX architecture. This can be accomplished without segregating the database since a single database can be shared by all the processors within the VAX cluster. For example, if the
processing capacity of the transaction processor (CIMs) is insufficient, a new VAX processor could be added to the cluster which would share the database. In the case of the Demand Deposit Account systems, capacity can be augmented by adding new hardware to the VAX architecture which would have been impossible with the previous PDP/11s. Moreover, programs can also be expanded for added functionality since there are no limitations regarding the maximum size of a program.
In this chapter, an analysis of the Financial Institution systems is made using the CIS methodology. The CIS model for Financial Institution systems is illustrated in Figure 15. This model proposes a top down methodology where the strategic goals are first identified, then the composite information systems are analyzed focusing on technical and organizational obstacles that constrain the goal of integration. These obstacles are further evaluated in the next chapter. Finally, in chapter 9 alternative recommendations are made to attain the integration goal.

7.1 STRATEGIC GOALS OF FINANCIAL INSTITUTIONS

COST LEADERSHIP

According to Michael Porter [11], one way to gain competitive advantage is by being the cost leader in the industry. The demand for Financial Institution services has been increasing in the last years, thus the objective is to capture market share in the continuously growing market of Financial services. In order to achieve this, the Financial Institution Systems must be able to handle more volume without a corresponding cost increase. Several indications to
**COST LEADERSHIP**

- REDUCE OPERATIONAL COSTS
- REDUCE DEVELOPMENT OR MAINTENANCE COSTS

---

**CIS Model for Financial Institution Systems**

**Figure 15**

CIS Model for Financial Institution Systems
illustrating the increase in the demand of Financial services are: The number of funds transfer operations through CHIPS has been increasing at a rate of about 25% per year; there has also been an increased traffic of SWIFT messages for inter-bank customer balance and transaction reporting; and the volume of FED funds transfers which require immediate confirmation has increased considerably [9]. The systems objective is to accommodate this growth without incurring high operational, personnel, or maintenance costs. These costs could be controlled in the following areas:

1. **Operational costs.**

   This category refers to the cost involved in running the data centers, the data entry, the verification functions, as well as tape hands-off. If these functions require a lot of manual intervention, the costs involved handling the increasing demand would increase proportionally with demand. For example, if confirmation of transactions were done manually because the dispersity of data prevents the possibility of real-time checks, the number of people required to handle the confirmations would increase as the demand grows.

   An example where the bank has already taken action to reduce operational costs is found in the Funds Transfer System. An expert system was developed to support the exception processor functions. About 25% of the messages processed by FTS need some kind of manual intervention in order to be interpreted.
and formatted. This 25% represents an average of 16,250 messages per day. The expert system helps to reduce the number of people needed to perform this task by doing the interpretation automatically. The task of the operator is then limited to checking the output produced by the expert system. This reduces dramatically the time required per unstructured message. Even if a high price was paid to develop the expert system, the savings in operational costs are well worth the additional initial expense.

Another example where costs can be reduced is by the elimination of tape hands-off to communicate across systems by merging redundant databases, or by using the communication network to communicate between systems. Not only do tape hands-off involve manual intervention, but also setting up procedures to handle it, and the possibility of introducing errors.

2. Development and/or maintenance costs.

There is a trade-off between initial development costs and maintenance costs. If systems have been developed to accomodate growth, and change, the up-front cost of development will most likely be higher whereas reducing the future cost of maintenance. Several examples are provided to illustrate this point. If the databases are dispersed, changes need to be propagated as they occur. Thus, procedures need to be set up and carried out to either make the change real time
or off-line as needed. If database technology is not used, a change in the data implies changing all the applications that use those databases. Therefore, changes to the software are needed. Moreover, whereas the price of hardware has been going down in the last years, the price of developing and maintaining software has shown an increasing trend.

7.2 CIS - THE FINANCIAL INSTITUTION SYSTEMS

As it was described in chapter 6, the Financial Institution Systems are still very independent of each other. So, although a common communication structure has been developed, most of them still have their independent databases which contain redundant data, thus constraining the goal of integration.

7.3 TECHNICAL OBSTACLES

The main technical obstacles towards integration that are found in the current implementation of Financial Institution systems are:

1. High level of autonomy.

Even though a strategic systems plan was developed in 1982 which resulted in the conceptual model discussed in chapter 5, most of the Financial Institution systems have still been developed independently. An example is provided by the Funds
Transfer system, and the Letter of Credit system. Although they both belong to the Transaction Processing component of the Conceptual model, they were developed independently resulting in two independent databases.

An example found in the current implementation where the principles of autonomy and integration are mediated is given by the implementation of the Funds Transfer system in a VAX cluster. The applications belonging to the Funds Transfer system have been hosted in independent hardware. However, these applications were grouped together in a VAX cluster which provided the ability to share data. The use of VAX clusters which has a maximum capacity of 16 nodes (i.e., processors, disk and communication controllers, etc.) allowed for independent hardware to be used for different applications. However, as more applications are added to this cluster, or as demand for FTS products grows, the capacity of the cluster may be exhausted, thus posing a technical problem since with the current implementation, data can only be shared within a cluster.

2. **High level of flexibility.**

The Financial Institution systems require a high level of flexibility. By keeping the systems small, higher flexibility is obtained but it presents problems for interconnecting these "small" systems. However, compared to the previous architecture, there has been a certain level of aggregation
providing gains in terms of integration capabilities.

3. **Segregated databases.**
The Financial Institution systems present the inefficient characteristic of having disperse and redundant databases. Each system has an independent database, even though some of the data is common across systems. This dispersity of databases represent the main obstacle towards integration. It also poses other problems, such as capacity and performance of the network and processors, duplicate processing and maintenance, etc. These problems are discussed in the next chapter.

4. **Lack of use of Data Base Management Systems technology.**
The lacking of DBMS technology in the current implementation has consequences on the maintenance of the databases. If no DBMS is used, and databases are shared among applications, update propagation, security, and consistency of the data needs to be controlled by the application. Moreover, changes to the database implies changing all the applications that access it. Note however that, for performance reasons, the use of databases is not always feasible in the bank. This represents a technical obstacle that can only be resolved with future advances in DBMS technology.
7.4 ORGANIZATIONAL OBSTACLES

The main organizational obstacles towards integration found in the current implementation of FI systems are:

1. **High level of local autonomy.**
The organizational structure of the bank is highly decentralized. Individual managers are given complete autonomy to make decisions, and operate their units. Consequently, managers are also fully accountable and responsible for meeting goals. Thus, they are reluctant to relinquish the control over their operations to other organizations in the bank. An example to illustrate this point is provided by the development of the historical database (HDB), and the MIS system. These two applications, which started at the same time, ended up going in different directions [8]. Due to the lack of communication and trust among these two groups, the MIS system ended up obtaining this information from many different applications. In the process having to perform several layers of aggregation. Organizational obstacles prevented the MIS application from taking some of the aggregated data from a single source, the historical database. This in turn led to duplication and inefficient resource allocation.

2. **High level of local flexibility.**
Given the decentralized philosophy of the bank, individual
managers want to have the maximum amount of flexibility to accommodate change and growth.

3. **Immediate results.**
The culture of the bank is to engage in projects that provide immediate benefits. This has a disadvantage for implementing any long term plan that would provide benefits in the long run, such as planning for ease and reduction of future maintenance costs versus initial development costs.

4. **Budget constraints.**
Budget constraints jeopardize the goal of integration since given the dispersity of databases, their integration may require a high up-front cost. The goal of the business divisions is to get the work done. This leads to inefficiencies in the implementation and operation of systems. An example where considering only immediate results, and operating under budget constraints had a negative effect, is in the lack of integration of the data. Sharing the data that is used by several applications is a very desirable goal. This however, would have involved an initial high cost which under these constraints was impossible.

After reviewing the CIS concept and applying it to the Financial Institution Systems, the following chapters will analyze and discuss the technical obstacles in the current implementation, and some alternatives for future directions.
Chapter 8
Evaluation of the current implementation
of Financial Institution Systems

After having analyzed the systems composing the Financial Institution Systems in chapter 6, this chapter presents an analysis of the integration capabilities across the FI systems using the CIS methodology.

8.1 INTEGRATION CHARACTERISTICS OF FI SYSTEMS

The current implementation of the Financial Institution Systems present the following characteristics:

1. **Common communication facility.** The centralized network facility (DECNET/ETHERNET) enables the exchange of messages across Financial Institution's systems, i.e., Funds Transfer, Letter of credit, Cash Management, Investigations, and Demand Deposit Account. For example, a funds transfer operation which generates funds or receives incoming funds for a bank's customer, produces one or more transactions that are automatically routed to the proper application to either credit an incoming fund or debit the account for a funds transfer.

2. **Segregated databases.** All the systems, Funds Transfer, Cash
Management, Demand Deposit, and Investigations have their independent databases as illustrated in figure 16. All these databases contain redundant information, i.e. some databases are partially a "shadow database" of the original database. The redundant information in the "shadow databases" includes, static customer information such as, customer name, account number, etc, and dynamic information such as, customer account balances, authorizations, etc.

8.2 PROBLEMS POSED BY CURRENT IMPLEMENTATION OF FI SYSTEMS.

The main problem posed by the current implementation is the dispersion and redundancy of data which generates "shadow databases." A discussion of the implications that these "shadow databases" have on performance, consistency of data, systems development, and systems maintenance is presented in the next section. All of these have an effect on the cost of operation, and maintenance of the Financial Institution Systems.

In the Financial Institution Systems, updates are classified as either real-time updates or off-line updates. In some cases, "shadow databases" require real-time updates which implies that a change in one database needs to be propagated to the "shadow databases" as soon as it occurs. In other cases, changes to a database do not require to be propagated in real time, but they can be done off-line at a later time.
Figure 16

Applications with independent databases

93
In the latter case, procedures need to be set up and carried out on periodical basis. However, either case requires some kind of communication among the Financial Institution Systems to propagate the updates. It can vary from sending a message through the network in real time to tape hands-off.

8.2.1 REAL-TIME UPDATES

An example of a real time update is given by the funds transfer operation which spawns messages to other Financial Institution systems with the sole purpose of updating "shadow databases". There are two types of funds transfer transactions: 1) Transactions that change the bank books since one of the players in the funds transfer is a customer of the bank. And 2) Those transactions that do not change the bank books since the bank is acting as an intermediary to other bank’s customers.

Currently, each funds transfer transaction which generates funds or receives incoming funds for a bank’s customer, (i.e., requires a change on the bank books), spawns at least two messages to other Financial Institution systems. These messages are called "shadow postings". The "shadow postings" generated by a Funds Transfer transaction is illustrated in Figure 17. The messages spawned by FTS are:

- Two (2) posting messages to the DDA system.
- One (1) posting message to the Cash Management system, in
Funds transfer transaction
(DDA and CM customer)

Communication network

1 CM posting message
2 DDA posting messages

Figure 17
Funds transfer transaction generating posting messages
the case that the customer is also a Cash Management customer.

The sending of messages is necessary since each application keeps in its independent database a record of the customer status (e.g., the account balance,) and any change to the customer status needs to be propagated in real time to all "shadow databases" (e.g., Demand Deposit Account and Cash Management databases). However, the bank also acts as an intermediary for other banks; in those cases, it is not necessary for the funds transfer transactions to send a message to either the DDA or CM systems.

Assuming that 50% of funds transfers are for other banks, in average, for every funds transfer transaction processed, 1 1/5 transactions are spawned to other FI systems. The average 1 1/5 transactions spawned by the Funds Transfer system can be broken down into two components: 1) One (1) posting message transaction to the Demand Deposit Account System; 2) 1/5 posting message transaction to the Cash Management System (i.e., in average, 20% of the funds transfer transactions are for Cash Management customers).

Thus, given that the average number of funds transfer transactions per day is currently 60,000 transactions, the average number of messages (i.e., shadow postings) spawned by the Funds Transfer system in real time to update "shadow
databases" is:

- 60,000 messages to the Demand Deposit Account System;
- 12,000 messages to the Cash Management System.

8.2.2 OFF-LINE UPDATES

Other "shadow databases" updates are not required in real time. Some examples in this category are, the maintenance of static customer information such as, addition of a new customer, change in customer address, and the reconciliation of customer balances. The latter is done daily by DDA, and produces a tape that is sent daily to all FTS, CM, and LOC. Thus, procedures need to be set up to propagate the updates as frequently as necessary. This usually requires manual intervention, e.g. tape hands-off.

8.3 IMPLICATIONS OF THE CURRENT IMPLEMENTATION OF FI SYSTEMS

The current implementation has implications on the performance, maintenance, and operation of the FI systems. These are summarized in figure 18.

1. Network Performance.

In average, there are 72,000 messages flowing across the network with the only purpose of propagating updates to
<table>
<thead>
<tr>
<th>Issues</th>
<th>Current implementation</th>
</tr>
</thead>
</table>
| Message posting      | 60,000 from FTS to DDA  
                      | 12,000 from FTS to CM                                      |
| Processing overhead  | Considerable                                                |
| DDA performance      | Good                                                        |
| CM performance       | Good                                                        |
| Tape hands-off       | 1 tape from DDA to:  
                      | FTS  
                      | CM  
                      | LOC                                        |
| Off-line updates     | Send transaction file from FTS to Investigations             |
| Duplicate maintenance| Maintenance of Static DB in:  
                      | FTS, DDA, CM, LOC, and Investigations systems             |
| Duplicate Information| Dynamic and Static DB in:  
                      | FTS, DDA, CM, and LOC                                      |

**Figure 18**

Current implementation issues
"shadow databases". The flow of messages across systems is proportional to the number of funds transfer transactions. Thus, as the number of funds transfer transactions is growing at a rate of 25% per year, the flow of messages will increase at the same rate. This increasing number of messages flowing through the network, may have a negative effect on the capacity and performance of the network.

2. Processing overhead.
In order to propagate updates to "shadow databases" in real time, the applications need to have software (i.e. application's code) to make decisions on whether or not to send a message to other applications, and to receive the messages sent by other applications. For example, for any funds transfer transaction processed, decisions are made in the FTS application program on whether to send a message to either DDA, or to CM, depending on the customer status. If the bank books need to be changed as the result of a funds transfer operation, the appropriate posting messages need to be constructed and be sent to the appropriate application(s). All these require the use of processing time to perform functions which are not relevant to the funds transfer operation. It also requires the development and maintenance of the additional software to handle the sending and receiving of messages.

Taking into account that the average number of funds transfer
transactions is about 60,000, this could mean a lot of processing time with the solely purpose of propagating updates. On the other hand, the DDA and CM applications need to have code to receive and process the update messages that are sent by FTS.

3. **Tape hands-off.**

Some update information is transmitted via batch. At night, posting and balance updates are sent via tape from the Demand Deposit Account system to the Funds Transfer system, the Cash Management system, and the Letter of Credit system. To perform this updates via tape hands-off implies that procedures need to be set-up (i.e., development of software), and carried out which involves the intervention of an operator.

4. **Off-line update propagation.**

Every night, an ASCII file with the transaction history of the day is sent via the communications network from the Funds Transfer System to the Transaction Investigation system to update the Historical Database. However, this seems like a plausible approach since the Funds Transfer database contains only the transactions that are currently being processed (i.e. today's transactions) whereas the historical database contains a transaction history for the 90 previous days. Updating the historical database real-time would generate a tremendous amount of messages that would not provide greater benefits.
On the other hand, the historical database is for obtaining information rather than processing transactions, so this application belongs to the Information Processing component of the conceptual model described in chapter 5. Sharing the historical database with the Transaction Processing database is not possible because first, these databases serve different purposes, and second, the VAX cluster has a maximum of 16 nodes and not all of the Financial Institution systems can fit in the same cluster.

As the transfer of daily transactions from the FTS database to the historical database is done daily at night, the historical database contains information up to the previous day. Thus, if there is any investigation that requires looking at today's data, the AIRS operator will have to query the Funds Transfer database.

5. **Duplicate maintenance.**

The databases contain other redundant information which is not required to be updated in real-time. However, procedures need to be set up and carried out to propagate any change made to the original database duplicating the database maintenance. For example, if there is a change in static information such as, a customer address is changed, or a new customer is added to the database, a procedure is needed to propagate this change to "shadow databases". In these cases, it is usually a manual intervention to either start a procedure or to perform
the change manually across the databases. Currently in the bank, all the changes to static information are performed manually by an operator. Moreover, since the change needs to be transmitted to the disperse databases, this information is manually entered into all the systems that use this information, (i.e., Funds Transfer, Demand Deposit Account, Transaction Investigation, and Cash Management). Operators belonging to different divisions will perform the same function to enter the new information in the different databases. Furthermore, this operation is prone to errors.

6. **Duplicate information.**

The Funds Transfer system besides having redundant information about the customer, name, address, account balances, etc, needs to have information about which customers are DDA and which are CM customers. This implies further use and waste of storage.

7. **Integration of FTS and Letter of Credit.**

The transaction processor in the conceptual model is composed of two systems: the Funds Transfer and the Letter of Credit system. In the conceptual model, these systems share the same database but in the current implementation, they both have independent databases. Thus, information about funds transfers is not known to the letter of credit system until the next day, and viceversa.
All these factors, processing overhead, tape hands-off, duplicate maintenance, etc. have an effect on the cost of operation and maintenance of the FI systems.

8.4 COSTS OF OPERATION AND MAINTENANCE OF THE REDUNDANT DATABASES.

The total costs of operation and maintenance of the redundant databases, namely the dynamic database, and the static database can be quantified in terms of, processing costs, personnel costs, storage, etc. and they are shown in the cost function below. The cost of operating a system is shown as:

\[ C_0 = C_{om} + C_{others} \]  \hspace{1cm} (1)

The cost of operation and maintenance is given by the following equation:

\[ C_{om} = F(P, T, OL, DM, DI) \]
\[ = P + T + OL + DM + DI \]  \hspace{1cm} (2)

where

\( P = \) processing overhead,
\( T = \) tape hands-off,
\( OL = \) Off-line updates through network,
DM = Duplicate maintenance, and
DI = Duplicate information.

This cost function is additive, since in the current implementation, all these costs are present. Moreover, the dynamic database is partially redundant in the Funds Transfer system, the Demand Deposit Account System, the Cash Management system, and the Letter of Credit system.

The definition of P, T, OL, DM, and DI follows,

1. Processing overhead (P). Processing overhead is the amount of processing spent on exchanging messages to propagate updates in real time. Processing overhead is a concern since it constraints the ability to accomodate growth without having to migrate to a more powerful processor. Also, removing processing overhead provides the capability to have processing slack that can be used to host other applications, thus using resources more efficiently.

\[
P = P_f + P_d + P_c
\]

\[
P_f = \left[ (N_f * t_{f1}) + (N_f * (N_d + N_c) * t_{f2}) \right] * k_p
\]

\[
P_d = (N_f * N_d * t_d) * k_p
\]

\[
P_c = (N_f * N_c * t_c) * k_p
\]
where

\( P_f = \) Processing overhead in Funds Transfer System.

\( P_d = \) Processing overhead in Demand Deposit Account System.

\( P_c = \) Processing overhead in Cash Management System.

\( N_f = \) Total number of funds transfer transactions which are processed daily. This includes those transactions that spawn messages to other applications, and those transactions that do not spawn any message.

\( N_d = \) Percentage of funds transfer transactions which spawn a message to DDA. The current average is 100\%, thus \( N_d \) would be equal to 1.

\( N_c = \) Percentage of funds transfer transactions which spawn a message to CM. The current average is 20\%, thus \( N_c \) would be equal to 0.20.

\( t_{f1} = \) Units of time required by FTS to decide whether a particular transaction requires a message to be spawned.

\( t_{f2} = \) Units of time required for FTS to build and send a message to another application.

\( t_d = \) Units of time required by DDA to process the receiving message and update its database.

\( t_c = \) Units of time required by CM to process the receiving message and update its database.

\( k_p = \) Average cost (in dollars or fraction) per unit of processing time.

2. **Tape hands-off \((T)\).** Tape hands-off refers to the communication via tapes among FI systems to propagate updates which are not required in real time.
\[ T = A \times n_t \]  

\[ A = (n_A \times t_A \times k_A) + (t_t \times k_p) \]  

where

\( n_t \) = Number of tapes to be exchanged among FI systems.

\( n_A \) = Number of operators needed to handle a tape. For example, if a tape is sent to three different systems located in different sites, four operators would be needed.

\( t_A \) = Units of time required by an operator to handle one tape.

\( t_t \) = Units of processing time required to process one tape.

\( k_A \) = Average cost (in dollars or fraction) per unit of operator's time.

\( k_p \) = Average cost (in dollars or fraction) per unit of processing time.

3. **Off-line updates (OL).** Off-line updates refers to the exchanging of files across FI systems (through the network) to propagate updates which are not done in real time.

\[ OL = \sum_{i=1}^{N} B_i \times [ (t_1 + t_2) \times k_p ] \]  

where

\( N \) = Number of files to be transferred across FI systems.

\( B_i \) = Number of blocks of file \( i \).
\( t_1 \) = Units of processing time required by the originating application to create and transfer one block.

\( t_2 \) = Units of processing time required by the destination application to receive and process one block.

\( k_p \) = Average cost (in dollars or fraction) per unit of processor time.

4. **Duplicate maintenance (DM).** Duplicate maintenance refers to the manual maintenance of redundant databases. One example is provided by the maintenance of static information which is manually done by operators in each one of the databases.

\[
DM = M \times n_b \times n_c \times (1 + p)
\]  

\[
M = (t_M \times k_M) + (t_c \times k_p)
\]

where

\( n_c \) = Number of changes to be performed to one database/day.

\( t_M \) = Units of time required by one operator to make one change.

\( t_c \) = Units of processing time required to process the change.

\( k_M \) = Average cost (in dollars or fraction) per unit of operator's time.

\( k_p \) = Average cost (in dollars or fraction) per unit of processing time.

\( n_b \) = Number of databases to be updated. It is assumed that one operator is required per database. Note that as databases may belong to different divisions, different
operators perform the change in each of the databases.

\[ p = \text{The probability of making at least one error while performing a change in any of the databases. This probability has been calculated to be } p \ (1). \]

4. **Duplicate information (DI).** Duplicate information refers to the cost incurred on storage in order to maintain redundant databases.

\[ \text{DI} = (n_b - 1) \times B \times k_s \ (7) \]

where

- \( n_b \) = Number of databases that contain redundant information.
- \( B \) = Number of blocks.
- \( k_s \) = Average cost (in dollars or fraction) per block of storage.

This model can be used to compute the cost of maintaining and operating the current implementation which contains redundant databases.

---

(1) Assume that we have \( n \) operators. The probability of one operator making an error is \( p \), being \( p \) very small. Then, the probability of having one operator making no errors will be \( (1-p) \). Thus, the probability of having no errors by any operator will be \( (1-p)^N \). Therefore, the probability of having at least one error will be \( 1 - (1-p)^N \), which is approximately \( np \) when \( p \) is very small, as can be seen by using the binomial expansion.
In sum, the current architecture presents inherent drawbacks given that data is disperse and belongs to each application. Manual procedures and tape hands-off are still needed to accomplish propagation of updates. For real time updates, a continuous flow of messages is needed to transmit messages across applications. As can be seen in equation (3) above, as the number of funds transfer transactions increase, the processing overhead would increase linearly. All these represent costs that could be diminish with other approaches to be explored in the next chapter.
Chapter 9
Future Directions

As it was discussed in the previous chapter, the main drawback of current Financial Institution Systems is their disperse and redundant databases which require either manual procedures or a continuous flow of messages to propagate updates.

An alternative to the current implementation is to have a single database containing all the information that is otherwise redundant across the databases, or to have distributed databases that can be accessed by systems located in different hardwares. The consolidated database would be accessed by all the systems that need the information contained within it. Two approaches are then possible; one is to consolidate the data without sharing it, so that requests and updates will be done by exchanging messages across systems. The second approach is to share the databases that contain redundant information. The different alternatives considered are explained below.

9.1 CONSOLIDATION OF DYNAMIC DATABASE

This approach calls for the consolidation of information that is redundant across databases. In particular, the database
containing dynamic information (e.g., account balances, authorizations, etc.) would be consolidated in one of the FI systems. One of the following two systems can be chosen as the host to consolidate this database: 1) the Demand Deposit Account System, or 2) the Funds Transfer System. The host system is the only one owning the database. The other FI systems would not share the consolidated database, but instead they would access it by generating a message directed to the host system to either request information, or update the database.

In this case, the Demand Deposit Account system has been chosen as the host for dynamic information. The main reason to consolidate the dynamic database in this manner is the very nature of this data. The DDA system produces periodic reports which are mainly batch on balance information, a lot of them being produced daily. Having this information in the Funds Transfer System would imply a continuous stream of messages that would degrade the DDA processing and would delay its delivery schedules. As it was mentioned in chapter 6, meeting deliverables on time are critical for DDA. Figure 19 provides a summary which compares the merits of the consolidation of Dynamic data in either DDA or FTS to the current implementation.
<table>
<thead>
<tr>
<th>Issues</th>
<th>Current implementation</th>
<th>Consolidated DB in DDA</th>
<th>Consolidated DB in FTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message exchange and shadow posting</td>
<td>From FTS: 60,000 to DDA 12,000 to CM</td>
<td>&gt;60,000 from FTS to DDA 6,000 from CM to DDA</td>
<td>None from FTS or DDA 6,000 from CM to DDA</td>
</tr>
<tr>
<td>Processing overhead</td>
<td>Considerable</td>
<td>Considerable</td>
<td>Less</td>
</tr>
<tr>
<td>DDA performance</td>
<td>Good</td>
<td>Good</td>
<td>Bad</td>
</tr>
<tr>
<td>CM performance</td>
<td>Good</td>
<td>Less good</td>
<td>Less good</td>
</tr>
<tr>
<td>Tape hands-off</td>
<td>1 tape from DDA to: FTS CM LOC</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Off-line updates</td>
<td>Send transaction file from FTS to Investigations</td>
<td>Same</td>
<td>Same</td>
</tr>
<tr>
<td>Duplicate maintenance</td>
<td>Static DB in: FTS, DDA, CM, LOC, and Investigations systems</td>
<td>Almost same</td>
<td>Almost same</td>
</tr>
<tr>
<td>Duplicate Information</td>
<td>Dynamic and Static DB in: FTS, DDA, CM, LOC</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

**Figure 10**

Comparison of consolidation of dynamic database vs. current implementation

112
9.1.1 PROBLEMS SOLVED BY CONSOLIDATING THE DYNAMIC DATABASE

1. Eliminating tape hands-off among applications.
The daily posting and balance update that is sent via tape from the Demand Deposit Account system to the Cash Management system, Funds Transfer system, and Letter of Credit system is eliminated since the database would be consolidated in DDA. Thus, tape hands-off that involve manual intervention and setting of procedures are further eliminated from the FI systems.

2. Some Reduction in the propagation of updates.
At the moment, there is a flow of messages throughout the network with the sole purpose of propagating updates. In this alternative approach, inquiry messages will be generated on demand. This is to say, messages would be generated to either request information, whenever needed, or to update the bank books in real-time. A comparison of the flow of messages in the current implementation and in the consolidation of dynamic data in DDA is shown in Figure 20.

A. Exchange of Messages between FTS and Cash Management.
Currently, 20% of the transactions processed by the Funds Transfer System require a message to be sent to the Cash Management System to update its database. This represents an average of 12,000 messages that are sent daily. However, Cash Management only performs inquiries, not updates, and
Funds transfer transaction (DDA and CM customer)

**CURRENT IMPLEMENTATION**

1 CM posting message
2 DDA posting messages

**CONSOLIDATED DATABASE**

2 posting messages
request for balances
request for balance to DDA

Figure 20
Comparison of message exchanging
the average number of Cash Management inquiries is 6,000. With the new approach, the flow of messages from the Funds Transfer system to the Cash Management System to update the CM database will be eliminated, and only the demanded number of messages from Cash Management (i.e. an average of 6,000) will be generated. Therefore, the flowing of messages between FTS and CM is reduced by 50%.

B. Exchange of messages between FTS and DDA. The funds transfer system currently spawns an average of 60,000 messages to DDA. This number of messages would still be necessary since the dynamic database is consolidated in DDA. Moreover, more messages are needed to request information that was previously available in the redundant FTS database. However, a gain is still achieved since posting to the bank books would occur in real time, that means funds transfer and DDA transactions will be checking balances and overdraft authorizations against the real customer balances, (i.e., the bank books). Currently, funds transfer transactions checks are done against the previous day balances.

The two main gains obtained from this approach are, to reduce tape hands-off and duplicate storage, and performing overdraft and balance checking against real-time information.
9.1.2 **PROBLEMS NOT SOLVED BY THIS APPROACH**

On the other hand, the following problems would not be solved by this approach:

1. **Maintenance of static information.**
   Static information, customer names, addresses, etc. would still have to be maintained in the Funds Transfer System. The main reason is for performance. Currently, the Funds Transfer system maintains these tables in real memory for quick access. This information is used by the Message Processor, the Transaction Processor, and the Exception Processor. If the information is consolidated in DDA, FTS would have to send messages to DDA to obtain it, degrading the FTS performance. On the other hand, consolidating this information in FTS would degrade the reporting system that DDA handles.

   However, as the Cash Management system is an on-line system which mainly performs queries on balances, and it will be accessing the consolidated database, it does not need to have the static information locally, so the problem would be partially solved by eliminating one "shadow database", (i.e., the CM static database).

2. **Communication with the Transaction Investigation System.**
   The daily transmission of the transaction history (i.e., to update the historical database) which is sent from the Funds
Transfer System to the Transaction Investigation System would also not be eliminated.

3. **Processing overhead**.

The processing time and code required to send update messages from the Funds Transfer System to the Cash Management System would be eliminated. However, new code would have to be written to coordinate queries from Cash Management to be handled by DDA. The amount of code and processing time to manage the exchange of messages between FTS and DDA will be the same.

Even though the suggested approach does not represent the ideal system, it could be implemented as a transitory stage for the next proposed alternative which calls for a more cohesive integration (e.g., shared data resource) of the Financial Institution Systems.

9.1.3 **COST OF MAINTENANCE AND OPERATION**.

The cost function shown in chapter 8 will be affected as follows,

\[
C_{om} = F(P, T, OL, DM, DI) = P + T + OL + DM + DI
\]
T will be reduced in more than 50% since tape hands-off are dramatically reduced in order to propagate updates.

DI will be reduced in almost 100% since duplicate storage will be almost eliminated.

Therefore, the costs of operation and maintenance are reduced, and other intangible gains such as, checking against the current bank books are also obtained, although they are not quantified in the cost function above.

9.2 **SHARING THE DYNAMIC DATABASE.**

This approach calls for the consolidation and sharing of dynamic information. There are two locations where this data could be kept: 1) as part of the Transaction Processing component; 2) as part of the Information Processing component. The approach to be analyzed now is the first one, to locate the consolidated daily dynamic data as part of the Transaction Processor. The other approach is analyzed under the new technologies section.

With the current architecture, applications need to be in the same cluster in order to be able to share a database, thus by placing applications in the same cluster, sharing of data can
be accomplished. However, only the Funds Transfer system and
the Letter of Credit system are contained within the
Transaction Processing component, and the Letter of Credit
(LOC) system is not part of the cluster. So, DDA, LOC, and CM
would have to be moved into the Transaction Processing
component. The movement of the same-day processing DDA to TP
was included in the long-term Strategic Systems Plan for
Financial Institution systems. However, in this case, the
Cash Management system would also have to be moved into TP in
order to be able to share this database. If CM is not moved
to TP, messages will have to be sent on request in order to
obtain the desired information. This is a plausible
alternative since at the moment there are approximately 6,000
of these requests.

On the other hand, if DDA, LOC, and CM form part of the TP
cluster (currently named, Funds Transfer cluster), the dynamic
database would be consolidated and shared by FTS, CM, LOC, and
DDA. This approach provides several advantages: performing
all balance and overdraft validation against current data;
drastic reduction in the number of messages flowing through
the network, and reduction on duplicate maintenance and tape
hands-off. Figure 21 illustrates this architecture, and
Figure 22 presents a summary of the merits of this approach
compared to the current implementation. An explanation of the
pros and cons of this approach follows.
9.2.1 PROBLEMS SOLVED BY THIS APPROACH

1. Drastic reduction in update propagation.
At the moment, there is a flow of messages throughout the network with the solely purpose of tracking changes for the segregated dynamic databases. In this new approach, no exchange of messages will be necessary for this purpose. Moreover, updates of the bank books and verification against the bank books would be done in real time.

A. Messages from FTS to Cash Management.
   As explained before, the Funds Transfer System sends an average of 12,000 messages per day. This message flow could now be completely eliminated since the Cash Management system would be sharing the database that also contains the bank books (e.g., account balances).

B. Messages from FTS to DDA. The flow of messages from Funds Transfer to DDA, would be completely eliminated since the Demand Deposit Account and the Funds Transfer System will be sharing the same database and FTS will do the postings against this database. Moreover, the funds transfer transactions will be checking overdraft and balances against the actual customer balance.

2. Eliminating tape hands-off across applications.
The daily posting and balance update that is sent via tape
Figure 21
Sharing of daily dynamic database
<table>
<thead>
<tr>
<th>Issues</th>
<th>Current implementation</th>
<th>Share dynamic DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message exchange and shadow posting</td>
<td>From FTS: 60,000 to DDA 12,000 to CM</td>
<td>None</td>
</tr>
<tr>
<td>Processing overhead</td>
<td>Considerable</td>
<td>None</td>
</tr>
<tr>
<td>DDA performance</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>CM performance</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Tape hands-off</td>
<td>Tapes from DDA to: FTS CM LOC</td>
<td>Tapes from TP to: Historic DDA</td>
</tr>
<tr>
<td>Off-line updates</td>
<td>Send transaction file from FTS to Investigations</td>
<td>Same</td>
</tr>
<tr>
<td>Duplicate maintenance</td>
<td>Static DB in: FTS, DDA, CM, LOC, and Investigations systems</td>
<td>Reduced</td>
</tr>
<tr>
<td>Duplicate Information</td>
<td>Dynamic and Static DB in: FTS, DDA, CM, and LOC</td>
<td>None</td>
</tr>
</tbody>
</table>

**Figure 22**

Comparison of sharing dynamic database vs. current implementation
from the Demand Deposit Account system to the Funds Transfer System, Cash Management, and Letter of Credit systems would be eliminated since the data is consolidated in the TP cluster. However, the daily posting and balance update will have to be sent to update the historic DDA database which resides in the Information Processing component. This would be unavoidable since the databases contain different kind of information.

3. **Eliminating duplicate processing.**
The processing time and code required to send update messages from the Funds Transfer System to the Demand Deposit Account and Cash Management System, and the code in DDA and CM to receive the messages would be completely eliminated.

4. **Reduce duplicate maintenance.**
The duplicate maintenance that takes place to update static information would be completely eliminated for all applications within the Transaction Processing. Thus, no duplicated maintenance is needed for CM, daily DDA, and FTS. However, now any change made to static information needs to be reflected in the DDA historical database, and in the Transaction Investigations historical database. This can not be eliminated because the nature of the data contained in these databases is different.
9.2.2 PROBLEMS NOT SOLVED BY THIS APPROACH

1. Cash Management System.
A new problem is created since the historic database will remain in the Information Processing component, and some of the Cash Management inquiries would require to get a transaction history. In this were the case, a message would have to be sent to IP to respond to the transaction history inquiry. The number of requests of this type is currently less than 6,000.

2. Duplicate maintenance for historical databases.
As the historical and dynamic databases are different in nature, some duplicate maintenance will still take place. For example, a daily transaction file needs to be sent from FTS to be the Investigations system to update the historical database. Second, the daily postings made to the dynamic database have to be sent to the DDA historic database. This could be accomplished by sending a file through the network instead of doing it via tape hands-off. Finally, maintenance to the static database has to be performed in both sites, the transaction processing and the information processing. Note that less static databases need to be maintained.

9.2.3 COSTS OF MAINTENANCE AND OPERATION

The cost function shown in chapter 8 will be affected as
follows,

\[ C_{om} = F(P, T, OL, DM, DI) \]

\[ = P + T + OL + DM + DI \]

\( P \) will be reduced by 100% since the number of messages flowing through the system because of update propagation is completely eliminated.

\( T \) is reduced by at least 75% since the number of updates via tapes has also been reduced.

\( OL \) would be the same, and may increase if the daily DDA is sent through the network to the historic DDA.

\( DM \) is also reduced by about 50% due to a reduction in the maintenance of static information.

\( DI \) is reduced by 100% since storage is no longer needed to hold redundant databases.

Therefore, this approach shows to reduce the operation and maintenance costs significantly. The main advantages are, to enable growth with current architecture and technology, reduce personnel costs, and storage costs.
If the dynamic database is implemented without a database management system, a few problems are encountered. First, since different applications are sharing the same database, namely the dynamic database, concurrency has to be done at the application level, i.e., code needs to be written into the applications to perform concurrency control. Second, applications need to know exactly where the data is, so there is no transparency in the access to data. Third, security access to the database has also to be controlled at the application level. Fourth, the benefits that DBMS provides for reporting and screen formatters that speed the development process are not utilized. For all the reasons stated above, and given that the performance required to access this database is well within the limits provided by current DBMS, it is desirable to implement this database using DBMS technology. The use of DBMS will provide advantages for maintenance, and speed of development since the programmer would be relieved of coding tasks that would be taken care of by the DBMS.

9.3 NEW TECHNOLOGIES - DISTRIBUTED DATABASES

A new technology to be explored here is distributed databases. Currently, two commercial packages are available that work in
the VAX environments, Distributed Ingress and Distributed Oracle. This technology could be used for implementing the dynamic database. The distributed DBMS will handle concurrency and security control, and transparency in the access and update of information. Thus, applications do not need to take care of these tasks.

The main gain provided by this technology is that systems do not need to be in the same VAX cluster in order to share the database; databases can be located in different sites, and even in different hardware, and the distributed DBMS is responsible to get access to the appropriate database. Besides getting this new advantage, in this approach all the advantages listed under 9.2.1 still hold, namely reduction of message flow, and elimination of tape hands-off, duplicate maintenance, and duplicate processing. All this have an impact on the reduction of costs.

If this technology is used, two different approaches can be implemented, locate the daily dynamic database in the Transaction Processing component and the historic dynamic database in the Information Processing component; or the consolidation of the daily and historic database in IP. By following either of these approaches, the following additional advantages would be obtained.
9.3.1 PROBLEMS SOLVED BY USING DISTRIBUTED DBMSs

1. Reducing processing overhead.
By implementing distributed databases, a back-end processor can take care of all the database accesses. Therefore the front-end processors, which currently perform front-end and back-end tasks since they handle database accesses and updates (e.g. postings, query accesses, etc) would be further relieved of these tasks. As a consequence, front-end processors would have the capability of accommodating growth without having to add more processing power. For example, the Transaction Processor of the Funds Transfer System would be able to handle a larger volume of transactions without having to add more processing capacity. The extra capacity that would be obtained would be very useful since the demand for funds transfer transactions is increasing at a rate of 25% per year.

2. Share of data outside the VAX cluster.
With the distributed DBMS, applications can access the dynamic database even if they are not part of the same VAX cluster. This represents an alternative implementation to achieving the sharing of data. Currently, applications need to be in the same cluster in order to share a database, and there is a limit on the number of nodes that can share a VAX cluster. The current limit is 16 nodes per VAX cluster. Thus, by using distributed DBMS, data can be shared outside the VAX cluster.
providing further flexibility in achieving integration of data.

3. **Cash Management System.**

The Cash Management system needs to access both, the daily dynamic database, and the historical database. The distributed DBMS takes care of the access of the required information, and neither the application nor the user needs to know where the data is actually located. Moreover, the Cash Management system could be located in either the TP or IP since there are no restrictions on location in order to share a database.

4. **Demand Deposit Account System.**

With the first approach, the daily dynamic database is located in the Transaction Processing cluster, and the historic dynamic database in the Information Processing component. This is illustrated in Figure 23. The DDA system would be able to access either of the dynamic databases transparently. However, the current distributed DBMS technology does not handle the updating of redundant databases (i.e., deferred copies), therefore the historic dynamic database would have to be updated daily with the postings that occurred in the daily database, and procedures need to be set up to handle the redundancy. When that technology becomes available, the historical DDA database can be automatically updated, - as frequently as needed, probably once a day, - with all the
Figure 23
Distributed databases architecture
updates that have been done to the daily dynamic database. Thus, applications will not need to propagate the updates to other databases. Some distributed DBMS users believe that this feature should be available very soon.

With the second approach, the consolidation of dynamic databases in IP, the maintenance of the historical dynamic database, as a result of the changes occurring in the daily dynamic database would be completely eliminated since both databases are consolidated. Moreover, all the operational functions required to maintain both databases is eliminated. Distributed DBMS vendors argue that there is performance transparency, this means that performance does not depend on location. Thus, placing the dynamic database in IP should not degrade FTS performance (i.e. postings). However, the duplication of the daily database in FTS for performance reasons would only be possible when the commercial distributed DBMSs support deferred copies. In this case, the propagation of updates will be taken care by the distributed DBMS. However, if data is duplicated in different systems, issues such as, concurrency control, and locking, need to be taken care of in the redundant databases. Hopefully, the technology to become available supporting deferred copies will take care of these issues.

In sum, distributed databases seems to be an alternative to be considered for future directions. It reduces costs of
maintenance and operation by at least the same amount as the previous alternative. Moreover, if applications, such as the Investigations System is included in this architecture, investigation queries will reflect the books of the bank in real time.

Distributed databases enable the sharing of data without imposing restrictions on the location of the data. This removes the technological constraint imposed by the maximum limit of nodes supported by a VAX cluster. The main advantage of this approach is to provide flexibility in the sharing and access of databases by systems located in different sites. The main gains that would be obtained from it are elimination of duplicate processing, shadow postings, tape hands-off among systems, duplicate maintenance, and duplicate information which implies a waste of storage. All these will have an effect on the ability to reduce operational costs besides providing a much better environment to manage risk control and handle growth and evolution.
Conclusions

This thesis evaluates the validity of the conceptual model as the means to meet the goals of the Financial Institution organization. Although the conceptual model is a good theoretical vehicle that compromises the three CIS principles, namely, integration, autonomy, and evolution, its implementation has shown to be weak in some aspects, especially in those concerning integration.

Integration has been partially accomplished by building a communications network infrastructure that allows to communicate across independent systems. However, it has failed to provide an infrastructure for the integration of data. As a result, independent databases have been created, some of which contain redundant data.

The redundancy of data brings several issues, consistency of data, propagation of updates, duplication of efforts, tape hands-off, duplication of processing, etc. All of these have an effect on the cost to maintaining these systems. Autonomy, on the other hand, has been accomplished by allowing these systems to be developed independently, most of them being hosted in their own hardware, both of these factors have resulted in their successful implementation. However, autonomy has been mediated by the setting of hardware and software standards which will facilitate integration. Finally, these
systems are better prepared to accommodate evolution, mainly because there has been some level of systems aggregation, and some of the technical constraints imposed by the previous architecture have been removed.

The main question that remains opened and which needs to be addressed is the integration of the data. By reviewing the Composite Information System methodology, the main technical obstacles identified are, the dispersion and redundancy of the databases across the FI systems, and the lack of use of DBMS technology. The main organizational obstacles are, the high degree of local autonomy, the desire for immediate results, and budget constraints. These obstacles have an effect on the cost of operations, maintenance, and future performance of these systems. Integration was identified as a critical factor to reduce these costs in the long run. Even if integration of data may not be perceived as an immediate need, it may either jeopardize the ability of the systems to grow, or their maintenance can become a burden that may constraint the need to meet new requirements. Moreover, in order for these systems to accommodate growth without incurring future high costs of development, maintenance of the systems, and lost of operations due to a non competitive service, these systems should accommodate to the integration goal.

Integration can only be truly accomplished by consolidating the data, either physically or logically. The use of DBMS
technology would take care of the following issues, concurrency control, security, and transparency in the access of data. Thus, DBMS will alleviate the applications programs from performing these tasks which will have an effect on ease of development, and ease of maintenance. Although it has been argued that the high performance required by these systems does not permit the use of DBMS technology, in the particular case of the dynamic database, the use of DBMS technology may be feasible. Consolidating the dynamic database can also provide advantages for risk control, since checks can be performed against the current "bank books".

Another approach to be explored is that of distributed databases. This technology allows the sharing of segregated databases located in different sites by different applications. This technology removes the technical constraint of having to belong to the same VAX cluster in order to share a database. Moreover, the burden imposed by concurrency and security controls, and access to the data would be handled by the database manager. Although this technology does not allow yet for automatic updating of redundant databases, this technology is expected to emerge soon. When this technology is available, databases can be spread out allowing for redundancy to obtain higher performance –if needed–. Moreover, this databases are expected to increase in performance, and are currently available for the hardware architecture used in the bank,
namely VAXes and the MVS operating system.

Until that technology becomes more mature, other roads can be explored. The split of the dynamic database into daily database, and historical database seems to be the most plausible alternative. This requires moving the daily processing DDA, the LOC, and the CM systems to the Transaction Processor (TP) cluster. Moreover, the movement of the daily processing of DDA has already been considered in the Strategic Systems Plan for Financial Institution Systems. By moving DDA to TP along with LOC and CM, sharing of the daily dynamic database would be possible, eliminating all the costs incurred by having to update in real time the otherwise redundant dynamic databases. In this case, the TP component would have all the current (i.e., daily data) whereas the IP component would have all the DDA historic data. This is also consistent with the conceptual model. By using DBMS technology to implement the shared database, a further step towards the full integration of data is accomplished.
References


Bibliography


5. Keen, Peter; Bronsema, Gloria; and Zuboff, Shoshanna. "Implementing common systems: One organization's experience." CISR WP #80, August 1981.

