1996-09

Reengineering DoD through enterprise-wide migration to open systems

Cameron, Robert A.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

REENGINEERING DOD THROUGH ENTERPRISE-WIDE MIGRATION TO OPEN SYSTEMS

by

Robert A. Cameron
Kenneth G. Carrick

September, 1996

Co-Advisors: James C. Emery
Barry Frew

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 1996	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. **TITLE AND SUBTITLE**
REENGINEERING DOD THROUGH ENTERPRISE-WIDE MIGRATION TO OPEN SYSTEMS

5. **FUNDING NUMBERS**

6. **AUTHOR(S)** Cameron, Robert A. and Carrick, Kenneth G.

7. **PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Naval Postgraduate School
Monterey CA 93943-5000

8. **PERFORMING ORGANIZATION REPORT NUMBER**

9. **SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

10. **SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. **SUPPLEMENTARY NOTES** The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. **DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for public release; distribution is unlimited.

12b. **DISTRIBUTION CODE**

13. **ABSTRACT** ***(maximum 200 words)***

The Department of Defense cannot afford to develop and deploy information systems that have no growth potential. Legacy systems must be replaced with flexible, highly interoperable systems that produce high residual values. With shrinking budgets, depreciation of existing hardware, and rising maintenance of legacy systems, organizations must deploy systems that are capable of evolving with changing business requirements.

The Department of Defense enterprise vision for information management (IM) emphasizes integration, interoperability, flexibility, and efficiency through the development of a common, multi-purpose, standards-based technical infrastructure. This vision requires a new paradigm for building information systems.

The new paradigm relies on open systems, which make it easier, less expensive, and faster to develop and change applications and to employ new technology features. This research examines open systems and provides a strategy for organizations to migrate to them. A case study of the Naval Postgraduate School illustrates the strategy. Provisionally, a prototype application models the desired characteristics of an open system.

14. SUBJECT TERMS Open Systems, Client/Server, Reengineering			15. NUMBER OF PAGES 191 16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited.

REENGINEERING DOD THROUGH ENTERPRISE-WIDE MIGRATION TO OPEN SYSTEMS

Robert A. Cameron
Lieutenant, United States Navy
B.S., United States Naval Academy, 1989

Kenneth G. Carrick
Captain, United States Army
B.S., United States Military Academy, 1986

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 1996

Authors: [redacted]

Robert A. Cameron Kenneth G. Carrick

Approved by: [redacted]

James C. Emery, Co-Advisor

[redacted]

Barry Frew, Co-Advisor

[redacted]

Reuben T. Harris, Chairman
Department of Systems Management

ABSTRACT

The Department of Defense cannot afford to develop and deploy information systems that have no growth potential. Legacy systems must be replaced with flexible, highly interoperable systems that produce high residual values. With shrinking budgets, depreciation of existing hardware, and rising maintenance of legacy systems, organizations must deploy systems that are capable of evolving with changing business requirements.

The Department of Defense enterprise vision for information management (IM) emphasizes integration, interoperability, flexibility, and efficiency through the development of a common, multi-purpose, standards-based technical infrastructure. This vision requires a new paradigm for building information systems.

The new paradigm relies on open systems, which make it easier, less expensive, and faster to develop and change applications and to employ new technology features. This research examines open systems and provides a strategy for organizations to migrate to them. A case study of the Naval Postgraduate School illustrates the strategy. Provisionally, a prototype application models the desired characteristics of an open system.

TABLE OF CONTENTS

I. INTRODUCTION

Today's organizations are beginning to realize that one of their critical challenges is the management of information systems (IS). Technology is moving so rapidly that IS is no longer just a component *in support of* strategic goals and missions; it is quickly *becoming* the strategic fulcrum of an organization's success.

The reality of today's business environment dictates that managers in all functional areas must do more with less resources. Nowhere is this more true than in IS.

This research investigates the trends with which IS management must deal. Specifically we address migration to open systems in a distributed environment. Our research provides common-sense strategies for dealing with the complex and turbulent realities IS managers face today. A case study of a mission-critical system at the Naval Postgraduate School illustrates the implementation of our proposed strategies.

A. BACKGROUND

The 50's, 60's, and 70's were dominated by host-based systems with the mainframe as the dominant fixture of corporate computing. With the advent of the personal computer (PC) in the early 80's, computing power began to shift from the centralized control of the IS department to the desktops of users. Local area networking (LAN) and wide area networking (WAN) have decentralized organizations' computing power even further.

Client/server architectures and distributed computing have served to complicate the management of information systems. At the outset, these technologies were touted as a low-cost alternative to the expensive mainframes of the past. However, organizations have begun to realize the hidden costs associated with their implementation. Recent

research by the Gartner Group, a leading information technology research consultancy, revealed that the costs of implementing client/server solutions may be two to three times that of a comparable mainframe solution [Ref. 1].

The Internet and web technology, the latest computing craze, promises the advantages of client/server computing without the difficulties in getting heterogeneous systems to communicate with each other.

For a number of reasons, such as lack of money or lack of technical expertise, many organizations have been unable to evolve with the technology. Instead, they have built stovepipe systems or relied upon their existing systems. When their business requirements changed, they either built another stand-alone system to deal with the change, or they modified their existing system with high-maintenance kludges. Before long, these systems cannot keep up with the changing requirements.

These systems have come to be known as *legacy* systems. A legacy system is one that "significantly resists modification and evolution to meet new and constantly changing business requirements [Ref. 2]." Not only are these systems expensive - maintaining and operating legacy systems consume 80-95% of IS budgets [Ref. 2] - they are inflexible and prone to break.

Because of the huge expense of maintaining and operating these legacy systems, too few resources are allocated for new development. Little development leads to even more legacy systems. So how do organizations get beyond this downward spiral?

This huge legacy cost must be leveraged. Instead of deploying systems that meet an organization's short-term needs, organizations must deploy systems that are capable of change and capable of evolving with the changing business requirements.

Paul Strassmann, former Department of Defense (DoD) Director of Defense Information, describes this system characteristic of growth potential as residual value. Information systems should be developed and deployed, he writes, in such a manner that when the system becomes obsolete, a portion of the system remains for inclusion into the replacement system. [Ref. 3]

Organizations simply cannot afford to develop and deploy systems that have no growth potential. Legacy systems must be replaced with flexible, highly interoperable systems that produce high residual values. With shrinking budgets, depreciation of existing hardware, and rising maintenance of legacy systems, how should organizations forge ahead with the necessary changes?

B. OBJECTIVES

The primary objective of this research is to provide a practical, common-sense strategy for managers struggling to deal with their monolithic legacy systems while attempting to cope with changing business requirements. In developing this strategy we examine the current trends in information technology with which managers must deal. A case study of the Naval Postgraduate School provides a real world example of how this strategy should be implemented. Provisionally, we provide a fully functional accounting system developed with Borland's Delphi Version 2.0, an object-oriented visual development tool. Complete source code and sample screen displays are included.

C. SCOPE, LIMITATIONS, AND ASSUMPTIONS

This research does not examine organizational issues associated with new system deployment; it is limited to the technological issues organizations must consider. That is not to say that the organizational issues are trivial. To the contrary, organizational issues

such as politics, "turf battles," training, user resistance, vision, and fiscal problems are the most difficult challenges to overcome - so much so that they could be the subject of an entirely different research effort. We assume that IS managers have the capability to overcome these organizational issues, either internally or with the help of outside consultants, as they implement our strategy.

We also do not advocate a methodology for conversion to a new system. The four most common approaches to system conversion - parallel conversion, direct conversion, phased conversion, and pilot conversion - must be analyzed to determine which is best for the system under consideration.

D. ORGANIZATION

Chapter II investigates the trend toward "open systems" and the many definitions of this term. Here we propose our own working definition and suggest an approach for IS managers moving to open systems.

Chapter III continues the research of trends in IS. Here we explore the concept of distributed computing, client/server architectures, and the latest trend toward using web-based technologies.

In Chapter IV we present our strategy for the development and deployment of information systems that are more open.

Chapter V is a case study of the Naval Postgraduate School. Here we document the development and deployment of a new accounting system using the strategies we propose.

In Chapter VI, we present our conclusions and recommendations. Finally, we suggest areas of further research.

II. OPEN SYSTEMS

One method of developing and deploying systems that have some degree of growth potential is to move to "open systems." This phrase, however, means different things to different people. It also has a temporal connotation; the meaning has evolved over time as technology has changed. It would follow that the term will continue to change with the technology.

The purpose of this chapter is to investigate the importance of open systems. We will examine the myriad definitions of open systems and suggest a practical methodology for managers to analyze the "open-ness" of their systems.

A. WHY OPEN SYSTEMS?

The major push toward open systems in the past two decades has been driven by users' desires to extend the useful life-cycle of their information systems. Specifically, open systems offer the following benefits:

- Provide an infrastructure for distributed applications
- Less reliance on proprietary products
- More competition leading to lower cost
- Decreased probability of schedule delay
- Better tested products (more users)
- Portable applications
- Increased interoperability through the use of industry standard links
- Faster technology insertion

B. EVOLVING DEFINITIONS

The IEEE defines an open system as a system that "implements sufficient open specifications for interfaces, services, and supporting formats to enable properly engineered applications software:

(a) to be ported with minimal changes across a wide range of systems,

(b) to interoperate with other applications on local and remote systems, and

(c) to interact with users in a style that facilitates user portability [Ref. 4]."

This definition has evolved with the technology and the changing business landscape. It provides a good reference point from which to begin examination of the numerous notions of open systems.

In the 1980s, open systems primarily referred to telecommunications. The International Standards Organization's (ISO) Open System Interconnection (OSI) reference model was the guiding force behind the open systems movement. While this model did provide a practical modular approach for defining the ways in which systems communicate with each other, it did not prescribe specific standards for doing so. Without these standards, an organization's use of the term "open systems" generally was limited to an *intention* to implement products that followed the OSI reference model. The problem was that vendors did not agree upon how to implement this open model.

In about 1990, open systems expanded to include the operating system (OS). The increasing popularity of the UNIX operating system was a major factor for this shift. Because it ran on many more platforms than other OSs, UNIX became the leading contender to usurp the mainframe's hold on enterprise computing. [Ref. 5]

In 1992, the term "open system" expanded again to include standard interfaces between applications and components of a system. Application Programming Interfaces (APIs) provide a formal method for applications to access a computational resource such as a file or a procedure from the operating system. Other interfaces define how different components within a system should interact with one another. This is important because the degree of openness can be measured by the detail and breadth of these interfaces.

The data component of a system has become a critical element of the expanding notion of open systems. Accessing the global expanse of enterprise data is a mission-critical task that requires standards of connectivity and format. Structured Query Language (SQL) has become the de facto standard to accomplish this. Different versions of the standard and proprietary implementations of it, however, have served to muddy the notion of open data.

With all of these definitions of open systems, one can see the difficulty that arises as organizations attempt to move toward such systems. Which definition do they choose? We suggest that there is no such thing as an "open" system.

Rather, "openness" should be viewed on a relative scale (See Figure 1). When analyzing existing and proposed systems, organizations should consider the system's degree of openness as compared to the alternatives. It would not be practical to say that an "open system" is required; what is "open?" What is more practical is to say that a *more* "open" system is required as compared to the existing system. Proper analysis of the alternatives can then lead to a more meaningful approach to problem.

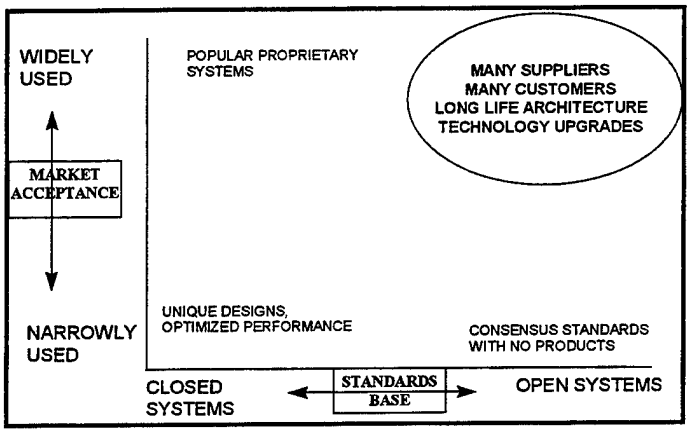


Figure 1. Open Systems as a Relative Scale

That analysis however, is an extremely complex undertaking. Like most complex problems, it is helpful to decompose them into more manageable and discrete pieces. In the next section we will analyze the characteristics of an open system and set the stage for our proposed strategy of moving toward more open systems.

C. CHARACTERISTICS OF AN OPEN SYSTEM

It would be extremely difficult to measure the openness of a system without looking at the system from a more detailed perspective. The following is an analysis of the characteristics of an open system.

1. Applications

Applications refer to the software that accomplishes the intended task of the system. The measures of openness would include the language in which the application was written, the availability of APIs, the ability to modify the code, and the availability of useful documentation.

The applications characteristic is perhaps the most critical part of an open system because this is where the work of the system is done. The potential openness of the other characteristics depends on the strength of the applications. It is possible to have great

skills sets, tools, data, and networks (any or all other characteristics), but without applications developed with a keen eye towards open standards to leverage these characteristics, the system will never provide residual value.

Application complexity and utility is directly proportional to the degree of openness desired. As systems become more open, applications are more able to take advantage of the benefits of distributed systems. This places application processing where it is most appropriate within an open environment. This is inherently more complex to manage and develop than applications running in a closed environment. By developing distributed applications built on open standards, an organization can fully utilize all available processing power, from the mainframe to the desktop PC.

This characteristic consumes a significant amount of analysis resources within organizations, as it is critical to have a modern, well-resourced application development program.

2. Hardware

This characteristic refers to the computers on which the clients and servers run. Naturally, for an even more granular analysis, separate analysis of the clients and servers may be practical.

At the closed end of the spectrum would be a completely proprietary system that runs perhaps on only one operating system. Because of their closed architectures and ability to only run proprietary operating systems, Apple computers have traditionally fallen toward the closed end of the openness spectrum. At the open end of the spectrum would be a system that could run many different operating systems and is capable of interchanging components. Because of the intentional open design of PCs by IBM, these

machines are able to support a variety of operating systems from DOS to OS/2 to certain strains of UNIX and LINUX.

3. Network

The network refers to the physical components that connect enterprise computing assets as well as the software that permits that connectivity. Again, if a system runs on several networks within an organization, it may be useful to analyze each one separately.

The degree of openness for this characteristic would be determined by the ability to support long-term vertical and horizontal growth and the ability for clients to access the necessary servers and required data. This implies the ability to support multiple platforms and hardware components as well as multiple protocols. We suggest that to even be considered, a network should support the vastly popular TCP/IP.

Network architectures must be capable of easily replacing components, simple plug-on multiple processors with higher clock speeds, or superscalar processing to allow for eventual growth of the target system. In addition, to be more open, the architecture should rely on industry-standard network connectivity cards and a fast I/O bus such as the 25-MHz SBus, which is supported by over 300 I/O cards from over 100 vendors.

4. Operating System

This characteristic refers to the operating systems running the servers and clients. The measure of openness would be the types of platforms that can run the OS and the applications that are able to run under it.

5. Data

Data is one of the most critical characteristics of an open system. The degree of openness can be measured by how easily the data can be accessed and manipulated by

different users and applications within and outside the organization. Proprietary systems that do not permit manipulation of data or conversion to useable formats would be at the closed end of the openness spectrum. Systems that conform to SQL standards or permit the conversion of data to open standards would be considered more open.

6. Skills Set

The organization's IS knowledge base should be considered in determining the degree of a system's openness. This characteristic refers to existing expertise as well as the ability to learn and adapt quickly to changing skill requirements. It essentially cuts across all other characteristics. That is, expertise in the other characteristics should be considered in determining how much the system will be able to grow.

For example, if the organization has an expert in SQL, and if the data conforms to SQL standards, then it is likely that the data components of the system would be capable of growth.

7. Tools

The development and system administration tools used certainly determine the degree of openness of a system. The measures of effectiveness would include the degree of ease and speed in which systems can be developed, and the ability to modify existing systems, diagnose problems, and integrate with other systems. While extremely powerful, industrial-strength development tools, such as CASE and I-CASE, exist, they are not applicable to the scope of our research, and will not be included in further discussion.

D. THE OPENNESS CONTINUUM MODEL

Each of these characteristics can be analyzed on a relative scale from completely closed to completely open. Figure 2 is a useful means to illustrate a system's degree of openness.

Characteristic	*Closed* ← → *Open*
Applications	X
Hardware	X
Network	X
Operating System	X
Data	X
Skills Set	X
Tools	X

Figure 2. Openness Continuum Model

This model can be a useful tool for organizations attempting to get a grasp on the complex issue of openness. At a glance, managers can easily see where they should concentrate their efforts and where existing characteristics can be leveraged. While it is desirable to move all characteristics to the most open alternative, this is not practical nor affordable for most organizations. This model will be used as a blueprint for analyzing the tradeoffs among characteristics. Chapters IV and V provide extensive discussion on its effective use.

III. DISTRIBUTED SYSTEMS

In Chapter II we discussed many of the benefits of open systems and offered a definition and a methodology for determining openness. We also stated that building applications that provide residual value ought to be an objective for organizations. Implicit in this is the notion that current legacy applications must be converted to a new environment when moving to open systems.

This chapter provides the technical background necessary to support our migration strategy described in Chapter IV. In addition, it discusses the technical issues with migrating to distributed systems and building distributed applications, such as client/server-based and e-mail- and world wide web-enabled client/server applications.

A. DISTRIBUTED SYSTEM CHARACTERISTICS

In analyzing distributed systems, we start by describing their components or attributes. The characteristics of these components are strongly coupled to the open systems characteristics of the environment as discussed in Chapter II; this will not be covered in-depth here. Distributed systems consist of the following attributes:

- Processing
- Connectivity
- Information storage
- Technical and organizational standards [Ref. 5]

1. Processing

Processing refers to the ability to manipulate data and perform computations in separate locations simultaneously. Distributed processing can be a service handled in the

background automatically by either the computer or the network operating system. Distributed processing can also be built into an application.

Choosing the processing location for a specific task in a distributed system requires examination of several factors such as:

- Type of task
- Client processing power
- Server processing power

By examining these three factors, developers can determine where best to distribute processing for a given task. For example, client processing power is generally a function of RAM, CPU, and swap file space. Depending on the relative strength of these parameters, a client can be classified as either "fat" or "thin."

A fat client performs a majority of required processing while a server supplies "raw" data. A fat client incorporates a GUI, business rules, and data integrity enforcement on the client machine, returning updates for storage on the server. File servers and many database server applications employ fat clients.

A thin client provides a GUI and transmits requests to "fat" servers. Fat servers provide high-level abstraction by exporting procedure calls and methods instead of raw data. This has an added benefit of reduced network traffic. Transaction, application, and object servers are examples of fat servers.

Load balancing technology is another alternative to dividing processing responsibility. An example is transaction processing (TP) monitors in database applications. All database requests are handled by a TP monitor that assigns transaction requests among a set of database servers equally, in order to maximize overall transaction

processing rate. TP monitors are not cheap and are used only in large distributed database application systems with many users. When they are required, they save time and reduce overall complexity in trying to manage transactions directly. [Ref. 6]

2. Connectivity

Clients require access to servers in order to process data. Likewise, servers need a path to provide data to the client. This simple relationship is the backbone of distributed applications and necessitates ubiquitous connectivity via a network.

The network is an integral part of the infrastructure and is often a top concern for IS. This is obvious, for without connectivity, real-time distributed systems are not possible. This management focus lies in network robustness and sound design. A poorly constructed network can suffocate IS financial resources just to maintain connectivity. Furthermore, a poorly designed network can severely impact performance in distributed applications. Proper design methods attempt to eliminate bottlenecks.

In distributed applications, the network is merely an extension of the client PC internal data bus. Therefore, loss of the network is the same as losing the client PC. Because of the dependence of distributed applications on the network, investment into the network infrastructure is money well spent.

3. Information Storage

Information storage refers to where data is located and how it is structured. Data may be stored in a large secondary storage pool associated with a mainframe, it may be stored on a PC server, or it may be scattered across an organization on desktop hard-disks. Data may be duplicated, out-of-date, erroneous, and inaccessible in many cases.

Distributed systems can bring order to data chaos and provide universal access to data - a precept of open systems.

SQL is the predominate language for access to data stored in relational model database servers. It is an ISO standard and well supported by DBMS vendors. There are several alternative data models, such as hierarchical and object-oriented. These alternative models either are legacy, or meet specific requirements for niche markets. Relational model database servers are the industry standard, with SQL providing an open avenue to data access. Two popular database architectures are distributed and federated databases.

a. Distributed database architecture

A distributed database system tracks data location on the network, and routes requests to the correct database node, making location transparent to the client. Several layers of middleware provide this transparency. The top middleware layer, located on the client side, provides an alias for use by client software. The bottom middleware layer, on the server side, the DBMS translates the logical update request into a physical data update. Figure 3 illustrates how this technology works. This alternative is not perfect, however. It has the following problems:

- The mechanisms that distribute data across servers are vendor-specific, locking an organization into a single vendor solution.
- It poorly encapsulates data and services. For example a local table cannot be changed, or restructured, if it is in use by another site.
- It requires too many low-level (SQL) message exchanges, resulting in excess network traffic.
- Due to the nature of transactions, partial updates are not possible. Therefore if a server becomes unavailable, all transaction processing that includes data on the unavailable server will fail, making administration difficult. [Ref. 6]

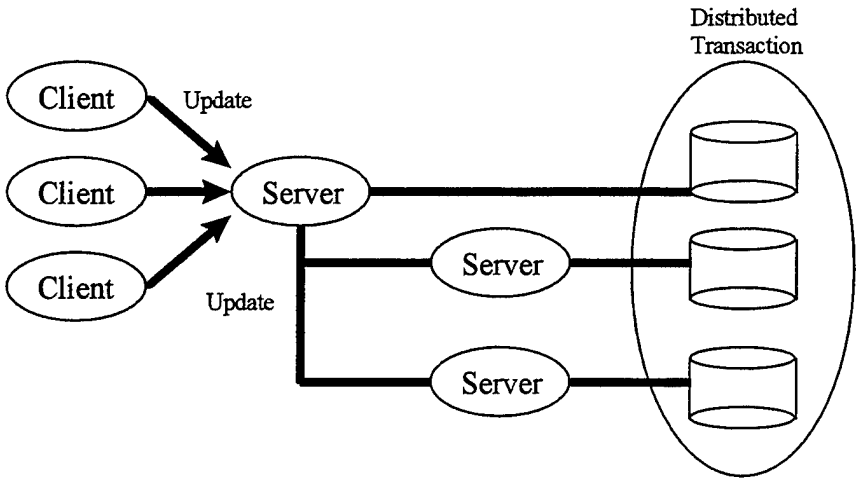


Figure 3. Distributed Database Model from [Ref. 6]

b. Federated databases

Federated databases are described as "multivendor, heterogeneous, database networks" [Ref. 6]. A federated database provides single-point access to all database servers within an enterprise through the use of middleware.

Single-vendor federated databases are the easiest to implement for the obvious reason that everything is common, e.g., SQL syntax, APIs, drivers, and stacks. Multivendor federated databases are much more difficult to implement and maintain due to proprietary interfaces and implementations. Despite these implementation challenges posed by federated databases, they provide many benefits:

- Legacy database servers do not require immediate replacement if they can adapt to a heterogeneous environment.
- They enable organizations to place database servers where the data originates, speeding access while promoting ownership.
- They reduce risk of downtime by storing data on multiple servers, providing redundancy unmatched in a single server configuration.

4. Technical and Organizational Standards

Technical and organizational standards refer to standards adopted by an organization in support of an open system environment. Examples of technical standards are network protocols, operating systems, and hardware capabilities. Examples of organizational and administrative standards are security, end-user computing guidelines, and content management.

Standards are the foundation of open systems. When to adopt a standard is an aspect of risk management. Any given standard can be evaluated on several levels. Is the standard supported by an international body? How much does market presence play in deciding to adopt a de facto standard? Is there a need to adopt cutting-edge standards or should a more conservative position be taken? These questions are not part of this research. They are merely presented to indicate that standards adoption is a strategic decision and should be treated as such.

B. CLASSIFYING DISTRIBUTED SYSTEMS

The following system configurations have appeared over time, with a hierarchy of processors the first to appear and cooperative processing the latest trend. All of these systems are in existence today, which is why we discuss them all, providing background for our open systems strategy in Chapter IV. They can be used to classify distributed systems.

- A hierarchy of processors
- Decentralized stand-alone systems
- LAN-based systems

- LAN systems that communicate with mainframe-based systems
- Cooperative systems [Ref. 5]

1. A Hierarchy of Processors

This system has a large controlling computer, such as a mainframe, at the top of the hierarchy, followed by minicomputers at the next level with dumb terminals or microcomputers at the bottom. The key component is the mainframe because it controls all processing. Also, data can be distributed throughout the system on departmental mini-computers or it can be centrally stored on the mainframe. [Ref. 5]

2. Decentralized Stand-Alone Systems

This system consists of a mini-computer with terminals connected to it. They are small in scale and usually reside within a department. These systems generally do not connect with other departmental systems but may transfer information "up" to the enterprise mainframe. [Ref. 5]

3. LAN-Based Systems

This system consists of linking microcomputers. This configuration allows microcomputers to share resources such as printers, scanners, CD-ROM jukeboxes, etc., as well as to connect to other LANs via bridges, gateways, and routers. All of the computers are equal with no single computer controlling any of the others. [Ref. 5]

4. LAN Systems That Communicate With Mainframe-Based Systems

This is a combination of LAN-based systems and a hierarchy of processors. This is a compromise solution, as it does not fundamentally change how an organization operates. It has the advantage of conserving past IT investments while potentially allowing information to be shared throughout the organization.

This type of system looks good on paper because it is networked. In reality, organizational boundaries can still exist and data may not be shared because the same physical and logical application structure is in use. [Ref. 5]

5. Cooperative Systems

In this type of system all computers are networked via LANs and WANs. Resources are shared and all computers can be peers; however, large servers and/or mainframes can be the hub of a system, providing centralized database services due to their massive I/O and data storage capacity.

Cooperative systems allow applications to be divided among computers. This permits application designers to maximize (or conserve) computing resources such as processing power, permanent storage, and network bandwidth, by placing workload where it is best handled. For example, if an organization has a powerful client-side machine, then the bulk of CPU operations may be placed there, creating a "fat client." Another reason to divide applications is to balance network loading by placing application modules physically on a server in the application's functional area. Once the application module processes a request, it forwards data via the network backbone to a central server for storage and reporting. This effectively employs network bandwidth by limiting traffic that must traverse the network backbone, keeping all other network traffic local to the functional area.

A cooperative system is the most open category because it permits an organization to maintain existing hardware investments while more effectively utilizing computing resources. [Ref. 5]

C. BUSINESS APPLICATION COMPONENTS

In the previous section, we discussed the characteristics of distributed systems and described five different types of distributed systems. In this section we describe the general components of a business application. We do this to set the stage for our discussion of client/server architectures in the next section. Berson [Ref. 7] identifies four different components in a typical application:

- Presentation processing logic
- Business processing logic
- Data processing logic
- Database server processing logic

1. Presentation Processing Logic

This component is the front-end of a client application with which users interact to manipulate data. It consists of menus and a GUI, if in a windowing operating system. It also manages local services required of the client operating system such as file I/O.

2. Business Processing Logic

This component contains rules that are enforced as a user interacts and manipulates data. This logic is usually written in the native language of the application. These rules can apply prior to viewing, during manipulation, or after viewing. Examples are calculating totals and setting flags based on data values prior to viewing, preventing illegal manipulations during editing, and validating data entries/changes after manipulation.

3. Data Processing Logic

This component retrieves data for presentation and stores updated data after manipulation. The language standard is SQL. However, any data entries or changes made by a user are translated to SQL statements for processing by an application.

4. Database Server Processing Logic

Database server processing logic serves and stores data by processing SQL statements. This component function is usually performed by a DBMS, that serves data for presentation to a user and applies edits made by a user. (See Figure 4)

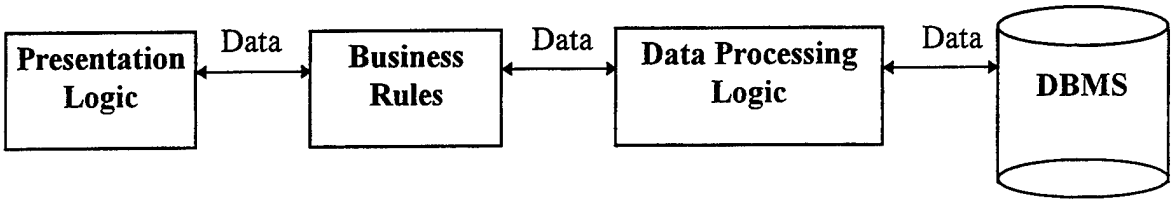


Figure 4. Diagram of a Typical Business Application Data Flow

In Figure 4, data is passed through several components or layers. Each layer processes data differently. This separation of processing is important to long term maintenance.

D. DISTRIBUTED APPLICATIONS

In the previous sections of this chapter, we discussed the categories of distributed systems and their characteristics. We also described a model for the components in a business application. In this section we concentrate on the popular client/server architectures used in building distributed business applications.

A distributed system can be described as a "tiered" architecture. The concept of tiers is used to differentiate where the five components of a business application are located.

1. Single-Tier

In a single-tier system, all components of a business application exist on a single machine. Because of this, application development is greatly simplified. The data can be stored locally so there may not be any network overhead, or it can be located on a logical network drive. Because there is no separation between the components, maintenance and modification will be difficult at best. An example is updating business rules as they change. It would be difficult to isolate the business rules code from the interface and database code. Worse, every application that is affected by the rule change must be updated separately. Finally, single-tier applications do not promote data sharing and should be avoided.

2. Two-Tier

In two-tier applications, there is a clear logical separation of the client and the server, be they located on a single machine or physically located on separate machines. In addition, depending on the configuration, many middleware layers may exist to move data between these two tiers. Examples of middleware are database vendor-specific software on the client and libraries to connect the vendor-specific software to an application, such as Microsoft's ODBC or Borland's IDAPI.

The server is usually one of two types: a file server or a database server. With a file server, the client passes requests for files over the network to the file server. File servers have been used to provide shared access to a desktop database , such as Paradox. Applications using this architecture were made possible by the PC revolution and were popular in the late 80's early '90s due to their ease of construction over mainframe

applications. They often provided quick solutions to departmental-level business problems, providing an excellent alternative to relying on a central mainframe.

Despite their appeal, file server-based applications have serious drawbacks. They tend to have large network bandwidth needs because of excessive message exchanges required to locate the requested data. Also, local desktop database applications, typical of file-server-based applications, lack robust backup functionality, transaction processing, and full featured SQL. Both of these drawbacks severely inhibit a file server-based business solution in scaling to the enterprise level.

Application design typical of file server-based applications tends to couple business rules with the user interface code. As with single-tier architectures, such coupling complicates design management and code maintenance, clouding the separation between user interface (UI) and business logic.

Because business logic is closely coupled with UI code, changes to the user interface often lead to complete redevelopment, necessitating rewrite of business rules as well. In addition, since the business rules are part of the UI, they must also be rewritten into every new application. Finally, business rules can change rapidly, demanding significant resource allocation to ensure all application versions are updated promptly and redistributed.

Remote database servers are an alternative to file servers. They permit applications to be separated into two distinct tiers in which the UI forms one tier and data and business rules are stored together in the second tier. In this architecture changes in the UI would not necessarily require modification of the business rules. As other applications require access to data, business rules are automatically applied.

The two-tier client/server architecture has been used to build mission-critical applications with success, but it is a complex undertaking. It requires expert knowledge of SQL, and DBMS-specific knowledge for programming business rules. Finally, business rules using SQL are difficult to write and debug. "These are not the tools for creating and maintaining good code. Doing it this way is SQL-abuse; SQL is a query language and was never designed to do this kind of procedural computation [Ref. 8]."

Despite its advantages over file server-based applications, the two-tiered approach has several limitations. First, database servers are optimized to provide data as quickly as possible. They are not optimized for enforcing business rules, so placing them in the database inhibits server performance. Because of the combined processing requirements of data and business rules, applications are difficult to scale upwards.

3. Three-Tier

The latest trend in client/server is the three-tier model, as diagrammed in Figure 5. This model consists of a server, a client, and a middle-tier where business rules are stored. Here, the database server performs its optimal role of serving data, the client manages the UI, and an application server applies business rules. These components can be physically or logically separated.

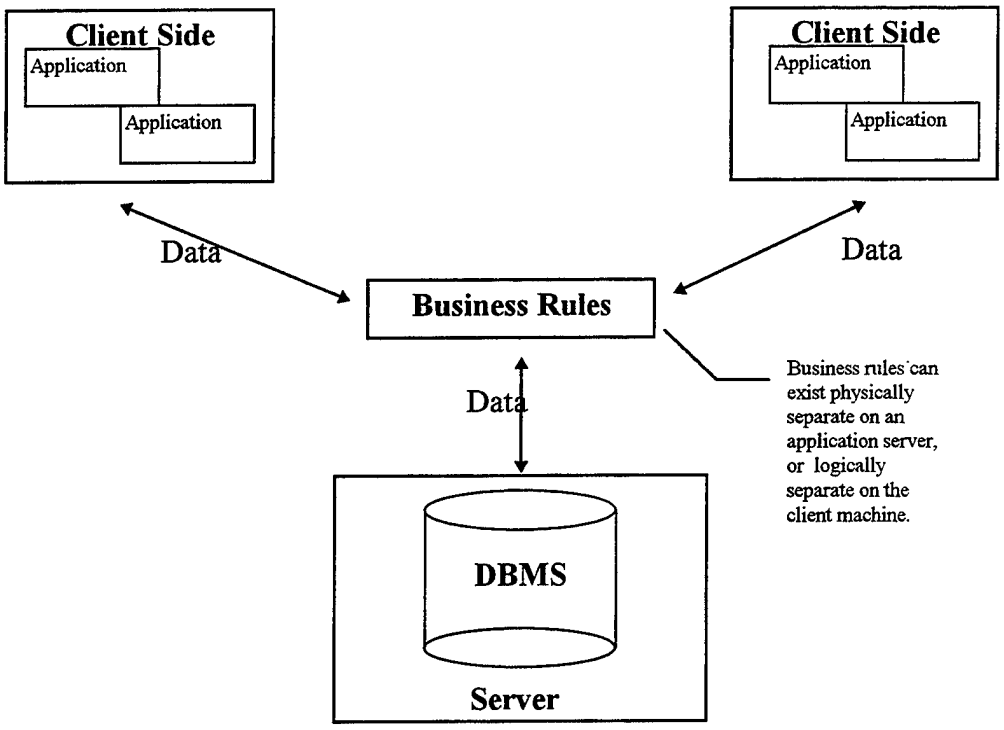


Figure 5. Three-Tier Business Application

a. Physical Separation

The physical three-tier architecture places another machine as an application server in the business rule layer. As new applications are needed, the middle tier provides a consistent interface between the UI and the database for each application, ensuring consistent enforcement of business rules irrespective of the client application seeking access to the database.

In very large environments, such as major DoD installations, the use of additional hardware in the business rule layer provides the added benefit of load distribution. Multiple middle tier servers can be distributed across the organization to provide scalability.

Physical three-tier models are not appropriate for all situations. The use of a third physical layer adds additional expense and complexity in the form of extra

hardware and software with their attendant implementation and maintenance costs. Project management is further complicated by the potential use of three separate programming languages for each of the three tiers.

Recent technologies promise easier access to and implementation of business rules in a physically separated middle tier. Microsoft's Remote OLE Automation holds promise for the future, especially with the dominance of clients with Intel processors and Microsoft operating systems. Other technologies such as CORBA, ISAPI, and JDBC also hold promise, but they have yet to mature.

Whatever standard dominates, client-oriented tools add support for developing the middle tier natively. This greatly reduces the complexity of the physical three-tier architecture by eliminating the need for a third development environment specific to the middle tier. Until an alternative becomes available, the logical three-tier is most appropriate in most cases.

b. Logical Three-Tier

The primary difference between logical and physical three-tier architectures is that with logical three tiers, business rules are implemented on the client machine. This is also a significant change from the two-tier architecture where business rules are implemented in the DBMS server. In the logical three-tier architecture, the middle tier is physically located on the client hardware but logically separate from the UI portion of the application. This has several benefits, such as:

- Developers are able to create business rules in the middle tier using a language and tools with which they are already skilled.
- Often, all that is required to re-implement new business rules is a recompilation of the existing applications with the modified middle tier.

- Development effort is reduced when creating new applications that use the same database. Business rules can be shared objects.
- Modular design of the middle tier provides re-usable software, thus increasing residual value.

c. Physical or Logical?

Currently, the physical three-tier is only a possibility in large organizations with mature IS support. This will change when a clear distributed computing standard emerges and vendors bring products to market that give programmers more flexibility to develop in their native environment. Until then, the cost and complexity resulting from implementing an application server is prohibitive in most cases.

For most organizations, the logical three-tier model is the quickest opportunity to achieve scalable client/server solutions. Also, both physical and logical three-tier architectures are best implemented in an object-oriented environment due to the inherent modularity, abstraction, and inheritance characteristics of object-oriented languages. This is most important in the logical three-tier architecture where the UI and business rules share the same platform. Without careful attention to achieve logical separation, business rules can easily end up in the UI tier.

d. N-Tier

The three-tier architecture can be extended to *n*-tiers with multiple middle-tier layers providing connections to various types of services, integrating and coupling them to the client, and to each other. An *n*-tiered system can also be created by partitioning the application logic among various hosts. Encapsulation of distributed functionality in such a manner provides significant advantages.

- Network bottlenecks are minimized because the application layer does not transmit extra data to the client, but only what is needed to handle a task.

- When business logic changes are required, only the server has to be updated. In two-tier architectures, each client application must be modified when logic changes. With the *n*-tier architecture, the client is modified only when functions are discontinued or the function's parameters change. The client is insulated from database and network operations. It can access data easily and quickly without having to know data location or how many servers are on the system.

- An organization has database independence because the data layer is written in standard SQL and is platform independent. In addition, the enterprise is not tied to vendor-specific stored procedures for business rule implementations.

- An effective communications pipeline is created between the application layer and the client.

- An application layer can be written in standard third- or fourth-generation languages, with which the organization's in-house programmers are experienced. [Ref. 9]

Alternative solutions have arisen as a result of the difficulty in developing application servers as a middle tier. One such alternative, an intranet, provides great promise because it supports open systems and truly abstracts out the UI from the other portions of the application. Intranets are not a replacement for client/server; they are an extension to client/server.

E. EXTENDING CLIENT/SERVER

The holy grail of client/server is the "universal client" meaning that a client can access an application from any platform. This goal has been almost impossible to achieve with the traditional client/server architecture because of operating system and microprocessor dependencies. One solution is to take advantage of existing technology used on the Internet. Figure 6 represents a client/server architecture that includes another tier, the web server, with the client UI managed via a web browser.

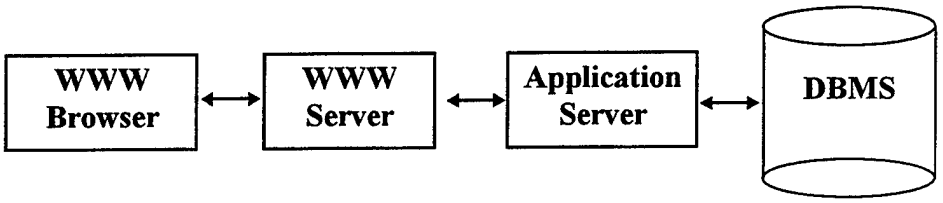


Figure 6. Web Based Client/server Application

In this scenario, any client operating system or microprocessor that supports TCP/IP and has an Internet WWW browser can interact with this application. The WWW browser initiates a request for data. The WWW server translates the request and passes it to the application server for processing. The application server validates the request and retrieves the data from the DBMS. The application server then applies its business rules and passes it to the WWW server. The WWW server takes the data and packages it in the form of a web page that then can be viewed by the browser.

Notice that this is a four-tier architecture with several advantages over the logical three-tier model. The application server provides the interface between the WWW server and the DBMS. This tier can be constructed using current RAD tools such as Delphi, Visual Basic, PowerBuilder, or Oracle 2000. The benefits of this are enormous. The business rules are implemented as a separate tier and they are separated from the UI. This promising technology, known as an intranet, deserves further examination.

1. Defining an Intranet

An intranet is a concept defined formally in January 1995 by Steven L. Telleen, Ph.D., Director, IntraNet Solutions at the Amdahl Corporation, in a white paper entitled *IntraNet Methodology*™. Simply put, an intranet is an enterprise communications architecture that relies on *Inter*net technologies. The "intra" part of intranet means that

information is shared only within the confines of the enterprise, although access to the Internet can be, and most times is, implemented.

Not too long ago, most networked devices were 3278 and 3279 character-mode terminals that had coaxial cable connections to large mainframe-based networks. Most of these devices have since been replaced by personal computers and local area networks. The intranet portends a similar revolution in information processing. There are several distinguishing features of an intranet.

- It uses TCP/IP for both wide-area and local-area transport of information.
- It uses HTML, SMTP, and other open Internet-based standards as the means of moving information from clients to servers.
- It is completely owned by the organization and not accessible from the Internet-at-large by the general public. This is usually enforced by a firewall.
- It is managed by IS with similar attitudes and procedures as they currently manage their legacy mainframe-based networks; only the tools are different. [Ref. 10]

A detailed review of these features will illustrate the trends leading up to the widespread use of intranets.

a. TCP/IP as the Emerging Protocol of Choice

Today, most enterprise networks are a mixture of many protocols: IPX, TCP/IP, SNA, Banyan Vines, and AppleTalk, to name a few. Many organizations have begun careful evaluation to standardize on one protocol; typically TCP/IP is chosen. The reasons for its popularity are many.

- It can handle both LAN and WAN traffic well.
- It is supported by the majority of computing platforms, ranging from Macintoshes to Windows NT to the largest mainframes.

- It has a robust set of management tools and an active development community to enhance them.
- It is the protocol used by the vastly popular Internet. [Ref. 10]

While the merits of standardizing on TCP/IP are worthwhile, it is not without a downside. TCP/IP is hampered by large memory requirements - up to 150 KB - especially on DOS-based machines. Many organizations, like DoD, still have a preponderance of DOS-based machines and the migration to more advanced systems could mean a huge hardware investment. Fortunately, hardware costs for Windows-based machines is dropping at a rapid pace. These systems operating under Windows95 and NT offer tighter integration and better support of TCP/IP.

b. Internet Open Standards

HTML is the language used to define the structure of hypertext documents on the intranet provided by servers. The display of HTML documents is the responsibility of clients or browsers. While HTML is an open standard, it is not wholly "standard." Different vendors use different versions of the standard, and some vendors inject their own feature sets into their products. Most likely, the standard will continue to evolve. Most browsers do, however, provide a core set of standard features according to the HTML 2.0 standard.

Even the core standards provide for the transport of data in a wide variety of media, to include text, recorded speech, and graphics. Most browsers have the capability to transport formatted tables, video clips, and animation.

The flexibility with which data can be presented has led to the explosive growth of the Internet in the last 18 to 24 months. The "web craze" has now hit

organizations seeking ways to minimize costs for their communications architectures. Web sites can range from the most mundane of lists to the most sophisticated of multimedia shows and from the most personal to the most corporate, depending on the content, author, and effort.

The openness of web-based technology allows for information to be cross-linked from web server to web server, whether they be located around the world, just down the street, or just down the hall. It is this ability to link, when designed correctly, that enables the power of the web as a distributed corporate information source. [Ref. 10]

The web is only one of many capabilities of an intranet, however. Along with this technology is support for other standards such as FTP servers, Gopher servers, SMTP and others originally developed for UNIX computers, but have become commonplace among PC-based LANs and WANs.

Because of the use of these open and pervasive standards, intranets are quickly becoming the architecture of choice in corporate America. In fact, the use of web-based technologies for intranets is quickly catching, and will soon surpass, Internet usage (See Figure 7). According to Zona Research, a market-research firm located in Redwood City, California, revenues from the sale of web servers for an intranet will be four times those for the Internet [Ref. 11].

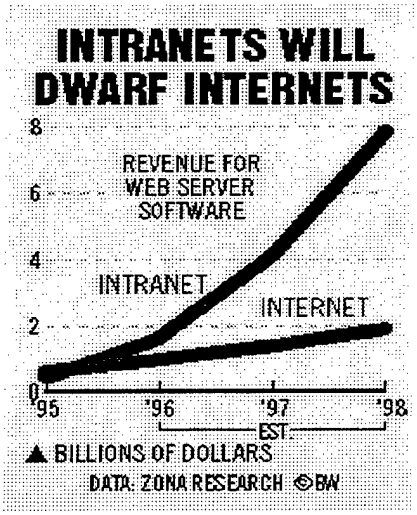


Figure 7. Intranet Trends

c. Internal Enterprise Access Only

Intranets, if designed properly, offer the benefits of the Internet without the risks of invasion. Pure intranets are those that run solely on a LAN and provide no access to the Internet. However, organizations have begun to achieve secure intranets behind firewalls.

A firewall is a system or group of systems that enforces an access control policy between two networks. The actual means by which this is accomplished varies widely, but, in principle, the firewall can be thought of as a pair of mechanisms: one that exists to block traffic, and the other that exists to permit traffic. In an intranet, a firewall is typically implemented to permit access from within the intranet to the outside Internet, while limiting traffic from the outside Internet into the organization's intranet.

However, as the popularity of the web continues to increase, organizations have begun to allow public access to certain portions of their intranets. These "extended" intranets offer the best of both worlds: the ability to allow the public to see that which is desired while limiting access to that which is not. This type of intranet, by its nature,

poses an increased security threat. Organizations should perform careful risk analysis before implementing an extended intranet. Analysis of vulnerabilities is beyond the scope of this research. Suffice it to say that current technology will support such implementation, but organizations must thoroughly analyze the risks and benefits of such an undertaking.

2. Intranet Management

Management of an intranet is no different than that of a typical legacy mainframe-based network. The purpose and procedures are the same; only the tools required to manage the two are different.

Although an intranet is based on Internet technology, this does not mean that an enterprise intranet must follow the same, anything-goes policies that have grown with the Internet.

3. Analysis of Intranet's Merits

The web is enabled by a widely adopted set of standards for creating and communicating information across networks. These standards provide six powerful benefits:

a. Platform Independence

Information can be created, served, viewed, and moved without modification across different hardware, operating systems, and software.

b. Information Transparency

Information can be retrieved from another computer anywhere on the web without the user needing to know where the information is physically located, what kind

of machine and operating system is serving the information, or what network commands are involved.

c. Ease of Use

Web page development complexity is increasingly more transparent to users with the emergence of powerful new tools such as Microsoft's Frontpage and Internet Assistant for Word and countless HTML editors. These tools greatly reduce the learning curve.

d. Universal Client

Web pages can provide users with an easy-to-use, common interface across multiple applications and data, including legacy applications. This translates into significant savings in user training costs. Users need only to be trained on the enterprise browser of choice.

e. Cost Efficiency

Information distribution costs can be reduced compared to distributing the same information on paper or in person. On the external web this means reaching potential customers directly for the cost of publishing the information once. On the internal web it means reducing paper, printing, copying, mailing, and faxing costs.

f. Time Efficiency

Users can control their own information flow, reducing the time consuming activities of sorting, evaluating, and filing just-in-case information that currently floods their in-baskets.

4. Analysis of Intranet Shortfalls

Intranets have several limitations. From an application development perspective, current generation WWW development tools have not matured enough to support mission-critical applications using the web browser client as the UI. This is a short-term issue, however, as vendors are rushing to provide out-of-the-box solutions to intranet construction. Gradient Technologies Inc. in Marlboro, Mass., and WayFarer Communications in Mountain View, Calif., are two startups working on that challenge. The basic problem, says WayFarer CEO Edward Colby, is that Web servers were not designed for high-speed transactions, things like getting a credit authorization. WayFarer's QuickServer uses its own messaging protocol to speed up transactions and juggle high-volume database requests.

Another issue with web-based development is maintaining state. The web is a "stateless" environment, meaning that developers have no programmatic control over different users and what they will do next. End-users may branch off to another page unconnected with an application and may or may not return. If application program logic requires closure from end-users, it may not happen. One solution is use of "cookies" embedded in web pages. Cookies are transmitted to the client browser from the web server with state information. If an end-user returns to the web page, state can be retrieved from the cookie by the web server. A more robust solution is provided by HREF Tools' WebHub™ product that provides an object framework for Delphi which manages state from the server application, in large part providing necessary programmatic control.

Despite the seemingly easy facilities needed to create web content, mission- critical applications still require substantive tools. Web application development is no different than traditional application development in many ways. There is still a need for tools like project management to enable many people to work on projects, for debugging and testing tools, and for production-level database design and modeling tools for back end development.

IV. STRATEGY FOR OPEN SYSTEMS MIGRATION

Chapter II of this research provided an overview of the evolving meaning of open systems. We also discussed the benefits of open systems and suggested a useful model with which managers could analyze the openness of their systems.

In Chapter III we extended the discussion to distributed systems, an architecture made possible by open systems standards.

The previous discussion essentially prescribes *what* organizations need to do in order to develop and deploy information systems that are not legacy the day they are put into action. This chapter aims to prescribe *how* organizations should go about the daunting task of moving their IS infrastructure into a viable and contemporary asset, capable of growth and evolution in a changing business environment. Our premise is that an open systems infrastructure, capable of supporting modern distributed computing, cannot be built overnight: it is too costly and time consuming. Rather, it must be evolved over a realistic time frame. The remainder of this chapter outlines our strategy for doing so.

A. STEP ONE: BASELINE ANALYSIS

Using the Openness Continuum Model discussed in Chapter II, organizations should examine their major information systems. This analysis should not consume excessive time or resources. Its intended purpose is only to give an organization a general indication of the current condition of their infrastructure. Results of this cursory analysis may include the following:

- Alarming or encouraging trends in the current state of a particular characteristic (Hardware, Operating System, Network, etc.)

- Identification of likely candidates for migration, thus allowing for management to keep a closer watch on performance

- Conversely, identification of systems that fall toward the open end of the spectrum and thus may not require much attention for some time to come

B. STEP TWO: IDENTIFY SYSTEMS THAT REQUIRE MIGRATION

Any migration process will be extremely costly to an organization. It is simply not realistic to expect that organizations will be able to reengineer their entire information infrastructure at once. They may be able to continue using many existing systems. This step of the strategy is to identify those systems that must be migrated. Some of the indicators that a system should be migrated to a more open one are:

- Results from cursory analysis in Step One indicate the need to migrate.

- A system simply becomes too costly to maintain.

- A system no longer meets the requirements and modifications are too costly or not cost effective.

- A system is incapable of any additional kludges (longevity of many legacy systems is extended through use of patchwork fixes.

- Although all systems still meet the requirements, money is available to reengineer one or more systems that may require migration in the near future.

- A mandate from a higher authority requires migration.

C. STEP THREE: SYSTEM REQUIREMENTS ANALYSIS

Once an organization identifies that a system requires migration, it must conduct a detailed analysis of what the system should do. This analysis should result in the specifications for basic information, functions, performance, behavior, and interfaces. This step is no different than any other system development effort. This is a critical step because in the next step, openness analysis, competing alternatives of each characteristic

will be analyzed for openness. Only those alternatives that meet all system requirements should be considered as candidates for the new system.

D. STEP FOUR: OPENNESS ANALYSIS

The requirements analysis in Step 3 should identify the core functionality of the system. From this foundation, the drive to a more open system can proceed.

1. Identify Competing Alternatives

For each characteristic, attempt to place the existing system on a relative scale among the available alternatives (See Figure 8). For example, for the operating system characteristic, the existing system may run on a Windows 3.x. That would be considered more open than, say, DOS 5.0, but less open than Windows 95, Windows NT, and UNIX.

Characteristic	Closed ⟷ Open
Applications	*Existing System* — *Alt A* — *Alt B*
Hardware	*Existing System* — *Alt A* — *Alt B*
Network	*Existing System* — *Alt A* — *Alt B* — *Alt C*
Operating System	*Existing System* — *Alt A* — *Alt B* — *Alt C*
Data	*Existing System*
Skills Set	*Existing System* — *Alt A* — *Alt B*
Tools	*Existing System* — *Alt A* — *Alt B*

Figure 8. Example Use of the Openness Continuum Model

2. Pare Down the Alternatives

Rule out any characteristic alternative that does not meet the minimum system requirements identified in Step Three and the minimum openness desired. For example,

the DOS operating system might be eliminated because of its lack of native TCP/IP support.

This process should also remove from further consideration those alternatives that are not possible. This could include those that are not affordable or those that are not permitted by higher authority mandate.

3. Define Competing "Packages" of Alternatives

Because of the interdependencies among characteristics in the Openness Continuum Model, it is most likely not possible to create a system consisting of the "winner" (the most open alternative) from each characteristic. Some characteristics are dependent upon others. For example, an application development environment based on a 32-bit development tool would naturally require a 32-bit operating system.

In this step it is important to determine the characteristics with which the success of the system will hinge. The alternatives for these critical characteristics will form the basis for competing "packages" of alternatives that will work well with each other and will result in a new system that meets the desired degree of openness. These packages are comprised of one or more competing alternative from each characteristic from the Open Continuum Model, each a viable solution to the minimum system requirements and the desired degree of openness.

In defining these packages, an organization should consider a "mixed portfolio" strategy of low-risk alternatives mixed with high-risk alternatives. For example, the organization may counter an emerging, highly proprietary database management system from an upstart vendor with a relatively low risk selection in the network characteristic

such as one that uses the existing network hardware and industry leader Novell's Netware 3.x.

The result of this step should be a number of competing packages that (a) meet the minimum system requirements, and (b) identify an alternative for each characteristic of the Openness Continuum Model. A simplistic example package may look like Table 1.

Characteristic	Alternative Selected
Applications	Custom developed
Hardware	Windows/Intel Machines
Network	Peer to peer
Operating System	Windows NT
Data	Oracle
Skills Set	In-house, existing workforce
Tools	Delphi

Table 1. Sample Package

From this detailed analysis of each of the characteristics of the Openness Continuum Model, organizations will be postured to make prudent decisions on the direction for the migration. This is essential to the follow-on steps of the migration strategy.

E. STEP FIVE: COST/BENEFIT ANALYSIS AND SELECTION OF TARGET SYSTEM

Each of these packages should be put through a rigorous cost/benefit analysis. This will be the most arduous step because of the interdependencies among characteristics and the difficulty in quantifying intangibles such as the ability to better support future interoperability.

Entire books have been written about the subject of cost/benefit analysis. Examination of how to best accomplish this complex task is beyond the scope of this

research. What we offer here are some considerations in identifying the costs and benefits of the competing packages.

This analysis should include rigorous research into all costs associated with the alternatives. This research should include cost of purchase, maintenance contracts, training, license fees, and any costs that may arise from the conversion process. For example, if a parallel approach to conversion is required, the incremental costs of running two systems should be considered, especially if another alternative would support a direct conversion approach. Intangible costs such as user resistance need only to be identified or, if possible, quantified in some manner.

It is important to consider how well an alternative's characteristics fit into the long-range plans of the organization. The selection of a particular alternative in a package may result in long-term costs or benefits. For example, if a less than optimal data alternative is part of a package, this may make it more difficult to migrate other existing systems in the future. An organization may be able to quantify these costs, but most likely the costs will fall into the intangible category.

In addition, it is critical to analyze the effects of choosing one alternative on the other characteristics. For example, how does the selection of a particular operating system impact the existing application base? Does the organization have the skills set to effectively manage and administer a particular network operating system?

In most cases an organization will be able to capitalize on an alternative that is available for little or no cost. It may be that a desired alternative is in use in another system in the organization. These "free" characteristics of the target system should be leveraged.

It may be that the existence of these less expensive alternatives may even alter the migration plan. The benefits of these characteristics may be so overwhelming that, despite not providing all of the desired openness, they may provide a significant portion of the budget to be used elsewhere.

For example, the organization may have one or more system administrators that, from a previous assignment or job, have extensive experience on application development in C++. While the goal may have been to establish an application development environment using an object-oriented RAD tool such as Delphi, the organization may opt to develop the new system in C++ and delay plans to standardize on Delphi. By doing this, they may save extensive training dollars on this system that may be needed in another characteristic that is woefully "un-open," such as their network infrastructure.

The result of this analysis should be a single package of openness characteristics that will form the target system.

F. STEP SIX: IMPLEMENT THE PLAN

The final step in this strategy is to implement the plan. This will not be a trivial undertaking. At a minimum, the implementation phase will include the following:

- Contracting for hardware, software, and services
- Sequencing analysis (Which characteristic should be completed first? Hardware? Network? Operating System?)
- Software development (Design, coding, testing, review, prototyping, etc.)
- Possible construction
- Conversion analysis (Will the new system be converted directly, in parallel with the old system, in phases, or through a pilot?)
- Training and education

- Policy modifications.

G. CONCLUSION

The goal of this strategy is to produce systems that can evolve and grow with changing business requirements. This strategy recognizes that organizations are working with shrinking IS budgets and provides a commonsense and in-depth approach to leveraging open systems for the accomplishment of their mission.

V. A CASE STUDY: NAVAL POSTGRADUATE SCHOOL

The goal of the strategy discussed in the last chapter is to move from legacy systems to distributed applications built on open standards, leveraging the desktop, sharing information, and empowering the user. In this chapter we apply our methodology to NPS.

Before we can apply the strategy, however, we must provide necessary background information about NPS. The focus of our research is limited to the accounting system at NPS, which is the target system for the illustration of our strategy. Because it is the target of our strategy, we must also include necessary background on DoD accounting principles.

This case study is not meant as a prescription for a specific course of action. Rather, its purpose is to illustrate how our strategy can be applied to a real world organization.

A. BACKGROUND - THE NAVAL POSTGRADUATE SCHOOL

NPS is uniquely postured as a re-invention lab for DoD, having a vast array of resources available to tackle the hard IT issues that face DoD and the federal government in general. Part of these unique resources are the top-notch civilian faculty and almost two thousand highly motivated military students. In addition, NPS has several hundred PCs, Sun Workstations, and Apple computers, as well as a 1990 Amdahl Mainframe and a Cray J90 for computational-intensive research. Finally, NPS has a fair amount of connectivity as well as access to the Internet.

NPS is a major command within DoD, and endeavoring to understand how to develop systems supporting the Defense Information Infrastructure (DII) provides a rich area of research at NPS. The DII is "a seamless web of communications networks,

computers, software, databases, applications, and other capabilities that meets the information processing and transport needs of DoD users in peace and in all crises, conflict, humanitarian support and wartime roles. It includes physical facilities, applications, network standards and protocols, people and assets [Ref. 12]."

The basis for the following background information is the Strategic Plan for Computing at the Naval Postgraduate School [Ref. 13].

The Naval Postgraduate School encounters many of the information technology issues found in DoD: multiple network operating systems, various levels of cabling, stovepipe (non-integrated) application programs, and weak infrastructure. The current focus of IT at NPS is on enhancing the campus network, analyzing computing resources, and developing improved administrative systems.

1. The Network

The existing network at NPS falls well short of meeting academic or administrative needs. The cabling requires a major investment to bring it up to reasonable standards. In general, additional investments are needed to organize the network, improve information, and manage the network more reliably and efficiently. According to the Strategic Plan for Computing, a total network investment of about $4 million spread over two or three years would bring the School up to an adequate standard. In addition, it further states that four additional GS billets are needed to support an enhanced network infrastructure.

2. Computing Resources

a. The mainframe

Usage on the Amdahl mainframe has changed substantially over the years. Most research and instructional computing has migrated to PCs and workstations (as well

as to the Cray for processor-intensive work) away from heavy reliance on the mainframe for computing power. Some mission-critical accounting and registrar applications still reside on the mainframe. Transaction volumes of these applications are fairly modest, and could easily be moved to a smaller machine if software conversion were possible.

b. Processor-Intensive computing

This type of computing is used extensively in modeling for meteorology and oceanography. Such computing will remain an essential element of NPS' computing portfolio. The Cray J90 has just been recently installed in a no-cost swap of the previous Cray. This gives NPS a modern machine that incorporates contemporary technology and low maintenance costs; it also provides considerable growth potential. With relatively modest additional upgrades, the J90 should be capable of meeting the School's need for high-performance computing over the next few years.

3. Administrative Applications

Existing administrative applications require reengineering in order to achieve minimum efficiency and accuracy. Currently, almost all automated applications were developed as "stovepipe" applications without considering the strategic implications. The applications vary greatly in their functionality, quality of implementation, and adherence to contemporary design practices. NPS continues or plans to install new stovepipe applications that conflict with enterprise-wide needs. This will result in excessive labor costs and the inability of managers and staff to obtain accurate and timely information needed to perform their functions. NPS will not realize the savings of effective data management because current administrative systems cannot share mission-critical data. This leads to highly inefficient and costly duplicate data entry and storage.

B. BACKGROUND - DOD ACCOUNTING

This section provides general information concerning the accounting requirements at NPS. The information presented in the following subsections was derived from the text used in the Navy Comptrollers Course taught at NPS [Ref. 14].

1. Official vs. Unofficial Accounting Records

Most activities establish and maintain three sets of accounting records. The official records, maintained by the activity's Defense Accounting Office (DAO), are the source of standardized accounting reports to higher authority. The activity comptroller maintains unofficial records for the entire activity. Finally, each cost center within the activity, such as an academic department in the case of NPS, maintains another set of unofficial records for its own use. The unofficial records in the latter two categories are sometimes referred to as memorandum records.

Activity memorandum records are normally locally designed and administered by the Comptroller Department. They are used as a medium to provide near real-time financial status of the activity and as an independent source of data to reconcile against the official DAO records. Official DAO records can sometimes be inaccurate and dated due to input errors, processing delays, and computer down time. Without locally maintained memorandum records, activity comptrollers would be vulnerable to miscalculation of their true financial standing and over-obligation of funding authority.

Cost center accounting records (PC spread sheets, manual ledger books/OPTAR logs) are used by cost centers to provide near real-time financial status and to reconcile against activity memorandum and official DAO records. Since obligations originate at the

cost center level, it is crucial that these records be kept current and accurate at all times. These records are the focus of our case study and the prototype system we designed.

2. Legal Requirements

There are three primary limits of an appropriation: time limits, purpose limits, and dollar limits. Time limitations (obligational/expenditure availability periods) are based on the type of appropriation and determine when all unspent funds "expire" by law. The other two limits, purpose and dollar limit, are part of United States law as follows:

> 31 U. S. Code Section 1301(a) - requires that appropriated funds only be used for programs and purposes for which the appropriation is made. Currently, there are no reporting requirements associated with these violations. However, when a violation has been determined, adjustments must be recorded. If the adjustment results in an over obligation or over expenditure of the appropriation of fund charged, a 31 U.S. Code Section 1517 has occurred, and a report of violation must then be prepared. Commanding Officers must ensure that violations do not occur by having adequate controls.
>
> 31 U.S. Code Section 1517 - prohibits any officer or employee from making or authorizing an obligation in excess of the amount available in an appropriation or subdivision thereof (operating budget/allotment) or in excess of the amount permitted by agency regulations. It also requires that the person who caused the violation may be subjected to discipline which may include suspension without pay or removal from office. If action is done knowingly and willfully, that person may be subject to criminal penalties of a fine up to $5,000 or imprisonment for not more than two years, or both.

The U.S. Code also states that Congress can further limit how money is spent in any way it wishes. Examples of this type of limit are setting caps on how much money can be put into Morale Welfare, and Recreation (MWR) activities, limiting the hiring of

civilian employees, or limiting how much can be spent in Operation and Maintenance, Navy (O&M,N).

This information is provided to emphasize the importance of maintaining accurate accounting records. It further emphasizes the need for real-time balance information and reporting of discrepancies as early as possible.

3. Job Order Cost System

The Navy's Job Order Cost System is a detailed cost accounting tool used to facilitate proper recording and classification of costs. Costs are classified and accumulated by assignment of job order numbers which are related to the various categories into which costs are classified. Our focus is on labor and project job order numbers.

Job order numbers are structured to provide information as to who has spent funds and for what purpose. They provide details at the subactivity group, functional category, subfunctional category, cost account code, and, when necessary, the expense element level. Figure 9 provides an example of a typical job order number.

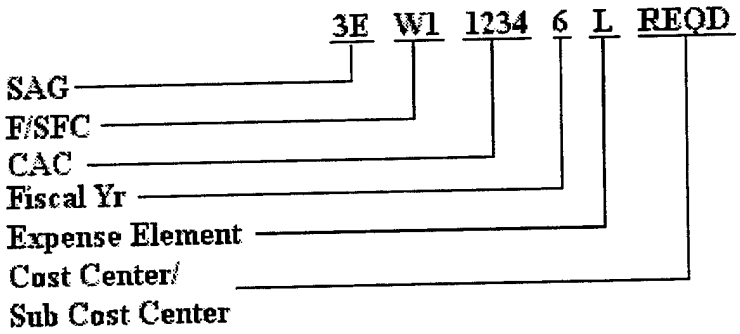


Figure 9. Breakdown of a job order number from [Ref. 14]

Job order numbers (codes) are annotated on all documentation for procurement, consumption/application and accounting for operating resources at NPS.

When assigning job order numbers on documentation, input accuracy is critical. Input personnel must be competent and they must be motivated to go through the trouble of looking up the correct code. Potential accounting code input problems can be overcome through awareness and proficiency training, supervisor interest and oversight, and with the help of validation systems.

When dealing with reimbursable funds, it is very important that germane reimbursable job order numbers be charged for actual work performed/expenses incurred in connection with that specific reimbursable order. Failure to charge appropriate reimbursable job order numbers can result in erroneously charging direct funding job order numbers. The majority of job order numbers utilized in the academic departments are for reimbursable funds.

C. STRATEGY APPLIED TO NAVAL POSTGRADUATE SCHOOL

Now that we have provided necessary background information on NPS and DoD accounting, we can begin the illustration of our strategy.

1. Step One: Baseline Analysis

Many systems are currently under review because of NPS' status as a reinvention lab. However these studies are not of the type we advocate in this step. Here NPS would conduct a cursory review of its major systems, concentrating on the openness infrastructure. This is a macro-level analysis, not requiring extensive resources or time. The goal of this step is to gain a feel for characteristic trends and systems that may require migration.

2. Step Two: Identify Systems That Require Migration

The memorandum accounting system at NPS is a leading candidate for migration. The current system actually consists of several disparate legacy systems. These systems range from manual pen and paper systems to more sophisticated desktop computer systems such as DBase, and Paradox. However, none of these systems work together. The current system is highly ineffective, extremely costly to maintain, and incapable of growth.

The Comptroller's office uses a DBase III+ application developed by a former employee several years ago. As the business environment changed over the years, this program was modified with great difficulty. One of the reasons is that the system contains virtually no documentation. Each of the modifications were performed by other employees to fix a short-term crisis that would allow the office to continue to function. In other words, the modifications were not part of a planned and coordinated effort to move the system into a more productive role.

Besides the poor maintenance record, the system is simply not meeting the needs of the users that depend on it - the customers at the academic departments. Because of a poorly designed original system, data entry is extremely time intensive. Every field in the flat file database requires at least one keystroke by data entry personnel, even though only 10-20% of the original fields are being used. The user interface is antique - a character-based environment, requiring cryptic key combinations to perform even the simplest task.

Because of this poor design and the recent acquisition of Public Works and Presidio of Monterey labor accounting into the NPS system, the Comptroller's data entry staff cannot keep up with the work flow. Seven full-time technicians, many working long

overtime hours, are dedicated to simply transferring the data from timecards into this primitive system. This workforce seems excessive to perform its function. However, consider the potential number of labor charges for a single Public Works employee. For Public Works to accurately charge customers, they must account for worker labor down to 30 minute increments. This could result in as many as 160 separate labor records per pay period for each employee. The office is often weeks to months behind schedule, despite the workers' hard and long work. This means that users at the department level do not get accurate representations of their account status. The situation is especially chaotic at the end of the fiscal year when users attempt to spend exactly what is left in their accounts, without breaking the law and overspending.

The current system is also plagued by duplication of data entry and redundant storage. For cost center accounting, each academic department maintains its own set of data. This data is not shared and must be manually entered on several counts so that all sub-systems get updated. In addition, the data that is input at the cost center level and then again at the Comptroller's office - for memorandum accounting- must again be entered to update the official accounting records. Not only is this practice prone to errors, it is labor intensive, and therefore extremely costly. Because all of these sub-systems act independently, at any given point the users are presented with conflicting figures on account balances.

Still another shortfall of the current system is its inflexible reporting capability. Poor database design inhibits ad hoc queries on specific accounts during a particular time period. The original system design called for a set of standard reports that, today, are highly obsolete.

In short, the current memorandum accounting system at NPS does not meet the requirements, is difficult and costly to maintain, and does not take advantage of the School's technology infrastructure. What is needed is a modern, more open system that will evolve with changing business requirements. Because of the labor-intensive nature of the existing system, and the high costs associated with it, it is quite possible that a new system would be able to pay for itself in a relatively short period of time.

3. Step Three: System Requirements Analysis

The accounting system is in serious need of migration to a more open environment that will permit change and evolve with changing business requirements. Before the issue of openness is addressed, however, NPS must analyze the functional requirements the system must meet. These requirements are listed in Appendix B. A system that meets these requirements is not necessarily guaranteed to endure the certain changes that will challenge NPS and the DoD in the future. What is needed is a detailed analysis from an openness perspective.

4. Step Four: Openness Analysis

This step of the strategy requires detailed analysis using the Openness Continuum Model. Each characteristic *down* the model is analyzed for available alternatives *across* the model (from least open alternative to most open alternative).

a. Identify competing alternatives

(1) Applications. As we stated in Chapter II, the applications characteristic is perhaps the most critical in the model. If developed properly, the application can take full advantage of the open qualities of the other characteristics. If developed improperly, the new system will not be capable of its potential openness.

Our discussion in the previous step of the strategy showed that the existing system application is extremely closed. NPS is currently considering several alternative applications.

FastData is a DOS-based, mid 80's technology product that would be provided "free of charge" by DoD. It was designed to meet the needs of a specific organization. It was not developed with a view toward making it the DoD standard; as a result, it is not adaptable, transportable, or capable of meeting all of NPS accounting requirements. The user interface is arcane, it relies on excessive use of keyboard codes, and would be a step backwards from the existing systems. While this system would be an improvement in the transfer of data from the Comptroller's office to the official records, it does nothing to improve the duplication of data from the academic departments to the Comptroller's office and the inaccuracy of account balances would still be a problem.

The other "free" DoD software package is Electronic Time and Attendance Certification (ETAC), which is being developed locally (at considerable cost). This software is focused entirely on labor accounting processing within the Comptroller's office. It does improve upon the current user interface of the DBASE III application. However, it is simply replacing one stovepipe system with another more aesthetic stovepipe system and does not address the problems we have discussed thus far.

NPS is seriously considering commercial off the shelf (COTS) products to fulfill the needs of this mission-critical system. We suggest that the majority of these products would not be a step in the right direction. Most offer proprietary and closed data formats. Most are also generic in nature and would not be capable of

performing in the unique environment imposed by government regulation. Some, however, show promise and deserve a closer look.

One such product is Solomon IV. This product appears to offer a high level of configurability. In addition, the company claims the data can be stored in a number of DBMS products, such as Oracle, giving it a high degree of openness. The test for Solomon IV is the ease with which the data can be accessed and manipulated.

The most open alternative would be a custom built application using a modern, robust development tool. This approach would allow the School to build a system that (a) meets exactly its requirements, (b) permits development of a system that could be scaled upward, and (c) takes advantage of existing infrastructure and available technology. A commitment to a specific development environment would almost ensure long-term growth potential.

Because of the importance of the applications characteristic, we have developed a prototype labor accounting system using Borland Delphi Client/Server Version 2.0, an object oriented rapid application development tool. This application embodies the principles of open systems and distributed applications. The package also comes with a fully integrated set of tools for data management and manipulation, reporting, and data migration.

Our prototype is an illustration of what a modern, robust development environment can yield. This application was developed by two part-time developers in about one month. Although this is only a prototype and addresses only labor accounting, we believe similar results can be achieved for the other modules necessary to make this a mission-critical enterprise information system. Interface samples

and full source code are included in Appendix C. Database specifications are included in Appendix D.

Of course, this approach would require an investment in qualified programmers, administrators, and tools. This expertise does not come cheaply. Many organizations simply cannot afford to recruit and then retain the necessary personnel. This is especially true in DoD, where organizations are constrained by hiring and compensation regulations. In essence, DoD is priced out of the market for professional developers because of the high demand and superior opportunities that exist in the corporate world.

(2) Hardware. The current system runs predominately on mid-range PCs. Despite their relatively low processor speeds and memory, in terms of openness, they are quite adequate for the task. This hardware base permits the target system to be written for a multitude of platforms. Currently, all systems are capable of running Windows 3.x operating system, and most are capable of running more powerful operating systems in the PC arena. The alternatives that permit a more open system are merely upgrades to the existing hardware. These upgrades would permit the system to run more efficiently from Windows 95 and Windows NT. Recommended minimum requirements for hardware upgrades include at least a 486 microprocessor for both operating systems and 16 megabytes memory for Windows 95 and 32 megabytes memory for Windows NT.

(3) Network. The existing system relies very little on the network characteristic. The Comptroller's office runs its DBase III application over an office LAN. This is only so that several technicians can work simultaneously. They are not, however, accessing the same data. The application has been apportioned and each technician works

on a separate portion of the workload. As stated before, academic departments use stand-alone systems and are in no way connected to the Comptroller's office. The existing system is essentially a closed system when it comes to the network characteristic. The target system will certainly rely more heavily on network technology.

Unfortunately, the School's existing network infrastructure may be incapable of supporting the openness requirements of the accounting system. Its problems are well documented in the NPS document, *Strategic Plan for Computing at the Naval Postgraduate School*, which outlines existing network deficiencies in four major areas: the cable plant, the network operating system, network access, and network management.

This document offers a specific and realistic vision of a reliable and open network infrastructure that will serve the School's mission well into the next century. The future NPS network is an example of the cooperative environment discussed in Chapter III. The target accounting system will be the beneficiary of network infrastructure improvements brought on by the overarching needs of the entire campus.

The system requirements essentially dictate a server-based network that would permit access from any machine on the campus. The alternatives would come down to using the existing network infrastructure and patching together fixes to support the system or waiting for the proposed upgrades to be completed and using it. These two alternatives essentially mean the same thing for the target system; it will be supported by the necessary network infrastructure.

(4) Operating System. The existing system runs on several operating systems. The Comptroller's office uses MS-DOS Version 5.0. The academic departments run various operating systems from DOS to Windows 3.x to Macintosh. The

use of DOS and the fact that incompatible operating systems are used, puts the existing system at the low end openness spectrum.

One alternative to a more open operating system would be standardization on Windows 3.x. While this would be a step in the right direction and could most likely meet the current requirements, its future as a viable operating system for mission-critical systems is questionable.

Windows 95 would be even more open. This operating system is capable of running more modern, 32-bit applications and includes a full TCP/IP stack on every client. It is fully backward compatible to all Windows 3.x and DOS applications. It has yet to gain the full acceptance of corporate America, however, partly because of the required hardware investment, and partly because of its vendor's (Microsoft's) long-term commitment to its powerful Windows NT. Microsoft has publicly stated its intention to migrate Windows 95 to this operating system.

Because of this long-term commitment to Windows NT, and Microsoft's status as industry leader, we would consider NT to be towards the open end of the operating system spectrum.

(5) Data. The data characteristic of the existing system again falls toward the closed end of the continuum. Documentation of data structures in all subsystems is non-existent or very poor at best. Data is not relationally stored and cannot be accessed from outside different sub-systems. This requires duplication of data entry several times over.

A more open solution would be to standardize throughout the School on a COTS desktop database package for data entry. Academic departments

could then complete data entry at their worksite, transfer the data to floppy disk, and the Comptroller's office could simply copy the data to their system. While this would solve the problem of duplication of effort, it would not solve the problem of giving timely, accurate, and flexible reporting down at the department level. It also does not take advantage of the existing infrastructure or the available technology. Finally, it also goes against the principles of distributed computing discussed in Chapter III.

The existing system and the alternative just discussed are extremely shortsighted. These systems require the use of data that is maintained elsewhere in the organization, but because they are stovepipe in nature, they require duplicate copies of the data that most likely will not be accurate or timely. For example, the accounting system will require extensive use of data on employees such as personal data and pay information. This data is and should be maintained by the Human Resources Office.

A more open solution would be for the accounting system to use the same data that HRO uses. One way to do this is through the use of a distributed database. Both the Solomon IV and custom-developed application solutions would be capable of such a data model.

A third, and most open alternative, is the use of the School's existing industrial-strength relational database management system, Oracle 7, within a campus-wide federated database. This would permit the exploitation of intranet technologies discussed in Chapter III. Both Solomon IV and a custom-developed application solution are capable of this data model.

(6) Skills Set. The users of the existing system have the requisite skills to perform the required tasks. In fact, the users' skills are overburdened by the manual and tedious nature of the existing system.

That will not be the case if the School migrates to a modern, open, enterprise system. Regardless of the target system, the School will require investment in database administration, network, and Windows 95/NT expertise. If a custom-developed application is chosen, the School should make a long-term commitment to training and supporting a modern, powerful development environment. Naturally, skills in the tools to support that environment should also get the same commitment.

(7) Tools. The existing system requires virtually no tools to perform its mission. A move to a more open system would require tools that would assist developers in rapidly producing, testing, and deploying workable solutions to the users' needs. Once it is deployed, these systems will aid in the modification and maintenance of code and the database. Several products offer all of these tools in an integrated package. The application development environment chosen would dictate which tools to purchase.

b. Pare Down the Alternatives

Within the applications characteristic, several alternatives can be removed from further consideration. The two DoD applications, FastData and ETAC, fall well short in several system requirements (character-based, lack of ad hoc reporting, duplicate data entry, etc.) and do not meet even the minimal open system goals (open data standards, client access of data from any platform, etc.). Any COTS product that does not support open data standards can also be eliminated. To date, the only affordable COTS alternative is Solomon IV.

In the data characteristic, the alternative to use COTS databases and floppy disks to transfer data can also be eliminated. This solution will not provide a solution to the requirement of reliable, accurate access of funds status by system users.

c. Define Competing "Packages" of Alternatives

This system's success will be driven by the strength of the applications and data characteristics. The hardware and operating system characteristics are not critical because the school's existing computer base is sufficient to support any solution. The network characteristic is essentially dictated by requirements. The skills set and tools heavily depend on the applications and data alternatives chosen.

Four packages that deserve further analysis are: Solomon IV with a COTS DBMS using either a distributed database model or a federated database model, and a custom-developed application using either a distributed database model or a federated database model. These packages are represented in Figures 10, 11, 12, and 13.

Characteristic	**Alternative**
Applications	Solomon IV
Hardware	Existing hardware or upgrades
Network	Server-based with or without TCP/IP
Operating System	Windows 3.1, Windows 95, or Windows NT
Data	Distributed database w/COTS DBMS
Skills Set	Existing workforce and either consultants or additional hires
Tools	COTS DBMS, Network Management Tools

Figure 10. Package 1

Characteristic	Alternative
Applications	Solomon IV
Hardware	Existing hardware or upgrades
Network	Server-based with or without TCP/IP
Operating System	Windows 3.1, Windows 95, or Windows NT
Data	Oracle 7 within a campus-wide federated database
Skills Set	Existing workforce and either consultants or additional hires
Tools	Application environment integrated tools, Oracle 7 tools, Network Management Tools

Figure 11. Package 2

Characteristic	Alternative
Applications	Custom-developed
Hardware	Existing hardware or upgrades
Network	Server-based with or without TCP/IP
Operating System	Windows 3.1, Windows 95, or Windows NT
Data	Distributed database w/COTS DBMS
Skills Set	Existing workforce and either consultants or additional hires
Tools	COTS DBMS, Network Management Tools

Figure 12. Package 3

Characteristic	Alternative
Applications	Custom-developed
Hardware	Existing hardware or upgrades
Network	Server-based with or without TCP/IP
Operating System	Windows 3.1, Windows 95, or Windows NT
Data	Oracle 7 within a campus-wide federated database
Skills Set	Existing workforce and either consultants or additional hires
Tools	Application environment integrated tools, Oracle 7 tools, Network Management Tools

Figure 13. Package 4

5. Step Five: Cost/Benefit Analysis and Selection of Target System

As we have previously stated, a formal cost/benefit analysis is beyond the scope of this research. We do, however, offer some critical considerations.

Either solution using a COTS DBMS in a distributed database scenario would be short lived. It would not permit exploitation of intranet technologies.

At the present, industry leader Oracle would provide the most open alternative in the data characteristic. This powerful DBMS should be the cornerstone of all future enterprise systems at NPS. As we have found out, however, development using this asset is not trivial. The school must be prepared to make a substantial investment in quality personnel to administer it.

The Solomon IV solution may cost less in the short term. However, the School should seriously consider how this solution will fit into its long-term goals. For example, other administrative systems will require migration to a more open environment in the future. Solomon IV must be further analyzed to determine how well its data can be integrated with other enterprise systems.

Any custom-developed solution will require the acquisition of qualified programmers. The School must consider if it can support a long-term commitment to this approach. NPS should also realize that this commitment will only require a small team (two to three) of developers.

6. Step Six: Implement the Plan

The School's network infrastructure should be the top priority. Without this critical element, any new system's potential openness will never be realized.

At the same time, the school should make a commitment to a standard custom application development environment. This may require another independent cost/benefit analysis to determine the best tool to meet the School's needs. If and when a commitment is made, it should come in the form of a qualified core of programmers and a training plan

to retain them. Their role would include the development and maintenance of new enterprise systems and support to departments that develop in this standard environment.

The target accounting system should initially be developed so that it can be run over the existing network infrastructure. It should provide the flexibility to easily scale up to the more modern network the School envisions. With the properly chosen development environment, this should not be a major dilemma.

Although we have not discussed organizational issues to this point, we reiterate that these problems will be the most difficult to overcome. It will be important to sell the benefits of this paradigm shift, educate and train personnel, and include them in the development process.

D. CONCLUSIONS

The goal of this strategy is to include the notion of open systems across the entire range of a system's development. By doing so, an organization is able to deploy systems that will better cope with changing business requirements and provide lasting residual value.

VI. CONCLUSIONS/RECOMMENDATIONS

A. SUMMARY

Traditionally, organizations have dealt with changing business requirements by building a stovepipe system that addresses only the immediate problem. This practice of deploying legacy systems is no longer a viable solution in today's fiscally constrained business environment. This is especially true in DoD, which faces a 20-30% budget reduction over the next five years.

The drastic corporate downsizing of the past decade has pushed decision making away from centralized control. A flatter, leaner organizational structure is the norm for today and the immediate future. This has led to the same decentralized structure in the IS world as well.

In the late '80s and early '90s, client/server was the preferred method of downsizing information systems. Large mainframe applications were broken into separate applications and moved onto PCs . This migration provided many benefits such as shorter development cycles, improved UI, more efficient use of computing power, and flexibility. However, organizations soon discovered the hidden costs and difficulty in this approach. Client/server without open standards required complex middleware which proved costly to implement and manage.

The current movement toward open systems makes client/server computing easier to accomplish. Better yet, it facilitates the use of distributed systems and modern technologies. Open standards reduce an information system to a set of commodity components that are better suited for replacement when they become obsolete. Paul Strassmann refers to this approach as a "snap-in, snap-out, disposable type of economy,"

which leads to the preservation of an organization's long-term assets - data and software [Ref. 3].

However, the prospect of developing an open system is a daunting task This research provides a methodology to tackle this effort. We propose a bottom-up review of all characteristics of an open system as defined by the organization. Through use of the Openness Continuum Model, organizations define competing packages of characteristics. Cost/benefit analysis of these packages will then lead to the system that best meets the desired degree of openness.

In conjunction with this research, we developed a prototype labor accounting program. Building this application provided valuable lessons and insight into development challenges for enterprise systems.

B. OVERVIEW OF PROTOTYPE APPLICATION

1. What Is it?

The prototype labor accounting system we developed, *D-Books*, is a 32-bit, multi-threaded Windows 95/NT application. It also demonstrates the use of intranet technologies for universal data access in a mission-critical application. We developed it using Delphi version 2.0, an object-oriented rapid application development tool from Borland.

2. What Can it Do?

D-Books provides the following functionality:

- Add/Edit/Delete of employees, departments, pay history, job orders, funds, and labor records
- Compliance with essential labor accounting business rules as determined by an in-depth requirements analysis

- Data integrity through two-phased commit transaction processing
- Proof of concept reporting capability
- Scalable from the department level to the Comptroller's office

3. How Is it Open?

D-Books was designed using Local InterBase Server and then ported to Oracle. This facilitates open data access. It also capitalizes on the School's large investment in and commitment to Oracle.

Though D-Books is limited to Windows 95 and Windows NT, portions of the application were ported to an intranet platform. Intranet D-Books was designed using HREF Tool's WebHub technology and Microsoft's Internet Information Server. This effectively permits enterprise-wide access to the data.

D-Books implements a logical three-tier architecture. The three tiers are the database server, the business logic, and the client UI. The business logic exists on the client but is logically separated from the UI code in the form of Delphi's new data module component (See Figure 14.). The use of this powerful component enabled us to easily port this application to its intranet implementation. We simply replaced the UI code with HTML. No modification of business logic was necessary. This is critical as it will enable D-Books to scale upwards as application server technology matures.

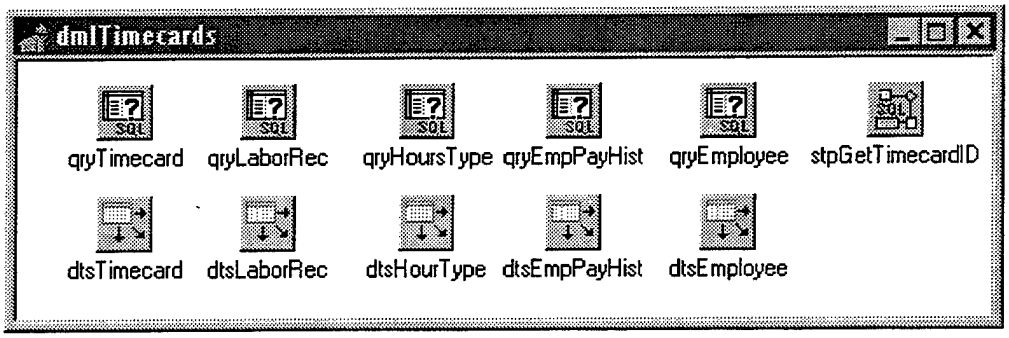


Figure 14. Example D-Books Data Module

4. Lessons Learned

a. Small Development Teams Using Modern Tools Can Achieve Extraordinary Productivity

D-Books was developed in about two man-months using Delphi and WebHub. Although this application is only a prototype and only fulfills a portion of the School's needs, we strongly believe that this project can be completed to production-quality standards in less than ten man-months by a team of two to three competent developers with only a familiarity of Delphi and fundamental skills in programming and database design.

b. A Sound Database Design Is Essential to Application Success

We originally developed D-Books using Delphi version 1.0. Before setting out to port our application to a 32-bit architecture, we resolved to ensure our data model was sound because we discovered that data changes were increasingly more difficult to deal with the further along in development we were. By taking the time up front to ensure we had a 90% solution, our development effort was infinitely more productive.

c. Early Adoption of Coding Standards Is Key to Effective Teamwork

Our first attempt at D-Books was hampered by confusion about naming of components and variables. Our code was difficult to read, especially that which was written by the other partner. At the outset of our second attempt, we developed a coding standard that outlined coding format and naming of components and variables. As a result, our code was easier to read, and, more importantly, we could pick up where our partner left off with no loss in efficiency.

C. RECOMMENDATIONS

1. Investment in the Campus Network Should Be the School's Top Infrastructure Priority

Successful deployment of modern, open systems is heavily dependent on the network infrastructure. Without it, the School will never be capable of capitalize on current technologies that leverage the network to permit open data access and sharing. We recommend wholesale adoption of the network upgrades as outlined in the *Strategic Plan for Computing at NPS.*

2. Further Analyze Solomon IV as a Viable Solution

The Solomon IV solution shows great promise. However, this product and the vendor have no experience with government accounting. The performance of this product has yet to be proven. We recommend careful analysis into all costs associated with this solution. Potential hidden costs must be investigated prior to its adoption. The vendor has yet to work out licensing arrangements with Oracle. In light of Oracle's expensive history, this factor cannot be overlooked. Because the product will require extensive configuration to meet the School's needs, this solution will require a considerable

investment in consultation and support from the vendor. Finally, this solution may require upgrades in the School's existing hardware base.

3. Investigate the Requirements of Establishing an In-House Custom Development Environment

We have previously discussed the difficulties with custom application development. However, our prototype application also demonstrates the extraordinary results that can be achieved with a small, skilled development team using modern tools. The School should investigate if this is a feasible solution. We submit that the School would be well served to assemble a core team of three personnel: an application developer with experience in a modern tool such as Delphi, a database administrator with experience in Oracle, and a web master to manage the intranet.

D. PROSPECTIVE AREAS OF RESEARCH

1. Cost/Benefit Analysis of Solomon IV

This is an excellent opportunity for a thesis student. This research should answer the following questions:

- In light of Solomon IVs lack of experience in government accounting, can the package meet fundamental requirements?
- Can the data be exported to Oracle 7 as the vendors purport?
- Does the documentation of the data model facilitate easy access to the data?
- What are all of the costs (licensing, consultation fees, required hardware upgrades, etc.) associated with this approach?
- Would it enable the School to capitalize on intranet technologies?

2. Organizational Issues

Evaluation of any new enterprise system will bring with it organizational challenges. These challenges are most likely to be more difficult and more costly than the development of the system itself. Each package will produce a unique set of organizational issues. Analysis of them will provide a rich source of research.

3. Continuation of Enterprise Accounting System Development

While the School may not be able to assemble the necessary staff to complete D-Books, this would be an excellent opportunity for a thesis student and it would greatly benefit the School. This research could investigate techniques that would reduce life-cycle support. Perhaps through this research the School would obtain a mission-critical system that requires considerably less maintenance.

LIST OF REFERENCES

1. Gartner Group conference presentation, *Client/Server Computing: Where Does it Lead?*, Gartner Group, 1994.

2. Brodie, Michael L., *Migrating Legacy Systems*, Morgan Kaufmann Publishers, Inc., 1995.

3. Strassmann, Paul A., *Change Management Through Open Systems*, http://www.strassmann.com/pubs/open-systems.html, 1996.

4. Institute of Electrical and Electronic Engineers (IEEE), *Draft Guide for Information Technology - Portable Operating System Interface (POSIX) - The Open Systems Environment, P1003.0/D16,* Portable Applications Standards Committee of the IEEE Computer Society, August 1993.

5. Sprague Jr., Ralph H. and McNurlin, Barbara C., Information Systems Management in Practice, Prentice Hall, 1993.

6. Orfali, Robert, Harkley, Dan, and Edwards, Jeri, *Essential Client/Server Survival Guide*, John Wiley & Sons, Inc., 1994.

7. Berson, Alex. *Client/server Architecture,* McGraw-Hill, 1992.

8. Sarna, David E., and Febish, George J., Client/Server Scalability Myth, *DATAMATION*, September 15, 1994.

9. Hicks, J., D., *N-tier Client/Server Architecture*, http://www.vsol.com/jdart.html, 1996.

10. Strom, David, *Creating Private Intranets: Challenges and Prospects for IS*, http://www.strom.com/pubwork/intranetp.html, November 16, 1995.

11. Derfler, Frank J., "The Intranet Platform: A Universal Client?," *PC Magazine*, April 23, 1996.

12. *Defense Information Infrastructure*, http://dubhe.cc.nps.navy.mil/~interop/dii.html, August 10, 1996.

13. Emery, James, *Strategic Plan for Computing at the Naval Postgraduate School*, unpublished report, Naval Postgraduate School, Monterey, California, November 2, 1995.

14. *Practical Comptrollership*, Naval Postgraduate School, Monterey, California, September 1994.

APPENDIX A. ACRONYMS AND TERMS

API	Application Programming Interface
CASE/I-CASE	Computer Aided Software Engineering/Integrated-CASE
CGI	Common Gateway Interface
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off The Shelf
CPU	Central Processing Unit
DBMS	Database Management System
DII	Defense Information Infrastructure
DoD	Department of Defense
ETAC	Electronic Time and Attendance Certification
FTP	File Transfer Protocol
GUI/UI	Graphical User Interface/User Interface
HTML	Hypertext Markup Language
I/O	Input/Output
IDAPI	Integrated Database Application Programming Interface
IEEE	Institute of Electrical and Electronics Engineers
IS	Information System(s)
ISAPI	Internet Server Application Programming Interface
ISO	International Standards Organization
JON	Job Order Number
LAN	Local Area Network
NPS	Naval Postgraduate School
ODBC	Open Database Connectivity
OLE	Object Linking and Embedding
OSI	Open Systems Interconnection
PC	Personal Computer
RAD	Rapid Application Development
RAM	Random Access Memory
SMTP	Simple Mail Transport Protocol
SQL	Structured Query Language
TCP/IP	Transport Control Protocol/Internet Protocol
TKAL	Time Keeping and Labor
TP Monitor	Transaction Processing Monitor
WAN	Wide Area Network
WWW	World Wide Web

APPENDIX B. MEMORANDUM ACCOUNTING SYSTEM REQUIREMENTS

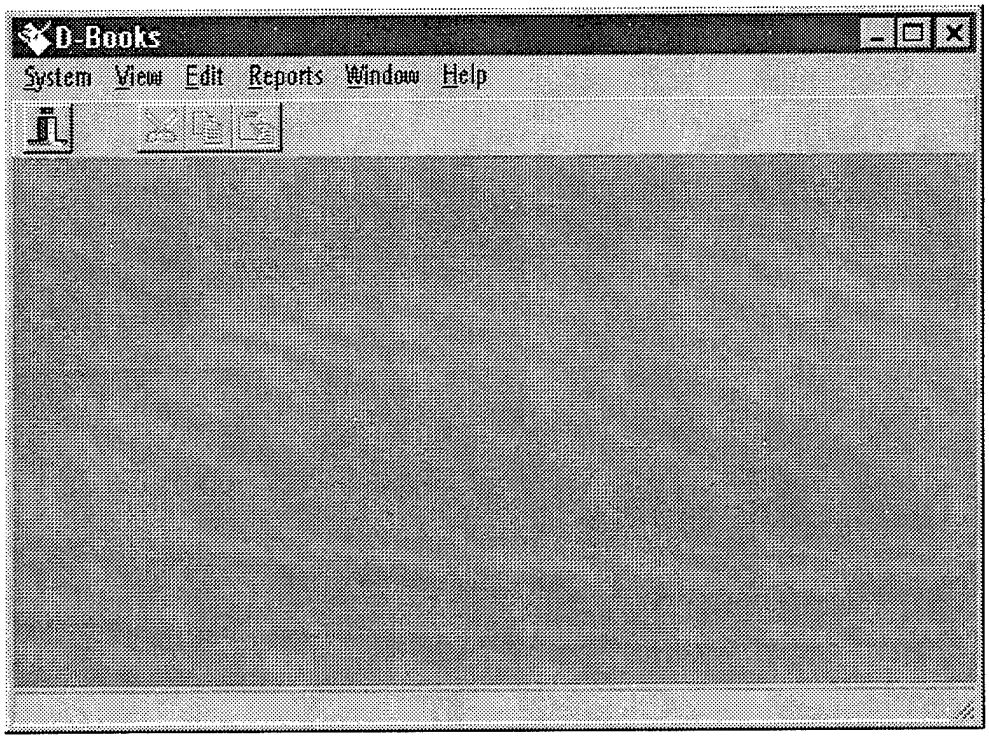
Item Number	REQUIREMENT DESCRIPTION
1	Must meet all legal and government accounting requirements.
2	Must utilize an open data system where accounting information is stored so that other applications can be quickly developed as requirements change.
3	Provide core relational data model to support command-wide accounting requirements.
4	Provide necessary security for safeguarding information from outside intrusion or to prevent unauthorized access from within NPS.
5	Support existing infrastructure for central system: • Network connections and protocols • Database Server
6	Ability to conduct spot checks on data entered at departmental level.
7	Ability to generate reports required by Mezzanine.
8	Ability to update central database as funding allocation changes.
9	Ability to easily inform departmental accounting technicians of changes.
10	Ability to process timecards from academic departments automatically.
11	Automatic exception reporting for any business rule violations, i.e. obligating over funding levels.
12	Provide department-specific data model linked to central data model.
13	Control access to data within the academic department.
14	Provide real time reporting to project managers.
15	Process timecards automatically.
16	Provide audits of project expenditures.
17	Ability to quickly enter information on all accounts.
18	Ability to check data entry automatically for business rule violation.
19	Ability to modify software so that new business rules can be added.
20	Ability to provide paper reports required by academic chairman/staff.
21	Ability to make corrections easily.
22	Ability to submit timecards to comptroller electronically.
23	Ability to perform drill down queries on account information based on account, accounting period, and types of items charged.
24	Ability to access accounting information for accounts responsible for independent of geographical location.
25	Accurate, timely, reporting.
26	Ability to submit time card information electronically to department accounting personnel.
27	Client access must be platform independent.
28	Track all types of accounts—Reimbursable Research, Direct Research, Direct Teach
29	Provide security to the field level.

30	Ability to manage security from multiple levels—i.e. campus wide or with an academic department.
31	Provide electronic signature on all submissions: • From end-user to accounting technician. • From accounting technician to academic department chairman • From academic department chairman to Comptroller.
32	Comply with Privacy Act.
33	Use electronic reporting with drill down query capability.
34	Comply with DoD civilian pay manual.
35	Provide interface to Human Resource Services to ensure no changes to civilian pay and grade data without the authorization from the SF50B.

APPENDIX C. ACCOUNTING APPLICATION FORMS AND SOURCE CODE

```
File Name: DBooks32.dpr
{*************************************************************************
 D-Books 0.1 Prototype; Naval Postgraduate School          Begun 07/11/96
 Written by LT Rob Cameron and CPT Ken Carrick

 Module:  Program file
   Notes:  This program requires the following non-standard
           Delphi 2.0 components:
              TfslformSizeLimit,
              TCalendarDialog,
              TISGMapi,
              TSendKey.
*************************************************************************}
program DBooks32;

uses
 Forms,
 main in 'main.pas' {frmMain},
 ChildEmployee in 'ChildEmployee.pas' {frmEmployee},
 About in 'About.pas' {frmAboutBox},
 dmdDbooks in 'dmdDbooks.pas' {dmlDBooks: TDataModule},
 dmdProjects in 'dmdProjects.pas' {dmlProjects: TDataModule},
 dmdTimecards in 'dmdTimecards.pas' {dmlTimecards: TDataModule},
 dmdJobOrder in 'dmdJobOrder.pas' {dmlJobOrders: TDataModule},
 dmdContracts in 'dmdContracts.pas' {dmlContracts: TDataModule},
 dmdOPTAR in 'dmdOPTAR.pas' {dmlOPTAR: TDataModule},
 dmdTravel in 'dmdTravel.pas' {dmlTravel: TDataModule},
 dmdEmployees in 'dmdEmployees.pas' {dmlEmployees: TDataModule},
 login in 'login.pas' {frmLogin},
 dmdFund in 'dmdFund.pas' {dmlFund: TDataModule},
 dmdDepartment in 'dmdDepartment.pas' {dmlDepartments: TDataModule},
 ChildTimeCard in 'ChildTimeCard.pas' {frmTimeCard},
 dmdEmpContactAndLocation in 'dmdEmpContactAndLocation.pas'
{dmlEMPContactAndLocation: TDataModule},
 ChildDepartment in 'ChildDepartment.pas' {frmDepartment},
 ChildEmpPayHist in 'ChildEmpPayHist.pas' {frmEmpPayHist},
 dmdEmpPayHist in 'dmdEmpPayHist.pas' {dmlEmpPayHist: TDataModule},
 Splash in 'Splash.pas' {frmSplash},
 qryThread in 'qryThread.pas';

{$R *.RES}

begin
```

```
  Application.Title := 'Dbooks32';
  Application.CreateForm(TfrmMain, frmMain);
  Application.CreateForm(TdmlDBooks, dmlDBooks);
  Application.CreateForm(TdmlProjects, dmlProjects);
  Application.CreateForm(TdmlTimecards, dmlTimecards);
  Application.CreateForm(TdmlJobOrders, dmlJobOrders);
  Application.CreateForm(TdmlContracts, dmlContracts);
  Application.CreateForm(TdmlOPTAR, dmlOPTAR);
  Application.CreateForm(TdmlTravel, dmlTravel);
  Application.CreateForm(TdmlEmpPayHist, dmlEmpPayHist);
  Application.CreateForm(TdmlEmployees, dmlEmployees);
  Application.CreateForm(TdmlFund, dmlFund);
  Application.CreateForm(TdmlDepartments, dmlDepartments);
  Application.CreateForm(TdmlEMPContactAndLocation,
dmlEMPContactAndLocation);
  Application.Run ;
end.
```




```
File Name: main.pas
{*************************************************************************
 D-Books 0.1 Prototype; Naval Postgraduate School       Begun 07/11/96
 Written by LT Rob Cameron and CPT Ken Carrick


 Module:  Main MDI Frame Form
  Notes:  Child form menus merge into the main form menu.
        Defaults for quarter screen form:  height := 251 width := 388
*************************************************************************}
unit Main;

interface

uses Windows, SysUtils, Classes, Graphics, Forms, Controls, Menus,
 StdCtrls, Dialogs, Buttons, Messages, ExtCtrls, ComCtrls, SendKey,
 Frmszlmt, Quickrep;

const
 strVERSION_NUMBER = '0.5 Prototype' ;
 strBUILD_NUMBER   = '3' ;

{Delcare an enumerated type used to indicate which type of child form is
being created or manipulated. It is delcared here for visibility
reasons.}
```

```
type TEnumChildFrm = (enumBudgetPage, enumTimecard,enumJobOrder,
enumContract, enumOPTAR,enumTravel,enumEmployee ,enumDepartment,
enumEmpPayHist, enumFund ) ;

type
 TfrmMain = class(TForm)
   opdMain: TOpenDialog;
   SpeedPanel: TPanel;
   spbCut: TSpeedButton;
   spbCopy: TSpeedButton;
   spbPaste: TSpeedButton;
   spbExit: TSpeedButton;
   StatusBar: TStatusBar;
   mnuDBooks: TMainMenu;
   mniSystemAs: TMenuItem;
   mniSLoginAl: TMenuItem;
   mniSLogoutAo: TMenuItem;
   N1: TMenuItem;
   mniSExitAx: TMenuItem;
   mniNavigateAn: TMenuItem;
   mniVBudgetPageAb: TMenuItem;
   mniVTimecardsAt: TMenuItem;
   mniVJobOrdersAj: TMenuItem;
   mniVContractsAc: TMenuItem;
   mniVOPTARAo: TMenuItem;
   mniVTravelAr: TMenuItem;
   mniVEmployeesAe: TMenuItem;
   mniVDepartmentsAd: TMenuItem;
   mniEditAe: TMenuItem;
   CutItem: TMenuItem;
   CopyItem: TMenuItem;
   PasteItem: TMenuItem;
   mniReportsAr: TMenuItem;
   mniWIndowAw: TMenuItem;
   mniWCascadeAc: TMenuItem;
   mniWTileAt: TMenuItem;
   mniWArrangeIconsAa: TMenuItem;
   mniWMinimizeAllAm: TMenuItem;
   mniHelpAh: TMenuItem;
   mniHAboutAa: TMenuItem;
   skyDBooks32: TSendKey;
   fslFormMain: TfslFormSizeLimit;
   mniVOptionsAi: TMenuItem;
   mniDividerV: TMenuItem;
   mniVFundsAf: TMenuItem;
```

```
    mniRProjectLaborAp: TMenuItem;
    mniREmployeeListAe: TMenuItem;
    procedure FormCreate(Sender: TObject);
    procedure mniCascadeAcClick(Sender: TObject);
    procedure UpdateMenuItems(Sender: TObject);
    procedure mniTileAtClick(Sender: TObject);
    procedure mniArrangeIconsAaClick(Sender: TObject);
    procedure mniExitAxClick(Sender: TObject);
    procedure CutItemClick(Sender: TObject);
    procedure CopyItemClick(Sender: TObject);
    procedure PasteItemClick(Sender: TObject);
    procedure mniMinimizeAllAmClick(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure mniSLoginAlClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure mniVDepartmentsAdClick(Sender: TObject);
    procedure mniVEmployeesAeClick(Sender: TObject);
    procedure mniVTravelArClick(Sender: TObject);
    procedure mniVOPTARAoClick(Sender: TObject);
    procedure mniVContractsAcClick(Sender: TObject);
    procedure mniVJobOrdersAjClick(Sender: TObject);
    procedure mniVTimecardsAtClick(Sender: TObject);
    procedure mniSLogoutAoClick(Sender: TObject);
    procedure mniVBudgetPageAbClick(Sender: TObject);
    procedure mniVFundsAfClick(Sender: TObject);
    procedure mniHAboutAaClick(Sender: TObject);
    procedure mniSExitAxClick(Sender: TObject);
    procedure mniREmployeeListAeClick(Sender: TObject);
    procedure mniRProjectLaborApClick(Sender: TObject);
  private
    { Private declarations }
    aReport : TQuickReport ;
    procedure ShowHint(Sender: TObject);
    function OKToCreate(const DesiredChild : TEnumChildFrm ) : boolean ;
  public
    { Public declarations }
    procedure CreateMDIChild(ChildType : TEnumChildFrm);
    function GetMDIChild(const DesiredChild : TEnumChildFrm) : TForm ;
  end;

var
  frmMain: TfrmMain;

implementation
```

```
{$R *.DFM}

uses About, dmdDbooks, ChildEmployee, ChildDepartment, login,
 dmdEmpPayHist, ChildEmpPayHist, Splash, ChildFund, ChildTimecard,
 ChildJobOrder, ChildBudgetPage, ReportEmployeeList, dmdReports,
 ReportProjectDetails;

{Returns the desired form, if it is showing }
function TfrmMain.GetMDIChild(const DesiredChild : TEnumChildfrm) :
TForm ;
var
 i : integer ;
begin
 result := nil ;
 case DesiredChild of

   enumBudgetPage:begin
              for i := 0 to MDIChildCount - 1 do
              begin {walk MDI children array }
               if MDIChildren[i] is
               TfrmBudgetPage  then {if one exists then }
               begin  {make the current one active}
                Result := MDIChildren[i] ;
               end ;
              end ;
             end ;
  enumTimecard :  begin
              for i := 0 to MDIChildCount - 1 do
              begin
               if MDIChildren[i] is TfrmTimecard then
               begin
                Result := MDIChildren[i] ;
               end ;
              end ;
             end ;
  enumJobOrder : begin
              for i := 0 to MDIChildCount - 1 do
              begin
               if MDIChildren[i] is TfrmJobOrder then
               begin
                Result := MDIChildren[i] ;
               end ;
              end ;
             end ;
  enumContract : begin
```

```
            (* for i := 0 to MDIChildCount - 1 do
             begin
              if MDIChildren[i] is TfrmContract then
              begin
               Result := MDIChildren[i] ;
              end ;
             end ; *)
            end ;
enumOPTAR    : begin
            (* for i := 0 to MDIChildCount - 1 do
             begin
              if MDIChildren[i] is TfrmOPTAR then
              begin
               Result := MDIChildren[i] ;
              end ;
             end ; *)
            end ;
enumTravel   : begin
            (* for i := 0 to MDIChildCount - 1 do
             begin
              if MDIChildren[i] is TfrmTravel then
              begin
               Result := MDIChildren[i] ;
              end ;
             end ; *)
            end ;
enumEmployee : begin
             for i := 0 to MDIChildCount - 1 do
             begin
              if MDIChildren[i] is TfrmEmployee then
              begin
               Result := MDIChildren[i] ;
              end ;
             end ;
            end ;
enumDepartment: begin
             for i := 0 to MDIChildCount - 1 do
             begin
              if MDIChildren[i] is TfrmDepartment then
              begin
               Result := MDIChildren[i] ;
              end ;
             end ;
            end ;
enumEmpPayHist : begin
```

```
                for i := 0 to MDIChildCount - 1 do
                begin
                 if MDIChildren[i] is TfrmEmpPayHist then
                 begin
                  Result := MDIChildren[i] ;
                 end ;
                end ;
               end ;
  enumFund :     begin
                for i := 0 to MDIChildCount - 1 do
                begin
                 if MDIChildren[i] is TfrmFund then
                 begin
                  Result := MDIChildren[i] ;
                 end ;
                end ;
               end ;
 end ;
end ;

function TfrmMain.OkToCreate(const DesiredChild : TEnumChildFrm) :
Boolean ;
var
 i : integer ;
begin
 Result := true ;  {default variable name for the return value}
 case DesiredChild of  {set to true for default}

  enumBudgetPage : begin
                for i := 0 to MDIChildCount - 1 do
                begin {walk MDI children array }
                 if MDIChildren[i] is TfrmBudgetPage then
                 begin {if one exists then }
                  Result := false ;  {return false }
                  MDIChildren[i].BringToFront ;
                 end ;
                end ;
               end ;
  enumTimecard :  begin
                for i := 0 to MDIChildCount - 1 do
                begin
                 if MDIChildren[i] is TfrmTimecard then
                 begin
                  Result := false ;
                  MDIChildren[i].BringToFront ;
```

```
            end ;
          end ;
        end ;
enumJobOrder : begin
          for i := 0 to MDIChildCount - 1 do
          begin
            if MDIChildren[i] is TfrmJobOrder then
            begin
              Result := false ;
              MDIChildren[i].BringToFront ;
            end ;
          end ;
        end ;
enumContract : begin
          (* for i := 0 to MDIChildCount - 1 do
          begin
            if MDIChildren[i] is TfrmContract then
            begin
              Result := false ;
              MDIChildren[i].BringToFront ;
            end ;
          end ;  *)
        end ;
enumOPTAR    : begin
          (* for i := 0 to MDIChildCount - 1 do
          begin
            if MDIChildren[i] is TfrmOPTAR then
            begin
              Result := false ;
              MDIChildren[i].BringToFront ;
            end ;
          end ; *)
        end ;
enumTravel   : begin
          (* for i := 0 to MDIChildCount - 1 do
          begin
            if MDIChildren[i] is TfrmContract then
            begin
              Result := false ;
              MDIChildren[i].BringToFront ;
            end ;
          end ; *)
        end ;
enumEmployee : begin
          for i := 0 to MDIChildCount - 1 do
```

```
            begin
              if MDIChildren[i] is TfrmEmployee then
              begin
                Result := false ;
                MDIChildren[i].BringToFront ;
              end ;
            end ;
          end ;
  enumDepartment : begin
            for i := 0 to MDIChildCount - 1 do
            begin
              if MDIChildren[i] is TfrmDepartment then
              begin
                Result := false ;
                MDIChildren[i].BringToFront ;
              end ;
            end ;
          end ;
  enumEmpPayHist : begin
            for i := 0 to MDIChildCount - 1 do
            begin
              if MDIChildren[i] is TfrmEmpPayHist then
              begin
                Result := false ;
                MDIChildren[i].BringToFront ;
              end ;
            end ;
          end ;
  enumFund :     begin
            for i := 0 to MDIChildCount - 1 do
            begin
              if MDIChildren[i] is TfrmFund then
              begin
                Result := false ;
                MDIChildren[i].BringToFront ;
              end ;
            end ;
          end ;
  end ;
end ;


procedure TfrmMain.FormCreate(Sender: TObject);
begin
  Application.OnHint := ShowHint;
```

```
  (*Screen.OnActiveFormChange := UpdateMenuItems;*)
  ShortDateFormat := 'mm/dd/yyyy' ;
  Height := 599 ;
  Width := 772 ;
end;

procedure TfrmMain.ShowHint(Sender: TObject);
begin
  StatusBar.SimpleText := Application.Hint;
end;

procedure TfrmMain.CreateMDIChild(ChildType : TEnumChildFrm);
  { create a new MDI child window }
var
  ChildEmployee : TfrmEmployee ;
  ChildDepartment : TfrmDepartment ;
  ChildEmpPayHist : TfrmEmpPayHist ;
  ChildTimecard : TfrmTimecard ;
  ChildFund : TfrmFund ;
  ChildJobOrder : TfrmJobOrder ;
  ChildBudgetPage : TfrmBudgetPage ;
begin
  { create a new MDI child window }
  case ChildType of

    enumBudgetPage : begin
                if OKToCreate(ChildType) then
                begin
                  try
                    ChildBudgetPage:=
                    TfrmBudgetPage.Create(Application);
                  except
                    on ExceptionRaised : Exception do
                    begin
                      messagedlg('Error'+''+
                      ExceptionRaised.Message+
                      '',mtError,[mbOK],0);
                    end ;
                  end;
                end ;
               end ;
    enumTimecard : begin
                if OKToCreate(ChildType) then
                begin
                  try
```

```
            ChildTimecard :=
            TfrmTimecard.Create(Application);
           except
            on ExceptionRaised : Exception do
            begin
             messagedlg('Error'+''+
             ExceptionRaised.Message+'',mtError
             ,[mbOK],0);
            end ;
           end;
          end ;
         end ;
enumJobOrder : begin
          if OKToCreate(ChildType) then
          begin
           try
            ChildJobOrder :=
            TfrmJobOrder.Create(Application);
           except
            on ExceptionRaised : Exception do
            begin
             messagedlg('Error'+''+
             ExceptionRaised.Message+'',mtError
             ,[mbOK],0);
            end ;
           end;
          end ;
         end ;
enumContract : begin
          (* if OKToCreate(ChildType) then
           begin
           try
            ChildContract :=
            TfrmContract.Create(Application) ;
           except
            on ExceptionRaised : Exception do
            begin
             messagedlg('Error'+''+
             ExceptionRaised.Message+'',mtError
             ,[mbOK],0);
            end ;
           end;
          end ;  *)
         end ;
enumOPTAR    : begin
```

```
          (* if OKToCreate(ChildType) then
           begin
            try
             ChildOPTAR  := TfrmOPTAR.Create(Application) ;
            except
             on ExceptionRaised : Exception do
             begin
              messagedlg('Error'+''+
              ExceptionRaised.Message+'',mtError
              ,[mbOK],0);
             end ;
            end;
           end ;  *)
          end ;
enumTravel   : begin
          (* if OKToCreate(ChildType) then
           begin
            try
             ChildTravel  := TfrmTravel.Create(Application) ;
            except
             on ExceptionRaised : Exception do
             begin
              messagedlg('Error'+''+
              ExceptionRaised.Message+'',mtError
              ,[mbOK],0);
             end ;
            end;
           end ;  *)
          end ;
 enumEmployee : begin
           if OKToCreate(ChildType) then
           begin
            try
             ChildEmployee  :=
             TfrmEmployee.Create(Application) ;
            except
             on ExceptionRaised : Exception do
             begin
              messagedlg('Unable to Create Employee form.'+
              'Error: '+''+ExceptionRaised.Message+'',
              mtError,[mbOK],0);
             end ;
            end;
           end ;
          end ;
```

```
enumDepartment : begin
          if OKToCreate(ChildType) then
          begin
           try
            ChildDepartment  :=
            TfrmDepartment.Create(Application) ;
           except
            on ExceptionRaised : Exception do
            begin
             messagedlg('Unable to Create Employee form. '+
             ' Error: '+''''+ExceptionRaised.Message+'''',
             mtError,[mbOK],0);
            end ;
           end;
          end ;
         end ;
enumEmpPayHist : begin
          if OKToCreate(ChildType) then
          begin
           try
            ChildEmpPayHist :=
            TfrmEmpPayHist.Create(Application) ;
           except
            on ExceptionRaised : Exception do
            begin
             messagedlg('Unable to Create Employee form.'+
             'Error: '+''''+ExceptionRaised.Message+'''',
             mtError,[mbOK],0);
            end ;
           end;
          end ;
         end ;
enumFund :  begin
          if OKToCreate(ChildType) then
          begin
           try
            ChildFund := TfrmFund.Create(Application) ;
           except
            on ExceptionRaised : Exception do
            begin
             messagedlg('Unable to Create Employee form.'+
             'Error: '+''''+ExceptionRaised.Message+'''',
             mtError,[mbOK],0);
            end ;
           end;
```

```
            end ;
          end ;
 end ;
end;

procedure TfrmMain.mniExitAxClick(Sender: TObject);
begin
  Close;
end;

procedure TfrmMain.CutItemClick(Sender: TObject);
begin
  {cut selection to clipboard}
end;

procedure TfrmMain.CopyItemClick(Sender: TObject);
begin
  {copy selection to clipboard}
end;

procedure TfrmMain.PasteItemClick(Sender: TObject);
begin
  {paste from clipboard}
end;

procedure TfrmMain.mniCascadeAcClick(Sender: TObject);
begin
  Cascade;
end;

procedure TfrmMain.mniTileAtClick(Sender: TObject);
begin
  Tile;
end;

procedure TfrmMain.mniArrangeIconsAaClick(Sender: TObject);
begin
  ArrangeIcons;
end;

procedure TfrmMain.mniMinimizeAllAmClick(Sender: TObject);
var
  I: Integer;
begin
  { Must be done backwards through the MDIChildren array }
```

```
 for I := MDIChildCount - 1 downto 0 do
   MDIChildren[I].WindowState := wsMinimized;
end;

procedure TfrmMain.UpdateMenuItems(Sender: TObject);
begin
;
end;

procedure TfrmMain.FormDestroy(Sender: TObject);
begin
 Screen.OnActiveFormChange := nil;
end;

procedure TfrmMain.mniSLoginAlClick(Sender: TObject);
begin
 with dmlDbooks do
  try                {Connect to database, triggers password box}
   dbsDBooks.Connected := true ;
   if dbsDBooks.Connected then  {Ensure proper connection exists}
    OpenDBooksDM ;  {Open the lookup tables in the datamodel.
                This is why it takes some time to log in.}
  except
   on ExceptionRaised : Exception do
    messagedlg('Unable to Open database. Error: '+'"'+
    ExceptionRaised.Message+'"',mtError,[mbOK],0);
  end ;
end;

procedure TfrmMain.FormShow(Sender: TObject);
var
 wrdModalResult : word ;
begin
 try
  frmLogin := TfrmLogin.Create(Application) ;
  wrdModalResult := frmLogin.ShowModal  ;
  if wrdModalResult = mrOK then
  begin
   try
    frmSplash := TfrmSplash.Create(Application) ;
    try
     frmSplash.Show ;
     frmSPlash.Update ;
     with dmlDBooks do
     begin
```

```
        dbsDbooks.Connected := false ;
        dbsDBooks.params.Clear ;
        dbsDBooks.Params.Add('USER NAME='+
        frmLogin.edtUserName.Text) ;
        dbsDBooks.Params.Add('PASSWORD='+
        frmLogin.edtPassword.Text) ;
        OpenDBooksDM ;
       end ;
      except
       on E : exception do
        if E is EDatabaseError then
         messagedlg('Error Connecting to database. Please contanct'+
         ' the DBA.'+#13+'ERROR: '+E.message,mtError,[mbOK],0) ;
        else
         messagedlg('Error creating form.  Please contact '+
         'technical support.',mtError,[mbOK],0) ;
      end ;
     finally
      frmSplash.Free ;
     end ;
    end
    else
    begin
     if wrdModalResult = mrAbort then
      messagedlg('Logon failure.  Please contact the DBA.',
      mtError, [mbOK],0);
      Close ;
    end ;
   finally
    frmLogin.Free ;
   end ;
end;

procedure TfrmMain.mniVDepartmentsAdClick(Sender: TObject);
var
 DesiredChildFrm : TEnumChildFrm ;    {User defined enum type }
begin
if dmlDbooks.dbsDbooks.Connected then
 begin
  DesiredChildFrm := enumDepartment ;
  try
   CreateMDIChild(DesiredChildFrm);
  except
   on ExceptionRaised : Exception do
   begin
```

```
      messagedlg('Error Could not create Department Form'+''+
      ExceptionRaised.Message+'',mtError,[mbOK],0);
    end ;
  end ;
 end
 else
 begin
  If messagedlg('Not Logged in Database--Click on "OK" to Login',
  mtConfirmation,[mbOK,mbCancel],0) = mrOK then
  begin
   mniSLoginAlClick(Sender);
   mniVDepartmentsAdClick(Sender);
  end;
 end ;
end;

procedure TfrmMain.mniVEmployeesAeClick(Sender: TObject);
var
 DesiredChildFrm : TEnumChildFrm ;    {User defined enum type }
begin
if dmlDbooks.dbsDbooks.Connected then
 begin
  DesiredChildFrm := enumEmployee ;
  try
   CreateMDIChild(DesiredChildFrm);
  except
   on ExceptionRaised : Exception do
   begin
    messagedlg('Error Could not create Employee Form'+''+
    ExceptionRaised.Message+'',mtError,[mbOK],0);
   end ;
  end ;
 end
 else
 begin
  If messagedlg('Not Logged in Database--Click on "OK" to Login',
  mtConfirmation,[mbOK,mbCancel],0) = mrOK then
  begin
   mniSLoginAlClick(Sender);
   mniVEmployeesAeClick(Sender);
  end;
 end ;

end;
```

```
procedure TfrmMain.mniVTravelArClick(Sender: TObject);
var
 DesiredChildFrm : TEnumChildFrm ;    {User defined enum type }
begin
if dmlDbooks.dbsDbooks.Connected then
 begin
  DesiredChildFrm := enumTravel ;
  try
   CreateMDIChild(DesiredChildFrm);
  except
   on ExceptionRaised : Exception do
   begin
    messagedlg('Error Could not create Travel Form'+''+
    ExceptionRaised.Message+'',mtError,[mbOK],0);
   end ;
  end ;
 end
 else
 begin
  If messagedlg('Not Logged in Database--Click on "OK" to Login',
  mtConfirmation,[mbOK,mbCancel],0) = mrOK then
  begin
   mniSLoginAlClick(Sender);
   mniVTravelArClick(Sender);
  end;
 end ;

end;

procedure TfrmMain.mniVOPTARAoClick(Sender: TObject);
var
 DesiredChildFrm : TEnumChildFrm ;    {User defined enum type }
begin
if dmlDbooks.dbsDbooks.Connected then
 begin
  DesiredChildFrm := enumOPTAR ;
  try
   CreateMDIChild(DesiredChildFrm);
  except
   on ExceptionRaised : Exception do
   begin
    messagedlg('Error Could not create OPTAR Form'+''+
    ExceptionRaised.Message+'',mtError,[mbOK],0);
   end ;
  end ;
```

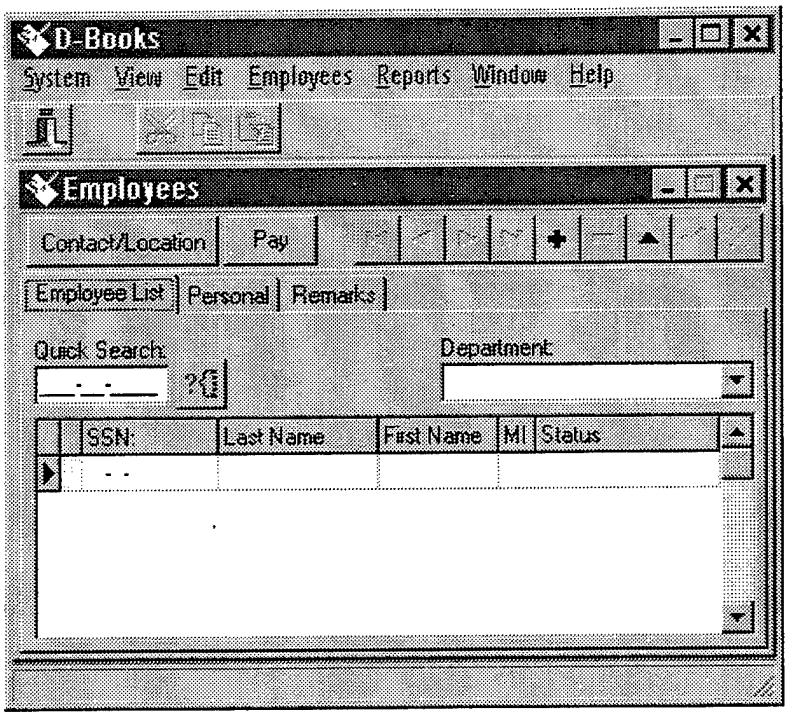
```
 end
 else
 begin
  If messagedlg('Not Logged in Database--Click on "OK" to Login',
  mtConfirmation,[mbOK,mbCancel],0) = mrOK then
  begin
   mniSLoginAlClick(Sender);
   mniVOPTARAoClick(Sender);
  end;
 end ;

end;

procedure TfrmMain.mniVContractsAcClick(Sender: TObject);
var
 DesiredChildFrm : TEnumChildFrm ;    {User defined enum type }
begin
if dmlDbooks.dbsDbooks.Connected then
 begin
  DesiredChildFrm := enumContract ;
  try
   CreateMDIChild(DesiredChildFrm);
  except
   on ExceptionRaised : Exception do
   begin
    messagedlg('Error Could not create Contracts Form'+''''+
    ExceptionRaised.Message+'''',mtError,[mbOK],0);
   end ;
  end ;
 end
 else
 begin
  If messagedlg('Not Logged in Database--Click on "OK" to Login',
  mtConfirmation,[mbOK,mbCancel],0) = mrOK then
  begin
   mniSLoginAlClick(Sender);
   mniVContractsAcClick(Sender);
  end;
 end ;

end;

procedure TfrmMain.mniVJobOrdersAjClick(Sender: TObject);
var
 DesiredChildFrm : TEnumChildFrm ;    {User defined enum type }
```

```
begin
if dmlDbooks.dbsDbooks.Connected then
 begin
  DesiredChildFrm := enumJobOrder ;
  try
   CreateMDIChild(DesiredChildFrm);
  except
   on ExceptionRaised : Exception do
   begin
    messagedlg('Error Could not create Job Order Form'+''+
    ExceptionRaised.Message+'',mtError,[mbOK],0);
   end ;
  end ;
 end
 else
 begin
  If messagedlg('Not Logged in Database--Click on "OK" to Login',
  mtConfirmation,[mbOK,mbCancel],0) = mrOK then
  begin
   mniSLoginAlClick(Sender);
   mniVJobOrdersAjClick(Sender);
  end;
 end ;

end;

procedure TfrmMain.mniVTimecardsAtClick(Sender: TObject);
var
 DesiredChildFrm : TEnumChildFrm ;   {User defined enum type }
begin
if dmlDbooks.dbsDbooks.Connected then
 begin
  DesiredChildFrm := enumTimecard ;
  try
   CreateMDIChild(DesiredChildFrm);
  except
   on ExceptionRaised : Exception do
   begin
    messagedlg('Error Could not create Timecard Form'+''+
    ExceptionRaised.Message+'',mtError,[mbOK],0);
   end ;
  end ;
 end
 else
 begin
```

```
   If messagedlg('Not Logged in Database--Click on "OK" to Login',
   mtConfirmation,[mbOK,mbCancel],0) = mrOK then
   begin
    mniSLoginAlClick(Sender);
    mniVTimecardsAtClick(Sender);
   end;
  end ;

end;

procedure TfrmMain.mniSLogoutAoClick(Sender: TObject);
begin
 try
  dmlDbooks.dbsDbooks.Connected := false ;
 except
  on ExceptionRaised : Exception do
   messagedlg('Database Error: '+''''+ExceptionRaised.Message+'''',
   mtError,[mbOK],0);
 end ;
end;

procedure TfrmMain.mniVBudgetPageAbClick(Sender: TObject);
var
 DesiredChildFrm : TEnumChildFrm ;    {User defined enum type }
begin
if dmlDbooks.dbsDbooks.Connected then
 begin
  DesiredChildFrm := enumBudgetPage ;
  try
   CreateMDIChild(DesiredChildFrm);
  except
   on ExceptionRaised : Exception do
   begin
    messagedlg('Error Could not create Budget Pages Form'+''''+
    ExceptionRaised.Message+'''',mtError,[mbOK],0);
   end ;
  end ;
 end
 else
 begin
  If messagedlg('Not Logged in Database--Click on "OK" to Login',
  mtConfirmation,[mbOK,mbCancel],0) = mrOK then
  begin
   mniSLoginAlClick(Sender);
   mniVBudgetPageAbClick(Sender);
```

```
  end;
 end ;

end;

procedure TfrmMain.mniVFundsAfClick(Sender: TObject);
var
 DesiredChildFrm : TEnumChildFrm ;    {User defined enum type }
begin
if dmlDbooks.dbsDbooks.Connected then
 begin
  DesiredChildFrm := enumFund ;
  try
   CreateMDIChild(DesiredChildFrm);
  except
   on ExceptionRaised : Exception do
   begin
    messagedlg('Error Could not create Fund Form'+""+
    ExceptionRaised.Message+"",mtError,[mbOK],0);
   end ;
  end ;
 end
 else
 begin
  If messagedlg('Not Logged in Database--Click on "OK" to Login',
  mtConfirmation,[mbOK,mbCancel],0) = mrOK then
  begin
   mniSLoginAlClick(Sender);
   mniVFundsAfClick(Sender);
  end;
 end ;
end;

procedure TfrmMain.mniHAboutAaClick(Sender: TObject);
begin
 try
  frmAboutBox := TfrmAboutBox.Create(Application) ;
  try
   frmAboutBox.ShowModal;
  finally
   frmAboutBox.Free ;
  end ;
 except
  on ExceptionRaised : Exception do
  begin
```

```
    messagedlg('Error'+''+ExceptionRaised.Message+'',
    mtError,[mbOK],0);
   end ;
  end;
end;

procedure TfrmMain.mniSExitAxClick(Sender: TObject);
begin
  Close ;
end;

procedure TfrmMain.mniREmployeeListAeClick(Sender: TObject);
begin
  try
    frmEmpListing := TfrmEmpListing.Create(Self) ;
    aReport := frmEmpListing.Report ;
    aReport.Preview ;
  finally
    frmEmpListing.Free ;
  end ;
end;

procedure TfrmMain.mniRProjectLaborApClick(Sender: TObject);
begin
  try
    frmProjectDetails := TfrmProjectDetails.Create(Self) ;
    aReport := frmProjectDetails.Report ;
    aReport.Preview ;
  finally
    frmProjectDetails.Free ;
  end ;
end;

end.
```



File Name: ChildEmployee.pas

```
{*************************************************************************
 D-Books 0.1 Prototype; Naval Postgraduate School          Begun 07/11/96
 Written by Rob Cameron and Ken Carrick

Module: Employees
 Notes:
*************************************************************************}
unit ChildEmployee;

interface

uses Windows, Classes, Graphics, Forms, Controls, Frmszlmt, Buttons, SysUtils,
 ExtCtrls, ComCtrls, StdCtrls, Mask, Grids, DBGrids, DBCtrls, DB, DBTables,
 SendKey, Menus, Calndar, Dialogs;

type
 TfrmEmployee = class(TForm)
  pgcEmployee: TPageControl;
  pnlDepartment: TPanel;
  fslEmployee: TfslFormSizeLimit;
  tblEmpList: TTabSheet;
  tbsEmpData: TTabSheet;
  dgrEmployees: TDBGrid;
  lblQuicjSearch: TLabel;
  medSSN: TMaskEdit;
```

lblDept: TLabel;
dlcDepartment: TDBLookupComboBox;
lblTitle: TLabel;
dlcTitle: TDBLookupComboBox;
tbsRemarks: TTabSheet;
dmmRemarks: TDBMemo;
lblEmpType: TLabel;
dlcEmpType: TDBLookupComboBox;
lblService: TLabel;
dlcService: TDBLookupComboBox;
dlcTenure: TDBLookupComboBox;
lblTenure: TLabel;
lblDateOb: TLabel;
dedDateOnboard: TDBEdit;
lblDateTerminated: TLabel;
dlcDateTerm: TDBEdit;
bbnContact: TBitBtn;
spbJump: TSpeedButton;
lblRemakrs: TLabel;
btnPayHist: TButton;
dngEmployee: TDBNavigator;
lblSSN: TLabel;
dedSSN: TDBEdit;
lblStatus: TLabel;
dlStatus: TDBLookupComboBox;
lblLastName: TLabel;
dedLastName: TDBEdit;
lblFirstName: TLabel;
dedFirstName: TDBEdit;
lblMI: TLabel;
dedMI: TDBEdit;
cldEmployee: TCalendarDialog;
spbDateOnboard: TSpeedButton;
spbDateTerminated: TSpeedButton;
mnuEmployees: TMainMenu;
mniEmployeesAe: TMenuItem;
mniEInsertAi: TMenuItem;
mniEDeleteAd: TMenuItem;
mniECanceRecAc: TMenuItem;
mniEApplyAa: TMenuItem;
mniECancelAllAn: TMenuItem;
mniEShowAs: TMenuItem;
mniESStandardAs: TMenuItem;
mniESDeletedAd: TMenuItem;
mniESModifiedAm: TMenuItem;

```
  mniESInsertedAi: TMenuItem;
  mniSeperator: TMenuItem;
  mniECloseAl: TMenuItem;
  procedure FormClose(Sender: TObject; var Action: TCloseAction);
  procedure FormShow(Sender: TObject);
  procedure spbJumpClick(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  procedure btnPayHistClick(Sender: TObject);
  procedure dmmRemarksDblClick(Sender: TObject);
  procedure SendKeyClick(Sender : TObject) ;
  procedure dngEmployeeClick(Sender: TObject; Button: TNavigateBtn);
  procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
  procedure spbDateOnboardClick(Sender: TObject);
  procedure spbDateTerminatedClick(Sender: TObject);
  procedure mniESInsertedAiClick(Sender: TObject);
  procedure mniESModifiedAmClick(Sender: TObject);
  procedure mniESDeletedAdClick(Sender: TObject);
  procedure mniESStandardAsClick(Sender: TObject);
  procedure mniECancelAllAnClick(Sender: TObject);
  procedure mniECanceRecAcClick(Sender: TObject);
  procedure mniEApplyAaClick(Sender: TObject);
  procedure mniEDeleteAdClick(Sender: TObject);
  procedure mniEInsertAiClick(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

implementation

uses dmdEmployees, dmdDepartment, dmdEmpPayHist, main, dmdDbooks;

{$R *.DFM}

procedure TfrmEmployee.SendKeyClick(Sender : TObject) ;
begin
 if sender is TDBEdit then
  with dmlEmployees do
   with Sender as TDBEdit do
    if qryEmployee.FieldByName(DataField).AsString = '' then
     frmMain.skyDBooks32.SendKeys ('{HOME}');
 if sender is TMaskEdit then
  with Sender as TMaskEdit do
   if Text = '' then
```

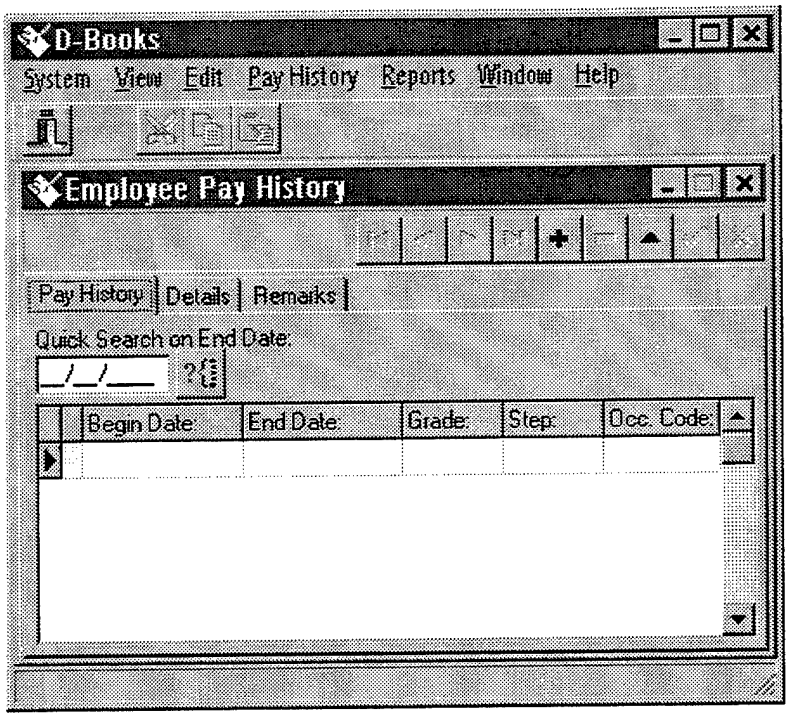
```
    frmMain.skyDBooks32.SendKeys ('{HOME}');
end ;

procedure TfrmEmployee.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 Action := caFree;
end;

procedure TfrmEmployee.FormShow(Sender: TObject);
begin
 Left := 0 ;
 Top := 0 ;
end;

procedure TfrmEmployee.spbJumpClick(Sender: TObject);
begin
 with dmlEmployees, qryEmployee do
  Locate('SSN',medSSN.Text,[loCaseInsensitive]) ;
end;

procedure TfrmEmployee.FormCreate(Sender: TObject);
begin
 pgcEmployee.ActivePage := tblEmpList ;
 with dmlEmployees do
 begin
  Menu := mnuEmployees ;
 end ;
end;

procedure TfrmEmployee.btnPayHistClick(Sender: TObject);
var
 DesiredChildFrm : TEnumChildFrm  ;
begin
 with dmlEmployees, dmlEmpPayHist do
  if qryEmployeeEMPLOYEE_ID.AsString <> '' then
   qryEmpPayHist.Filter := 'EMPLOYEE_ID =
'+qryEmployeeEMPLOYEE_ID.AsString ;
  DesiredChildFrm := enumEmpPayHist ;
  frmMain.CreateMDIChild(DesiredChildFrm);

end;

procedure TfrmEmployee.dmmRemarksDblClick(Sender: TObject);
begin
 with dmlEmployees do
```

```
 begin
  if not (qryEmployee.State in [dsEdit, dsInsert]) then  {Check to see if all ready editing}
  begin
   qryEmployee.Edit ;     {If not then start an edit }
  end ;
  with Sender as TDBMemo do
  begin
   Lines.Add(DateToStr(Now) + ' - ') ;  {Add a line }
   SelStart := SelStart - 1 ;       {Reposition cursor to end of line }
  end ;
 end ;
end;

procedure TfrmEmployee.dngEmployeeClick(Sender: TObject;
 Button: TNavigateBtn);
begin
 if Button = nbInsert then
 begin
  pgcEmployee.ActivePage := tbsEmpData ;
  dedLastName.SetFocus ;
 end ;
end;

procedure TfrmEmployee.FormCloseQuery(Sender: TObject;
 var CanClose: Boolean);
begin
 mniEApplyAaClick(Self) ;
 CanClose := true ;
end;

procedure TfrmEmployee.spbDateOnboardClick(Sender: TObject);
begin
 with dmlEmployees do
 begin
  cldEmployee.Date :=        {Initialize with current Date }
   FormatDateTime('mm/dd/yyyy',qryEmployeeDATE_ONBOARD.asDateTime) ;
  if cldEmployee.Execute then
  begin
   if not (qryEmployee.State in [dsEdit,dsInsert]) then
   begin
    qryEmployee.Edit ;
   end ;
   qryEmployeeDATE_ONBOARD.AsString := cldEmployee.Date ; {Store value into
field}
  end ;
```

```
  end;
end;

procedure TfrmEmployee.spbDateTerminatedClick(Sender: TObject);
begin
  with dmlEmployees do
  begin
    cldEmployee.Date :=        {Initialize with current Date }
      FormatDateTime('mm/dd/yyyy',qryEmployeeDATE_TERMINATED.asDateTime) ;
    if cldEmployee.Execute then
    begin
      if not (qryEmployee.State in [dsEdit,dsInsert]) then
      begin
        qryEmployee.Edit ;
      end ;
      qryEmployeeDATE_TERMINATED.AsString := cldEmployee.Date ; {Store value
into field}
    end ;
  end;
end;

procedure TfrmEmployee.mniESInsertedAiClick(Sender: TObject);
begin
  with dmlEmployees, qryEmployee do
    UpdateRecordTypes := [rtInserted] ;
  with Sender as TMenuItem do Checked := true ;
  mniEmployeesAe.Items[2].Caption := '&Delete New Employee' ;
  dgrEmployees.Font.Color := clNavy ;
end;

procedure TfrmEmployee.mniESModifiedAmClick(Sender: TObject);
begin
  with dmlEmployees, qryEmployee do
    UpdateRecordTypes := [rtModified] ;
  with Sender as TMenuItem do Checked := true ;
  mniEmployeesAe.Items[2].Caption := '&Undo Changes' ;
  dgrEmployees.Font.Color := clMaroon ;
end;

procedure TfrmEmployee.mniESDeletedAdClick(Sender: TObject);
begin
  with dmlEmployees, qryEmployee do
    UpdateRecordTypes := [rtDeleted] ;
  with Sender as TMenuItem do Checked := true ;
  mniEmployeesAe.Items[2].Caption := '&Undelete Employee' ;
```

```
  dgrEmployees.Font.Color := clRed ;
end;

procedure TfrmEmployee.mniESStandardAsClick(Sender: TObject);
begin
  with dmlEMployees, qryEmployee do
    UpdateRecordTypes := [rtModified, rtInserted, rtUnmodified] ;
  mniEmployeesAe.Items[5].Items[0].Checked := true ;
  mniEmployeesAe.Items[2].Caption := '&Cancel Record Updates' ;
  dgrEmployees.Font.Color := clWindowText ;
end;

procedure TfrmEmployee.mniECancelAllAnClick(Sender: TObject);
begin
  dmlEmployees.qryEmployee.CancelUpdates ;
  mniESStandardAsClick(Sender) ; {Go back to standard view if not there.}
end;

procedure TfrmEmployee.mniECanceRecAcClick(Sender: TObject);
begin
  if messagedlg('Are you sure?',mtWarning,[mbYes,mbNo],0) = mrYes then
    dmlEmployees.qryEmployee.RevertRecord ;
end;

procedure TfrmEmployee.mniEApplyAaClick(Sender: TObject);
begin
  with dmlEmployees, qryEmployee, dmlDBooks do
  begin
    dbsDBooks.StartTransaction ;
    try
      ApplyUpdates ;
      dbsDBooks.Commit ;
    except
      dbsDBooks.Rollback ;
      raise ;
    end ;
    CommitUpdates ;
    mniESStandardAsClick(Sender) ; {Go back to standard view if not there.}
    Close ;
    Open ;
  end ;
end;

procedure TfrmEmployee.mniEDeleteAdClick(Sender: TObject);
begin
```

```
  dmlEmployees.qryEmployee.Delete ;
end;

procedure TfrmEmployee.mniEInsertAiClick(Sender: TObject);
begin
  pgcEmployee.ActivePage := tbsEmpData ;
  dedLastName.SetFocus ;
  dmlEmployees.qryEmployee.Insert ;
end;

end.
```




```
File Name:  ChildEmpPayHist.pas
{*************************************************************************
 D-Books 0.1 Prototype; Naval Postgraduate School          Begun 07/11/96
 Written by LT Rob Cameron and CPT Ken Carrick

 Module:  Employee Pay History Form
   Notes:
*************************************************************************}
unit ChildEmpPayHist;

interface

uses  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
Frmszlmt, ComCtrls, Buttons, ExtCtrls, StdCtrls, Mask, DBCtrls, Grids,  DBGrids,
SendKey, Calndar, Menus;

type
 TfrmEmpPayHist = class(TForm)
   pnlNavigator: TPanel;
   pgcEmpPayHist: TPageControl;
   fslEmpPayHist: TfslFormSizeLimit;
   tbsPayHist: TTabSheet;
   dgrEmpPayHist: TDBGrid;
   tbsDetails: TTabSheet;
   Label9: TLabel;
   dedHourlyRate: TDBEdit;
```

```
dedDailyRate: TDBEdit;
dedAnnualRate: TDBEdit;
dedOTRate: TDBEdit;
dedRegAccelRate: TDBEdit;
dedOTAccelRate: TDBEdit;
lblQuickSearch: TLabel;
medPPE: TMaskEdit;
spbJump: TSpeedButton;
cldEmpPayHist: TCalendarDialog;
dngEmpPayHist: TDBNavigator;
tbsRemarks: TTabSheet;
lblRemarks: TLabel;
dmmRemarks: TDBMemo;
dedBeginDate: TDBEdit;
dedEndDate: TDBEdit;
dlcGrade: TDBLookupComboBox;
dlcStep: TDBLookupComboBox;
dlcOCCCode: TDBLookupComboBox;
lblDailyRate: TLabel;
lblAnnualRate: TLabel;
lblOTRate: TLabel;
lblRegAccelRate: TLabel;
lblOTAccelRate: TLabel;
lblBeginDate: TLabel;
lblEndDate: TLabel;
lblGrade: TLabel;
lblStep: TLabel;
lblOCCCode: TLabel;
spbBeginDate: TSpeedButton;
spbEndDate: TSpeedButton;
mnuEmpPayHist: TMainMenu;
mniPayHistoryAp: TMenuItem;
mniPInsertAi: TMenuItem;
mniPDeleteAd: TMenuItem;
mniPCancelRecAc: TMenuItem;
mniPApplyAa: TMenuItem;
mniPCancelAn: TMenuItem;
mniPShowAs: TMenuItem;
mniPSStandardAs: TMenuItem;
mniPSDeletedAd: TMenuItem;
mniPSModifiedAm: TMenuItem;
mniPSInsertedAi: TMenuItem;
mniDivider: TMenuItem;
mniCloseEmpPayHistPAl: TMenuItem;
procedure FormClose(Sender: TObject; var Action: TCloseAction);
```

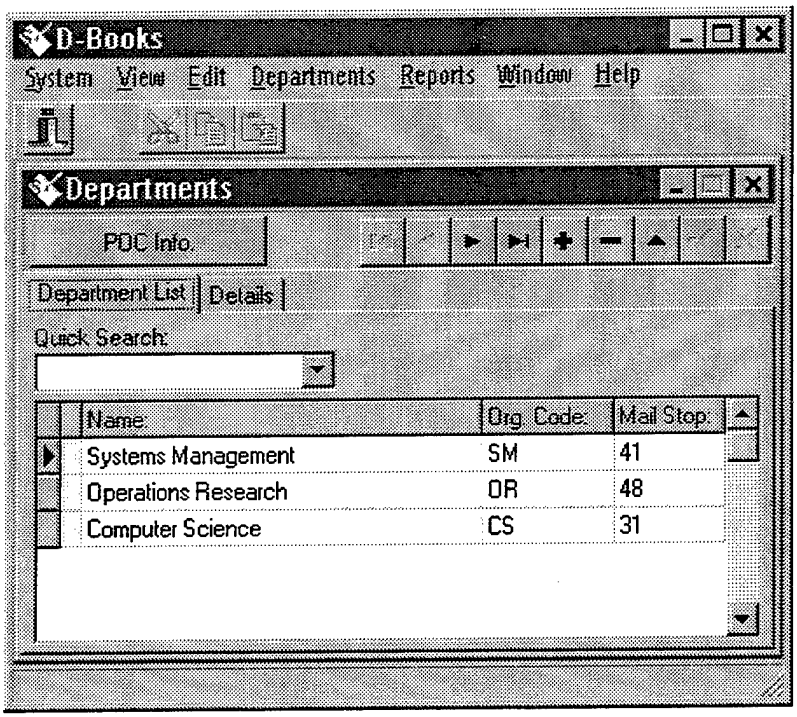
```
  procedure FormShow(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  procedure dedHourlyRateExit(Sender: TObject);
  procedure dmmRemarksDblClick(Sender: TObject);
  procedure SendKeyClick(Sender : TObject) ;
  procedure dngEmpPayHistClick(Sender: TObject; Button: TNavigateBtn);
  procedure dgrEmpPayHistEditButtonClick(Sender: TObject);
  procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
  procedure spbBeginDateClick(Sender: TObject);
  procedure spbEndDateClick(Sender: TObject);
  procedure mniPSInsertedAiClick(Sender: TObject);
  procedure mniPSDeletedAdClick(Sender: TObject);
  procedure mniPSModifiedAmClick(Sender: TObject);
  procedure mniPSStandardAsClick(Sender: TObject);
  procedure mniPCancelRecAcClick(Sender: TObject);
  procedure mniPCancelAnClick(Sender: TObject);
  procedure mniPApplyAaClick(Sender: TObject);
  procedure mniPDeleteAdClick(Sender: TObject);
  procedure mniPInsertAiClick(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

implementation

uses dmdEmpPayHist, main, dmdDbooks;
{$R *.DFM}

procedure TfrmEmpPayHist.SendKeyClick(Sender : TObject) ;
begin
 with dmlEmpPayHist do
  if qryEmpPayHist.FieldByName(TDBEdit(Sender).DataField).AsString = '' then
   frmMain.skyDBooks32.SendKeys ('{HOME}');
end ;

procedure TfrmEmpPayHist.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 Action := caFree ;
end;

procedure TfrmEmpPayHist.FormShow(Sender: TObject);
begin
 left := 0 ;
```

```
  Top := 250 ;
end;

procedure TfrmEmpPayHist.FormCreate(Sender: TObject);
begin
  pgcEmpPayHist.ActivePage := tbsPayHist ;
end;

procedure TfrmEmpPayHist.dedHourlyRateExit(Sender: TObject);
begin
  dmlEmpPayHist.CalcPayRates ;
end;

procedure TfrmEmpPayHist.dmmRemarksDblClick(Sender: TObject);
begin
  with dmlEmpPayHist do
  begin
    if not (qryEmpPayHist.State in [dsEdit, dsInsert]) then  {Check to see if all ready
editing}
    begin
      qryEmpPayHist.Edit ;     {If not then start an edit }
    end ;
    with Sender as TDBMemo do
    begin
      Lines.Add(DateToStr(Now) + ' - ') ;  {Add a line }
      SelStart := SelStart - 1 ;       {Reposition cursor to end of line }
    end ;
  end ;
end;

procedure TfrmEmpPayHist.dngEmpPayHistClick(Sender: TObject;
  Button: TNavigateBtn);
begin
  if Button = nbInsert then
  begin
    pgcEmpPayHist.ActivePage := tbsDetails ;
    dedBeginDate.SetFocus ;
  end ;
end;

procedure TfrmEmpPayHist.dgrEmpPayHistEditButtonClick(Sender: TObject);
begin
  with dmlEmpPayHist, dgrEmpPayHist do
  begin
    cldEmpPayHist.Date :=        {Initialize with current Date }
```

```
    FormatDateTime('mm/dd/yyyy',SelectedField.asDateTime) ;
   if cldEmpPayHist.Execute then
   begin
    if not (qryEmpPayHist.State in [dsEdit,dsInsert]) then
    begin
     qryEmpPayHist.Edit ;
    end ;
    SelectedField.AsString := cldEmpPayHist.Date ; {Store value into field}
   end ;
  end;
end;

procedure TfrmEmpPayHist.FormCloseQuery(Sender: TObject;
 var CanClose: Boolean);
begin
 mniPApplyAaClick(Self) ;
 CanClose := true ;
end;

procedure TfrmEmpPayHist.spbBeginDateClick(Sender: TObject);
begin
 with dmlEmpPayHist do
 begin
  cldEmpPayHist.Date :=        {Initialize with current Date }
   FormatDateTime('mm/dd/yyyy',qryEmpPayHistBEGIN_DATE.asDateTime) ;
  if cldEmpPayHist.Execute then
  begin
   if not (qryEmpPayHist.State in [dsEdit,dsInsert]) then
   begin
    qryEmpPayHist.Edit ;
   end ;
   qryEmpPayHistBEGIN_DATE.AsString := cldEmpPayHist.Date ; {Store value into
field}
  end ;
 end;
end;

procedure TfrmEmpPayHist.spbEndDateClick(Sender: TObject);
begin
 with dmlEmpPayHist do
 begin
  cldEmpPayHist.Date :=        {Initialize with current Date }
   FormatDateTime('mm/dd/yyyy',qryEmpPayHistEND_DATE.asDateTime) ;
  if cldEmpPayHist.Execute then
  begin
```

```
    if not (qryEmpPayHist.State in [dsEdit,dsInsert]) then
    begin
      qryEmpPayHist.Edit ;
    end ;
    qryEmpPayHistEND_DATE.AsString := cldEmpPayHist.Date ; {Store value into
field}
   end ;
  end;
end;

procedure TfrmEmpPayHist.mniPSInsertedAiClick(Sender: TObject);
begin
  with dmlEmpPayHist, qryEmpPayHist do
    UpdateRecordTypes := [rtInserted] ;
  with Sender as TMenuItem do Checked := true ;
  mniPayHistoryAp.Items[2].Caption := '&Delete New Pay History' ;
  dgrEmpPayHist.Font.Color := clNavy ;
end;

procedure TfrmEmpPayHist.mniPSDeletedAdClick(Sender: TObject);
begin
  with dmlEmpPayHist, qryEmpPayHist do
    UpdateRecordTypes := [rtDeleted] ;
  with Sender as TMenuItem do Checked := true ;
  mniPayHistoryAp.Items[2].Caption := '&Undelete Pay History' ;
  dgrEmpPayHist.Font.Color := clRed ;
end;

procedure TfrmEmpPayHist.mniPSModifiedAmClick(Sender: TObject);
begin
  with dmlEmpPayHist, qryEmpPayHist do
    UpdateRecordTypes := [rtModified] ;
  with Sender as TMenuItem do Checked := true ;
  mniPayHistoryAp.Items[2].Caption := '&Undo Changes' ;
  dgrEmpPayHist.Font.Color := clMaroon ;
end;

procedure TfrmEmpPayHist.mniPSStandardAsClick(Sender: TObject);
begin
  with dmlEmpPayHist, qryEmpPayHist do
    UpdateRecordTypes := [rtModified, rtInserted, rtUnmodified] ;
  mniPayHistoryAp.Items[5].Items[0].Checked := true ;
  mniPayHistoryAp.Items[2].Caption := '&Cancel Record Updates' ;
  dgrEmpPayHist.Font.Color := clWindowText ;
end;
```

```
procedure TfrmEmpPayHist.mniPCancelRecAcClick(Sender: TObject);
begin
 if messagedlg('Are you sure?',mtWarning,[mbYes,mbNo],0) = mrYes then
  dmlEmpPayHist.qryEmpPayHist.RevertRecord ;
end;

procedure TfrmEmpPayHist.mniPCancelAnClick(Sender: TObject);
begin
 dmlEmpPayHist.qryEmpPayHist.CancelUpdates ;
 mniPSStandardAsClick(Sender) ; {Go back to standard view if not there.}
end;

procedure TfrmEmpPayHist.mniPApplyAaClick(Sender: TObject);
begin
 with dmlEmpPayHist, qryEmpPayHist, dmlDBooks do
 begin
  dbsDBooks.StartTransaction ;
  try
   ApplyUpdates ;
   dbsDBooks.Commit ;
  except
   dbsDBooks.Rollback ;
   raise ;
  end ;
  CommitUpdates ;
  mniPSStandardAsClick(Sender) ; {Go back to standard view if not there.}
  Close ;
  Open ;
 end ;
end;

procedure TfrmEmpPayHist.mniPDeleteAdClick(Sender: TObject);
begin
 dmlEmpPayHist.qryEmpPayHist.Delete ;
end;

procedure TfrmEmpPayHist.mniPInsertAiClick(Sender: TObject);
begin
 pgcEmpPayHist.ActivePage := tbsDetails ;
 dedBeginDate.SetFocus ;
 dmlEmpPayHist.qryEmpPayHist.Insert ;
end;

end.
```



File Name: ChildDepartment.pas

```
{**************************************************************************
 D-Books 0.1 Prototype; Naval Postgraduate School          Begun 07/11/96
 Written by Rob Cameron and Ken Carrick

 Module: Departments
  Notes:
**************************************************************************}
unit ChildDepartment;

interface

uses
 Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 StdCtrls, Buttons, ExtCtrls, DBCtrls, ComCtrls, Grids, DBGrids, Mask,
 Frmszlmt, SendKey, DB, DBTables, Menus;

type
 TfrmDepartment = class(TForm)
  pgcDepartment: TPageControl;
  tbsDepartment: TTabSheet;
  pnlDept: TPanel;
  bbnContact: TBitBtn;
  tbsDetails: TTabSheet;
  lblQuicjSearch: TLabel;
  dgrDepartment: TDBGrid;
```

```
cboDeptName: TComboBox;
lblDistCode: TLabel;
dedDistCode: TDBEdit;
lblUIC: TLabel;
dedUIC: TDBEdit;
gboFincancialStatus: TGroupBox;
lblTotAuth: TLabel;
dedTotalAuth: TDBEdit;
lblTotCharges: TLabel;
dedTotalCharges: TDBEdit;
lblBalanceAvail: TLabel;
dedBalAvail: TDBEdit;
fslDepartment: TfslFormSizeLimit;
lblCostCenter: TLabel;
lblSubCostCenter: TLabel;
lblActivityGroup: TLabel;
dlcActivityGroup: TDBLookupComboBox;
lblSubActivityGroup: TLabel;
dlcSubActivityGroup: TDBLookupComboBox;
dlcSubCostCenter: TDBLookupComboBox;
dlcCostCenter: TDBLookupComboBox;
lblName: TLabel;
dedName: TDBEdit;
lblOrgCode: TLabel;
dedOrgCode: TDBEdit;
lblMailStop: TLabel;
dedMailStop: TDBEdit;
dngDepartment: TDBNavigator;
mnuDepartments: TMainMenu;
mniDepartmentsAd: TMenuItem;
mniDInsertAi: TMenuItem;
mniDDeleteAd: TMenuItem;
mniDCancelRecAc: TMenuItem;
mniDApplyAa: TMenuItem;
mniDCancelAn: TMenuItem;
mniDShowAs: TMenuItem;
mniDSStandardAs: TMenuItem;
mniDSDeletedAd: TMenuItem;
mniDSModifiedAm: TMenuItem;
mniDSInsertedAi: TMenuItem;
mniDivider: TMenuItem;
mniDCloseAl: TMenuItem;
procedure FormCreate(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
```

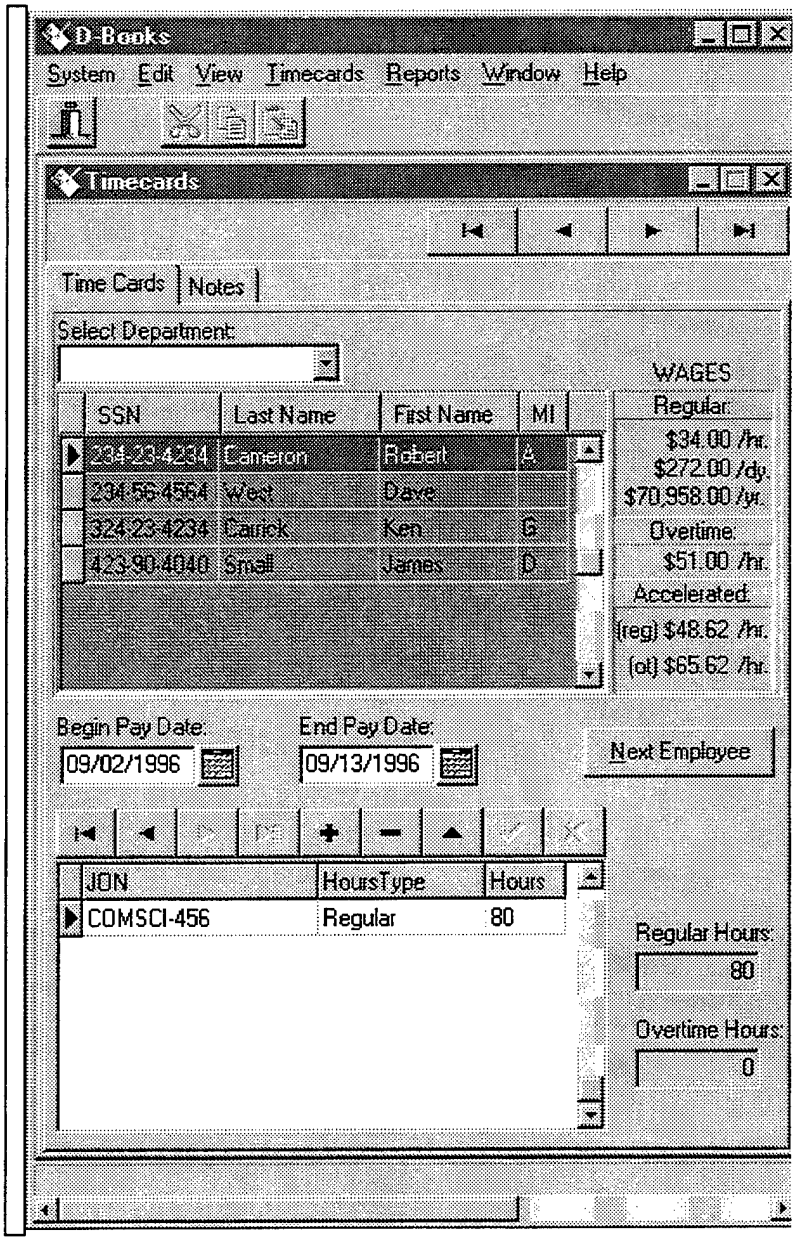
```
    procedure dngDepartmentClick(Sender: TObject; Button: TNavigateBtn);
    procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
    procedure SendKeyClick(Sender : TObject) ;
    procedure FormDestroy(Sender: TObject);
    procedure mniDSInsertedAiClick(Sender: TObject);
    procedure mniDSModifiedAmClick(Sender: TObject);
    procedure mniDSDeletedAdClick(Sender: TObject);
    procedure mniDSStandardAsClick(Sender: TObject);
    procedure mniDCancelRecAcClick(Sender: TObject);
    procedure mniDCancelAnClick(Sender: TObject);
    procedure mniDApplyAaClick(Sender: TObject);
    procedure mniDDeleteAdClick(Sender: TObject);
    procedure mniDInsertAiClick(Sender: TObject);
  private
    { Private declarations }
    procedure UpdateDeptComboBox ;
  public
    { Public declarations }
  end;


implementation

uses dmdDepartment, main, qryThread, dmdDbooks;

{$R *.DFM}
var
  sesDept : TSession ;
  dbsDept : TDatabase ;
  qryDept : TQuery ;

procedure TfrmDepartment.UpdateDeptComboBox ;
begin
  TQueryThread.Create(false, qryDept,'NAME',cboDeptName.Items) ;
end ;

procedure TfrmDepartment.SendKeyClick(Sender : TObject) ;
begin
  if sender is TDBEdit then
    with dmlDepartments do
      with Sender as TDBEdit do
        if qryDepartment.FieldByName(DataField).AsString = '' then
          frmMain.skyDBooks32.SendKeys ('{HOME}');
end ;
```

```
procedure TfrmDepartment.FormCreate(Sender: TObject);
begin
  pgcDepartment.ActivePage := tbsDepartment ;

  sesDept := TSession.Create(Self) ;
  sesDept.SessionName := 'session1' ;
  dbsDept := TDatabase.Create(Self) ;
  with dbsDept do
  begin
    AliasName := 'DBooksInterbase' ;
    DatabaseName := 'Database1' ;
    LoginPrompt := false ;
    Params := dmlDbooks.dbsDbooks.params ;
    SessionName := sesDept.SessionName ;
  end ;
  qryDept := TQuery.Create(Self) ;
  with qryDept do
  begin
    SessionName := sesDept.SessionName ;
    DatabaseName := dbsDept.DatabaseName ;
    SQL.Clear ;
    SQL.Add('Select * from TBL_DEPARTMENT') ;
  end ;
end;

procedure TfrmDepartment.FormShow(Sender: TObject);
begin
  Left := 386 ;
  Top := 0 ;
  UpdateDeptComboBox ;
end;

procedure TfrmDepartment.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  Action := caFree;
end;

procedure TfrmDepartment.dngDepartmentClick(Sender: TObject;
  Button: TNavigateBtn);
begin
  if Button = nbInsert then
  begin
    pgcDepartment.ActivePage := tbsDetails ;
    dedName.SetFocus ;
```

```
  end ;
end;

procedure TfrmDepartment.FormCloseQuery(Sender: TObject;
  var CanClose: Boolean);
begin
  mniDApplyAaClick(Self) ;
  CanClose := true ;
end;

procedure TfrmDepartment.FormDestroy(Sender: TObject);
begin
  qryDept.Free ;
  dbsDept.Free ;
  sesDept.Free ;
end;

procedure TfrmDepartment.mniDSInsertedAiClick(Sender: TObject);
begin
  with dmlDepartments, qryDepartment do
    UpdateRecordTypes := [rtInserted] ;
  with Sender as TMenuItem do Checked := true ;
  mniDepartmentsAd.Items[2].Caption := '&Delete New Department' ;
  dgrDepartment.Font.Color := clNavy ;
end;

procedure TfrmDepartment.mniDSModifiedAmClick(Sender: TObject);
begin
  with dmlDepartments, qryDepartment do
    UpdateRecordTypes := [rtModified] ;
  with Sender as TMenuItem do Checked := true ;
  mniDepartmentsAd.Items[2].Caption := '&Undo Changes' ;
  dgrDepartment.Font.Color := clMaroon ;
end;

procedure TfrmDepartment.mniDSDeletedAdClick(Sender: TObject);
begin
  with dmlDepartments, qryDepartment do
    UpdateRecordTypes := [rtDeleted] ;
  with Sender as TMenuItem do Checked := true ;
  mniDepartmentsAd.Items[2].Caption := '&Undelete Department' ;
  dgrDepartment.Font.Color := clRed ;
end;

procedure TfrmDepartment.mniDSStandardAsClick(Sender: TObject);
```

```
begin
  with dmlDepartments, qryDepartment do
    UpdateRecordTypes := [rtModified, rtInserted, rtUnmodified] ;
  mniDepartmentsAd.Items[5].Items[0].Checked := true ;
  mniDepartmentsAd.Items[2].Caption := '&Cancel Record Updates' ;
  dgrDepartment.Font.Color := clWindowText ;
end;

procedure TfrmDepartment.mniDCancelRecAcClick(Sender: TObject);
begin
  if messagedlg('Are you sure?',mtWarning,[mbYes,mbNo],0) = mrYes then
    dmlDepartments.qryDepartment.RevertRecord ;
end;

procedure TfrmDepartment.mniDCancelAnClick(Sender: TObject);
begin
  dmlDepartments.qryDepartment.CancelUpdates ;
  mniDSStandardAsClick(Sender) ; {Go back to standard view if not there.}
end;

procedure TfrmDepartment.mniDApplyAaClick(Sender: TObject);
begin
  with dmlDepartments, qryDepartment, dmlDBooks do
  begin
    dbsDBooks.StartTransaction ;
    try
      ApplyUpdates ;
      dbsDBooks.Commit ;
    except
      dbsDBooks.Rollback ;
      raise ;
    end ;
    CommitUpdates ;
    mniDSStandardAsClick(Sender) ; {Go back to standard view if not there.}
    Close ;
    Open ;
  end ;
end;

procedure TfrmDepartment.mniDDeleteAdClick(Sender: TObject);
begin
  dmlDepartments.qryDepartment.Delete ;
end;

procedure TfrmDepartment.mniDInsertAiClick(Sender: TObject);
```

```
begin
  pgcDepartment.ActivePage := tbsDetails ;
  dedName.SetFocus ;
  dmlDepartments.qryDepartment.Insert ;
end;

end.
```



File Name: ChildTimecard.pas

```
{*********************************************************************
 D-Books 0.1 Prototype; Naval Postgraduate School      Begun 07/11/96
 Written by Rob Cameron and Ken Carrick

Module:  Time Cards
 Notes:
*********************************************************************}
unit ChildTimeCard;

interface
```

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, Buttons, ExtCtrls, DBCtrls, Grids, DBGrids, StdCtrls,
  ComCtrls, Mask, Frmszlmt, Menus, DB, DBTables, QryThread, Calndar;

type
  TfrmTimecard = class(TForm)
    pnlTimeCard: TPanel;
    pgcTimeCard: TPageControl;
    tbsTimeCard: TTabSheet;
    fslTimecard: TfslFormSizeLimit;
    mnuTimeCards: TMainMenu;
    mniTimeCardsAt: TMenuItem;
    mniTInsertAi: TMenuItem;
    mniTDeleteAd: TMenuItem;
    mniTCancelRecAc: TMenuItem;
    mniTApplyAllAa: TMenuItem;
    mniTCancelAllAn: TMenuItem;
    mniTShowAs: TMenuItem;
    mniTSStandardAs: TMenuItem;
    mniTSDeletedAd: TMenuItem;
    mniTSEditedAe: TMenuItem;
    mniTSInsertedAi: TMenuItem;
    mniSeperator: TMenuItem;
    mniTCloseAl: TMenuItem;
    pnlUpper: TPanel;
    cboDeptName: TComboBox;
    lblDeptName: TLabel;
    lblBegindate: TLabel;
    dedBeginDate: TDBEdit;
    lblEndDate: TLabel;
    dedEndDate: TDBEdit;
    lblRegHrs: TLabel;
    dedRegHours: TDBEdit;
    lblOTHrs: TLabel;
    dedOTHours: TDBEdit;
    spbDateOnboard: TSpeedButton;
    spbEndDate: TSpeedButton;
    pnlEmpGrid: TPanel;
    hdcEmployeeList: THeaderControl;
    dgrEmployees: TDBGrid;
    tbsNotes: TTabSheet;
    lblNotes: TLabel;
    dmmNotes: TDBMemo;
    btnNextEmp: TButton;
```

```
cldTimecard: TCalendarDialog;
pnlWages: TPanel;
dblHourly: TDBText;
lblHrly: TLabel;
lblWeely: TLabel;
dblDaily: TDBText;
lblYearly: TLabel;
dblAnnual: TDBText;
lblRegular: TLabel;
bvlAccel: TBevel;
lblWages: TLabel;
lblAccel: TLabel;
bvlReg: TBevel;
dblOT: TDBText;
lblOTRate: TLabel;
lblOT: TLabel;
bvlOT: TBevel;
dblRegAccel: TDBText;
dblRegOT: TDBText;
lblAccelRegHrs: TLabel;
lblAccelOTRate: TLabel;
lblAccelOT: TLabel;
lblAccelReg: TLabel;
dgrLaborRec: TDBGrid;
dngLaborRec: TDBNavigator;
dngEmployee: TDBNavigator;
dngTimecard: TDBNavigator;
procedure FormShow(Sender: TObject);
procedure mniTInsertAiClick(Sender: TObject);
procedure mniTDeleteAdClick(Sender: TObject);
procedure mniTCancelRecAcClick(Sender: TObject);
procedure mniTApplyAllAaClick(Sender: TObject);
procedure mniTCancelAllAnClick(Sender: TObject);
procedure mniTSStandardAsClick(Sender: TObject);
procedure mniTSEditedAeClick(Sender: TObject);
procedure mniTSDeletedAdClick(Sender: TObject);
procedure mniTSInsertedAiClick(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
procedure FormCreate(Sender: TObject);
procedure FormDestroy(Sender: TObject);
procedure cboDeptNameChange(Sender: TObject);
procedure spbDateOnboardClick(Sender: TObject);
procedure spbEndDateClick(Sender: TObject);
procedure btnNextEmpClick(Sender: TObject);
```

```
    procedure ApplyTimecardUpdates ;
    procedure ApplyLaborRecordUpdates ;
  private
    { Private declarations }
    procedure UpdateDeptComboBox ;
  public
    { Public declarations }
  end;

var
  frmTimecard: TfrmTimecard;

implementation

uses dmdTimecards, dmdDbooks, dmdEmployees, dmdEmpPayHist, dmdJobOrder;

{$R *.DFM}

var
  sesTimeCard : TSession ;
  dbsTimeCard : TDatabase ;
  qryDept1    : TQuery ;
  qryDept2    : TQuery ;
  Closing     : boolean ;

procedure  TfrmTimecard.ApplyTimecardUpdates ;
var
  bmkEmployee : TBookmark ;
begin
  with dmlTimecards, qryTimecard, dmlDBooks do
  begin
    dbsDBooks.StartTransaction ;
    try
      bmkEmployee := qryEmployee.GetBookmark ;
      ApplyUpdates ;
      dbsDBooks.Commit ;
    except
      dbsDBooks.Rollback ;
      raise ;
    end ;
    CommitUpdates ;
    Close ;
    Open ;
    try
      qryEmployee.GotoBookmark(bmkEmployee) ;
```

```
    except
      on E : exception do
      ;
    end ;
    qryEmployee.Freebookmark(bmkEmployee) ;
    if not Closing then
      UpdateDeptComboBox ;
  end ;
end ;


procedure  TfrmTimecard.ApplyLaborRecordUpdates ;
begin
  with dmlTimecards, qryLaborRec, dmlDBooks do
  begin
    dbsDBooks.StartTransaction ;
    try
      ApplyUpdates ;
      dbsDBooks.Commit ;
    except
      dbsDBooks.Rollback ;
      raise ;
    end ;
    CommitUpdates ;
     Close ;
    Open ;
  end ;
end ;


procedure TfrmTimecard.FormShow(Sender: TObject);
begin
  Left := 386 ;
  Top := 0 ;
  UpdateDeptComboBox ;
  dedBeginDate.SetFocus ;
end;

procedure TfrmTimecard.mniTInsertAiClick(Sender : TObject);
begin
  dmlTimecards.qryTimecard.Insert ;
end;

procedure TfrmTimecard.mniTDeleteAdClick(Sender: TObject);
begin
```

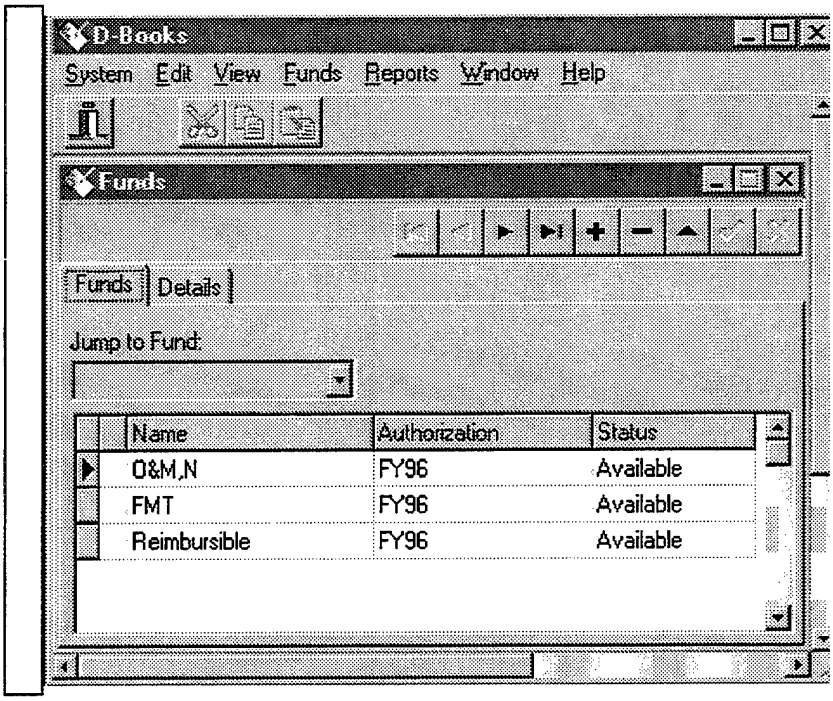
```
  if messagedlg('You must delete all labor records associated with'+#13+
   'this time card listed in the grid before deleting'+#13+
   'the timecard.  Continue?',mtConfirmation,[mbYes,mbNo],0)= mrYes Then
    dmlTimecards.qryTimecard.Delete
  else
    messagedlg('Timecard not deleted.',mtInformation,[mbOK],0) ;
end;

procedure TfrmTimecard.mniTCancelRecAcClick(Sender: TObject);
begin
  if messagedlg('Are you sure?',mtWarning,[mbYes,mbNo],0) = mrYes then
  begin
    dmlTimecards.qryTimecard.RevertRecord ;
  end ;
end;

procedure TfrmTimecard.mniTApplyAllAaClick(Sender: TObject);
begin
  dmlTimecards.UpdateJobOrderCharges ;{Apply Labor charges to Job Orders}
  ApplyTimecardUpdates ;
  ApplyLaborRecordUpdates ;
end ;

procedure TfrmTimecard.mniTCancelAllAnClick(Sender: TObject);
begin
  dmlTimecards.qryTimecard.CancelUpdates ;
end;

procedure TfrmTimecard.mniTSStandardAsClick(Sender: TObject);
begin
(*  with dmlTimecards, qryTimecard do
    UpdateRecordTypes := [rtModified, rtInserted, rtUnmodified] ;
  mniTimecardsAt.Items[5].Items[0].Checked := true ;
  mniTimecardsAt.Items[2].Caption := '&Cancel Record Updates' ;
  dgrTimecards.Font.Color := clWindowText ;   *)
end;

procedure TfrmTimecard.mniTSEditedAeClick(Sender: TObject);
begin
(*  with dmlTimecards, qryTimecard do
    UpdateRecordTypes := [rtModified] ;
  with Sender as TMenuItem do Checked := true ;
  mniTimecardsAt.Items[2].Caption := '&Undo Changes' ;
  dgrTimecards.Font.Color := clMaroon ;  *)
end;
```

```
procedure TfrmTimecard.mniTSDeletedAdClick(Sender: TObject);
begin
(*  with dmlTimecards, qryTimecard do
    UpdateRecordTypes := [rtDeleted] ;
  with Sender as TMenuItem do Checked := true ;
  mniTimecardsAt.Items[2].Caption := '&Undelete Timecard' ;
  dgrTimecards.Font.Color := clRed ;   *)
end;

procedure TfrmTimecard.mniTSInsertedAiClick(Sender: TObject);
begin
 (* with dmlTimecards, qryTimecard do
    UpdateRecordTypes := [rtInserted] ;
  with Sender as TMenuItem do Checked := true ;
  mniTimecardsAt.Items[2].Caption := '&Delete New Timecard' ;
  dgrTimecards.Font.Color := clNavy ;  *)
end;

procedure TfrmTimecard.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  Action := caFree;
end;

procedure TfrmTimecard.FormCloseQuery(Sender: TObject;
  var CanClose: Boolean);
begin
  Closing := true ;
  if dmlTimecards.qryTimecard.UpdatesPending then
    mniTApplyAllAaClick(Self) ;
  CanClose := true ;
end;

procedure TfrmTimecard.FormCreate(Sender: TObject);
begin
  pgcTimecard.ActivePage := tbsTimecard ;
  Closing := false ;
  sesTimecard := TSession.Create(Self) ;
  sesTimecard.SessionName := 'sesTimecard1' ;
  dbsTimecard := TDatabase.Create(Self) ;
  with dbsTimecard do
  begin
    AliasName := 'DBooksOracle' ;
    DatabaseName := 'dbsTimecard1' ;
```

```
    LoginPrompt := false ;
    Params := dmlDbooks.dbsDbooks.params ;
    SessionName := sesTimecard.SessionName ;
   end ;
   qryDept1 := TQuery.Create(Self) ;
   with qryDept1 do
   begin
    SessionName := sesTimecard.SessionName ;
    DatabaseName := dbsTimecard.DatabaseName ;
    SQL.Clear ;
    SQL.Add('Select * from TBL_DEPARTMENT') ;
   end ;
end;

procedure TfrmTimecard.UpdateDeptComboBox ;
begin
  TQueryThread.Create(false, qryDept1,'NAME',cboDeptName.Items) ;
end ;

procedure TfrmTimecard.FormDestroy(Sender: TObject);
begin
  try
    qryDept1.Free ;
    dbsTimecard.Free ;
    sesTimecard.Free ;
  except
    on E : exception do
      ;
  end ;
end;

procedure TfrmTimecard.cboDeptNameChange(Sender: TObject);
var
  strDeptID : String ;
begin
  with dmlTimecards.qryLookupID do
  begin
    Close ;
    Params[0].AsString := cboDeptName.Text ;
    Open ;
    strDeptID := FieldByName('DEPARTMENT_ID').AsString ;
  end ;
  with dmlTimeCards do
    qryEmployee.Filter := 'DEPARTMENT_ID = ' +strDeptID;
end;
```

```
procedure TfrmTimecard.spbDateOnboardClick(Sender: TObject);
begin
  with dmlTimecards do
  begin
    cldTimecard.Date :=        {Initialize with current Date }
      FormatDateTime('mm/dd/yyyy',qryTimecardBEGIN_PAY_DATE.asDateTime);
    if cldTimecard.Execute then
    begin
      if not (qryTimecard.State in [dsEdit,dsInsert]) then
      begin
        qryTimecard.Edit ;
      end ;
      qryTimecardBEGIN_PAY_DATE.AsString := cldTimecard.Date ; {Store value into
field}
    end ;
  end;
end;

procedure TfrmTimecard.spbEndDateClick(Sender: TObject);
begin
  with dmlTimecards do
  begin
    cldTimecard.Date :=        {Initialize with current Date }
      FormatDateTime('mm/dd/yyyy',qryTimecardEND_PAY_DATE.asDateTime) ;
    if cldTimecard.Execute then
    begin
      if not (qryTimecard.State in [dsEdit,dsInsert]) then
      begin
        qryTimecard.Edit ;
      end ;
      qryTimecardEND_PAY_DATE.AsString := cldTimecard.Date ; {Store value into
field}
    end ;
  end;
end;

procedure TfrmTimecard.btnNextEmpClick(Sender: TObject);
begin
  mniTApplyAllAaClick(Sender) ;
  dmlTimecards.qryEmployee.Next ;
end;

end.
```

```
aReport:=SimpForm.SimpRep;
    1 : aReport:=Bioform.BioRep;
    2 : aReport:=mdform.mdRep;
    3 : aReport:=LabelForm.Rep;
    4 : aReport:=TextRep.Rep;
  end;
  aReport.DisplayPrintDialog:=PrintDialogChk.Checked;
  if OrientationCombo.ItemIndex=0 then
    aReport.Orientation:=poPortrait
  else
    aReport.Orientation:=poLandscape;
end;

procedure TTQuickReportDemo.PrintBtnClick(Sender: TObject);
begin
  PickReport;
  aReport.Print;
end;
```



File Name: ChildFund.pas

```
{*************************************************************************
 D-Books 0.1 Prototype; Naval Postgraduate School      Begun 07/11/96
 Written by Rob Cameron and Ken Carrick

Module:
 Notes:
*************************************************************************}
unit ChildFund;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, Frmszlmt, ComCtrls, Buttons, ExtCtrls, SendKey, Grids,
  DBGrids, Menus, DBCtrls, StdCtrls, Mask, Calndar, DB, DBTables ;

type
  TfrmFund = class(TForm)
    pnlFund: TPanel;
    pgcFund: TPageControl;
    fsl: TfslFormSizeLimit;
    skyFund: TSendKey;
    dngFund: TDBNavigator;
    mnuFunds: TMainMenu;
    tbsFundList: TTabSheet;
```

```
mniFundsAf: TMenuItem;
tbsDetails: TTabSheet;
lblName: TLabel;
dedName: TDBEdit;
lblAuthorization: TLabel;
dedAuthorization: TDBEdit;
lblBeginDate: TLabel;
dedBeginDate: TDBEdit;
lblEndDate: TLabel;
dedEndDate: TDBEdit;
lblType: TLabel;
lblStatus: TLabel;
lblInitBal: TLabel;
dedInitBalance: TDBEdit;
lblTotalCharges: TLabel;
dedTotalCharges: TDBEdit;
cboFundName: TComboBox;
lblJump: TLabel;
dlcType: TDBLookupComboBox;
dlcStatus: TDBLookupComboBox;
mniFInsertAi: TMenuItem;
mniFDeleteAd: TMenuItem;
mnniFCancelAc: TMenuItem;
mniFApplyAa: TMenuItem;
mniFCancelAn: TMenuItem;
mniFShowAs: TMenuItem;
mniFDivider1: TMenuItem;
mniFCloseAl: TMenuItem;
mniFSStandardAs: TMenuItem;
mniFSDeletedAd: TMenuItem;
mniFSEditedAe: TMenuItem;
mniFSInsertedAi: TMenuItem;
dgrFunds: TDBGrid;
cldFund: TCalendarDialog;
spbBeginDate: TSpeedButton;
spbEndDate: TSpeedButton;
lblAmountAvailable: TLabel;
dedAmountAvail: TDBEdit;
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure mniFInsertAiClick(Sender: TObject);
procedure mniFDeleteAdClick(Sender: TObject);
procedure mnniFCancelAcClick(Sender: TObject);
procedure mniFApplyAaClick(Sender: TObject);
procedure mniFCancelAnClick(Sender: TObject);
procedure mniFSStandardAsClick(Sender: TObject);
```

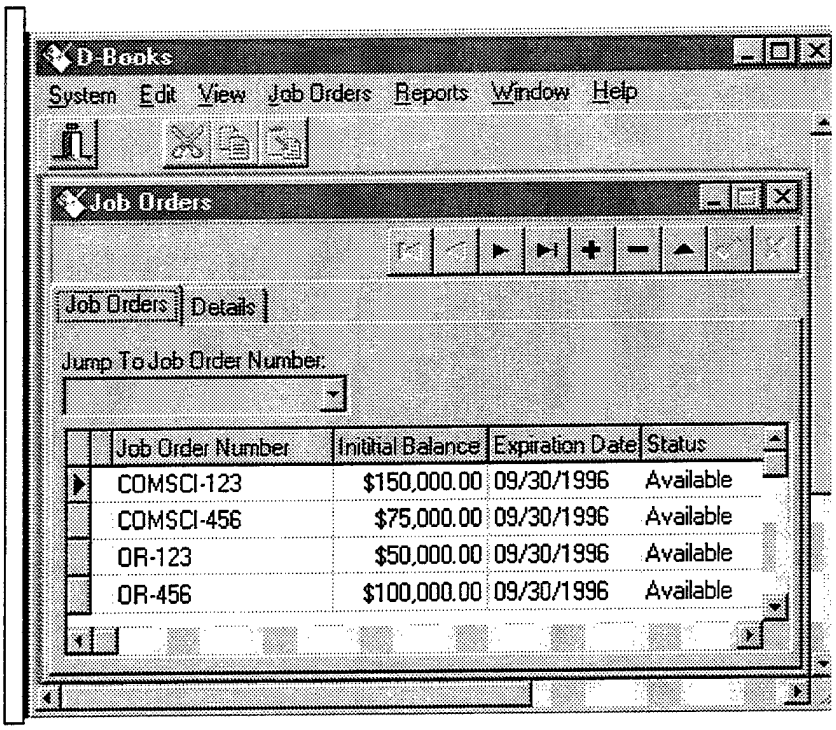
```
    procedure mniFSDeletedAdClick(Sender: TObject);
    procedure mniFSEditedAeClick(Sender: TObject);
    procedure mniFSInsertedAiClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure spbBeginDateClick(Sender: TObject);
    procedure spbEndDateClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure UpdateFundComboBox ;
    procedure FormDestroy(Sender: TObject);
    procedure cboFundNameChange(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
    procedure dngFundClick(Sender: TObject; Button: TNavigateBtn);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

implementation

uses dmdFund, qryThread, dmdDbooks;

{$R *.DFM}

var
  sesFund : TSession ;
  dbsFund : TDatabase ;
  qryFund : TQuery ;
  Closing : boolean ;
procedure TfrmFund.UpdateFundComboBox ;
begin
  TQueryThread.Create(false, qryFund,'NAME',cboFundName.Items) ;
end ;

procedure TfrmFund.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action := caFree ;
end;

procedure TfrmFund.mniFInsertAiClick(Sender: TObject);
begin
  pgcFund.ActivePage := tbsDetails ;
  dedName.SetFocus ;
  dmlFunds.qryFund.Insert ;
end;
```

```
procedure TfrmFund.mniFDeleteAdClick(Sender: TObject);
begin
  dmlFunds.qryFund.Delete ;
end;

procedure TfrmFund.mnniFCancelAcClick(Sender: TObject);
begin
  if messagedlg('Are you sure?',mtWarning,[mbYes,mbNo],0) = mrYes then
    dmlFunds.qryFund.RevertRecord ;
end;

procedure TfrmFund.mniFApplyAaClick(Sender: TObject);
begin
  with dmlFunds, qryFund, dmlDBooks do
  begin
    dbsDBooks.StartTransaction ;
    try
      ApplyUpdates ;
      dbsDBooks.Commit ;
    except
      dbsDBooks.Rollback ;
      raise ;
    end ;
    CommitUpdates ; {Go back to standard view if not there.}
    mnifSStandardAsClick(Sender) ;
    Close ;
    Open ;
    if not Closing then
      UpdateFundComboBox ;
  end ;
end;

procedure TfrmFund.mniFCancelAnClick(Sender: TObject);
begin
  dmlFunds.qryFund.CancelUpdates ;
  mniFSStandardAsClick(Sender) ;{Go back to standard view if not there.}
end;

procedure TfrmFund.mniFSStandardAsClick(Sender: TObject);
begin
  with dmlFunds, qryFund do
    UpdateRecordTypes := [rtModified, rtInserted, rtUnmodified] ;
  mniFundsAf.Items[5].Items[0].Checked := true ;
  mniFundsAf.Items[2].Caption := '&Cancel Record Updates' ;
```

```
  dgrFunds.Font.Color := clWindowText ;
end;

procedure TfrmFund.mniFSDeletedAdClick(Sender: TObject);
begin
  with dmlFunds, qryFund do
    UpdateRecordTypes := [rtDeleted] ;
  with Sender as TMenuItem do Checked := true ;
  mniFundsAf.Items[2].Caption := '&Undelete Fund' ;
  dgrFunds.Font.Color := clRed ;
end;

procedure TfrmFund.mniFSEditedAeClick(Sender: TObject);
begin
  with dmlFunds, qryFund do
    UpdateRecordTypes := [rtModified] ;
  with Sender as TMenuItem do Checked := true ;
  mniFundsAf.Items[2].Caption := '&Undo Changes' ;
  dgrFunds.Font.Color := clMaroon ;
end;

procedure TfrmFund.mniFSInsertedAiClick(Sender: TObject);
begin
  with dmlFunds, qryFund do
    UpdateRecordTypes := [rtInserted] ;
  with Sender as TMenuItem do Checked := true ;
  mniFundsAf.Items[2].Caption := '&Delete New Fund' ;
  dgrFunds.Font.Color := clNavy ;
end;

procedure TfrmFund.FormShow(Sender: TObject);
begin
  Left := 386 ;
  Top := 0 ;
  UpdateFundComboBox ;
end;

procedure TfrmFund.spbBeginDateClick(Sender: TObject);
begin
  with dmlFunds do
  begin
    cldFund.Date :=        {Initialize with current Date }
      FormatDateTime('mm/dd/yyyy',qryFundEND_DATE.asDateTime) ;
    if cldFund.Execute then
    begin
```

```
    if not (qryFund.State in [dsEdit,dsInsert]) then
    begin
      qryFund.Edit ;
    end ;
    qryFundEND_DATE.AsString := cldFund.Date ;{Store value into field}
   end ;
  end;
end;

procedure TfrmFund.spbEndDateClick(Sender: TObject);
begin
  with dmlFunds do
  begin
    cldFund.Date :=        {Initialize with current Date }
     FormatDateTime('mm/dd/yyyy',qryFundEND_DATE.asDateTime) ;
    if cldFund.Execute then
    begin
     if not (qryFund.State in [dsEdit,dsInsert]) then
     begin
       qryFund.Edit ;
     end ;
     qryFundEND_DATE.AsString := cldFund.Date ;{Store value into field}
    end ;
  end;
end;

procedure TfrmFund.FormCreate(Sender: TObject);
begin
  pgcFund.ActivePage := tbsFundList ;
  Closing := false ;
  sesFund := TSession.Create(Self) ;
  sesFund.SessionName := 'sesFund1' ;
  dbsFund := TDatabase.Create(Self) ;
  with dbsFund do
  begin
    AliasName := 'DBooksOracle' ;
    DatabaseName := 'DbsFund1' ;
    LoginPrompt := false ;
    Params := dmlDbooks.dbsDbooks.params ;
    SessionName := sesFund.SessionName ;
  end ;
  qryFund := TQuery.Create(Self) ;
  with qryFund do
  begin
    SessionName := sesFund.SessionName ;
```

```
    DatabaseName := dbsFund.DatabaseName ;
    SQL.Clear ;
    SQL.Add('Select * from TBL_FUND') ;
  end ;
end;

procedure TfrmFund.FormDestroy(Sender: TObject);
begin
  try
    qryFund.Free ;
    dbsFund.Free ;
    sesFund.Free ;
  except
    on E : exception do
      ;
  end ;
end;

procedure TfrmFund.cboFundNameChange(Sender: TObject);
begin
  dmlFunds.qryFund.Locate('NAME',cboFundName.Text,[loCaseInsensitive]);
end;

procedure TfrmFund.FormCloseQuery(Sender: TObject; var
CanClose: Boolean);
begin
  Closing := true ;
  if dmlFunds.qryFund.UpdatesPending then
    mniFApplyAaClick(Self) ;
  CanClose := true ;
end;

procedure TfrmFund.dngFundClick(Sender: TObject; Button: TNavigateBtn);
begin
  if Button = nbInsert then
  begin
    pgcFund.ActivePage := tbsDetails ;
    dedName.SetFocus ;
  end ;
end;

end.
```



File Name: ChildJobOrder.pas

```
{*************************************************************************
 D-Books 0.1 Prototype; Naval Postgraduate School      Begun 07/11/96
 Written by Rob Cameron and Ken Carrick

Module: Job Order
 Notes:
*************************************************************************}
unit ChildJobOrder;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, Frmszlmt, ComCtrls, Buttons, ExtCtrls, SendKey, Grids,
  DBGrids, Menus, DBCtrls, StdCtrls, Mask, Calndar, DB, DBTables ;

type
  TfrmJobOrder = class(TForm)
    pnlJobOrder: TPanel;
    pgcJobOrder: TPageControl;
    fslJobOrder: TfslFormSizeLimit;
    skyJobOrder: TSendKey;
    dngJobOrder: TDBNavigator;
    mnuJobOrder: TMainMenu;
    tbsJobOrderList: TTabSheet;
```

```
dgrJobOrder: TDBGrid;
tbsDetails: TTabSheet;
cboJobOrder: TComboBox;
mniJobOrderAj: TMenuItem;
mniJInsertAi: TMenuItem;
mniJDeleteAd: TMenuItem;
mniJCancelAc: TMenuItem;
mniJApplyAa: TMenuItem;
mniJCancelAll: TMenuItem;
mniJShowAs: TMenuItem;
mniJDivider: TMenuItem;
mniCloseAl: TMenuItem;
mniJSStandardAs: TMenuItem;
mniJSDeletedAd: TMenuItem;
mniJSEdit: TMenuItem;
mniJSInserted: TMenuItem;
lblJON: TLabel;
lblJONEdit: TLabel;
dedJON: TDBEdit;
lblExpirationDate: TLabel;
dedExpireDate: TDBEdit;
lblCharges: TLabel;
dedCharges: TDBEdit;
dlcType: TDBLookupComboBox;
dlcStatus: TDBLookupComboBox;
dlcFund: TDBLookupComboBox;
lblType: TLabel;
lblStatus: TLabel;
lblFund: TLabel;
lblInitBal: TLabel;
dedInitBal: TDBEdit;
dedAmountAvail: TDBEdit;
lblAmountAvailable: TLabel;
spbBeginDate: TSpeedButton;
cldJobOrder: TCalendarDialog;
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure mniJInsertAiClick(Sender: TObject);
procedure mniJDeleteAdClick(Sender: TObject);
procedure mniJCancelAcClick(Sender: TObject);
procedure mniJApplyAaClick(Sender: TObject);
procedure mniJCancelAllClick(Sender: TObject);
procedure mniJSStandardAsClick(Sender: TObject);
procedure mniJSDeletedAdClick(Sender: TObject);
procedure mniJSEditClick(Sender: TObject);
procedure FormShow(Sender: TObject);
```

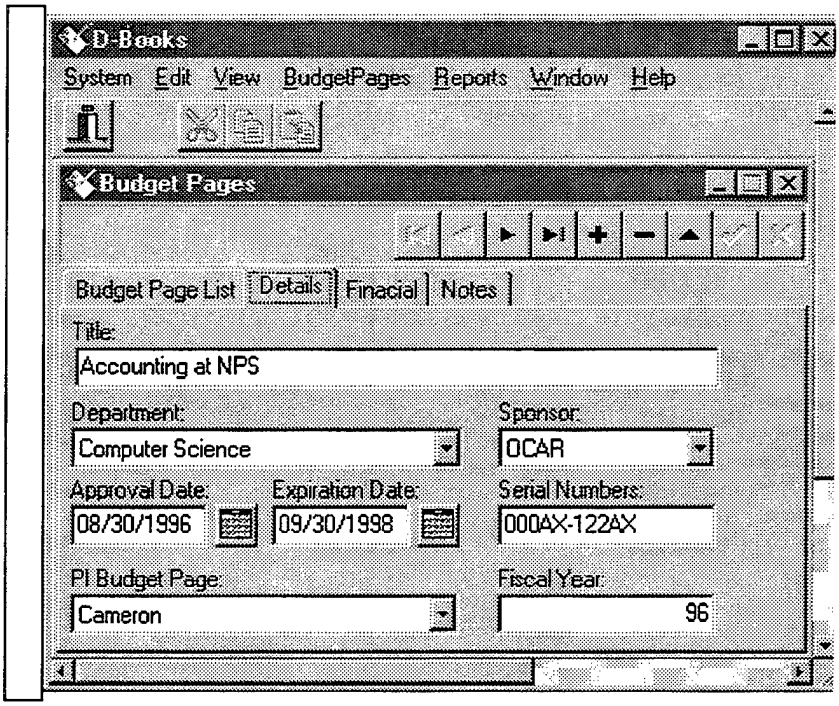
```
    procedure spbBeginDateClick(Sender: TObject);
    procedure dngJobOrderClick(Sender: TObject; Button: TNavigateBtn);
    procedure dgrJobOrderEditButtonClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure UpdateJobOrderComboBox ;
    procedure cboJobOrderChange(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
    procedure FormDestroy(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

implementation

uses dmdJobOrder, qryThread, dmdDbooks, dmdFund;

{$R *.DFM}

var
  sesJobOrder : TSession ;
  dbsJobOrder : TDatabase ;
  qryJobOrder : TQuery ;
  Closing     : boolean ;

procedure TfrmJobOrder.UpdateJobOrderComboBox ;
begin
  TQueryThread.Create(false, qryJobOrder,'JOB_ORDER_NUMBER',
    cboJobOrder.Items) ;
end ;

procedure TfrmJobOrder.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
  Action := caFree ;
end;

procedure TfrmJobOrder.mniJInsertAiClick(Sender: TObject);
begin
  pgcJobOrder.ActivePage := tbsDetails ;
  dedJON.SetFocus ;
  dmlJobOrders.qryJobOrder.Insert ;
end;
```

```
procedure TfrmJobOrder.mniJDeleteAdClick(Sender: TObject);
begin
  dmlJobOrders.qryJobOrder.Delete ;
end;

procedure TfrmJobOrder.mniJCancelAcClick(Sender: TObject);
begin
  if messagedlg('Are you sure?',mtWarning,[mbYes,mbNo],0) = mrYes then
    dmlJobOrders.qryJobOrder.RevertRecord ;
end;

procedure TfrmJobOrder.mniJApplyAaClick(Sender: TObject);
begin
  with dmlJobOrders, qryJobOrder, dmlDBooks do
  begin
    dbsDBooks.StartTransaction ;
    try
      ApplyUpdates ;
      dbsDBooks.Commit ;
    except
      dbsDBooks.Rollback ;
      raise ;
    end ;
    CommitUpdates ;
    mniJSStandardAsClick(Sender);{Go back to standard view if not there}
    Close ;
    Open ;
    if not Closing then
      UpdateJobOrderComboBox ;
  end ;
end;

procedure TfrmJobOrder.mniJCancelAllClick(Sender: TObject);
begin
  dmlJobOrders.qryJobOrder.CancelUpdates ;
  mniJSStandardAsClick(Sender);{Go back to standard view if not there}
end;

procedure TfrmJobOrder.mniJSStandardAsClick(Sender: TObject);
begin
  with dmlJobOrders, qryJobOrder do
    UpdateRecordTypes := [rtModified, rtInserted, rtUnmodified] ;
  mniJobOrderAj.Items[5].Items[0].Checked := true ;
  mniJobOrderAj.Items[2].Caption := '&Cancel Record Updates' ;
  dgrJobOrder.Font.Color := clWindowText ;
```

```
end;

procedure TfrmJobOrder.mniJSDeletedAdClick(Sender: TObject);
begin
 with dmlJobOrders, qryJobOrder do
  UpdateRecordTypes := [rtDeleted] ;
 with Sender as TMenuItem do Checked := true ;
 mniJobOrderAj.Items[2].Caption := '&Undelete Job Order' ;
 dgrJobOrder.Font.Color := clRed ;
end;

procedure TfrmJobOrder.mniJSEditClick(Sender: TObject);
begin
 with dmlJobOrders, qryJobOrder do
  UpdateRecordTypes := [rtModified] ;
 with Sender as TMenuItem do Checked := true ;
 mniJobOrderAj.Items[2].Caption := '&Undo Changes' ;
 dgrJobOrder.Font.Color := clMaroon ;
end;

procedure TfrmJobOrder.FormShow(Sender: TObject);
begin
 Left := 386 ;
 Top := 250 ;
// UpdateJobOrderComboBox ;
end;

procedure TfrmJobOrder.spbBeginDateClick(Sender: TObject);
begin
 with dmlJobOrders do
 begin
  cldJobOrder.Date :=       {Initialize with current Date }
   FormatDateTime('mm/dd/yyyy',
              qryJobOrderEXPIRATION_DATE.asDateTime) ;
  if cldJobOrder.Execute then
  begin
   if not (qryJobOrder.State in [dsEdit,dsInsert]) then
   begin
    qryJobOrder.Edit ;
   end ;       {Store value into field}
   qryJobOrderEXPIRATION_DATE.AsString := cldJobOrder.Date ;
  end ;
 end;
end;
```

```
procedure TfrmJobOrder.dngJobOrderClick(Sender: TObject;
  Button: TNavigateBtn);
begin
  if Button = nbInsert then
  begin
    pgcJobOrder.ActivePage := tbsDetails ;
    dedJON.SetFocus ;
  end ;
end;

procedure TfrmJobOrder.dgrJobOrderEditButtonClick(Sender: TObject);
begin
  with dmlJobOrders, dgrJobOrder do
  begin
    cldJobOrder.Date :=        {Initialize with current Date }
      FormatDateTime('mm/dd/yyyy',SelectedField.asDateTime) ;
    if cldJobOrder.Execute then
    begin
      if not (qryJobOrder.State in [dsEdit,dsInsert]) then
      begin
        qryJobOrder.Edit ;
      end ;        {Store value into field}
      SelectedField.AsString := cldJobOrder.Date ;
    end ;
  end;
end;

procedure TfrmJobOrder.FormCreate(Sender: TObject);
begin
  pgcJobOrder.ActivePage := tbsJobOrderList ;
  Closing := false ;
  sesJobOrder := TSession.Create(Self) ;
  sesJobOrder.SessionName := 'sesJobOrder1' ;
  dbsJobOrder := TDatabase.Create(Self) ;
  with dbsJobOrder do
  begin
    AliasName := 'DBooksOracle' ;
    DatabaseName := 'dbsJobOrder1' ;
    LoginPrompt := false ;
    Params := dmlDbooks.dbsDbooks.params ;
    SessionName := sesJobOrder.SessionName ;
  end ;
  qryJobOrder := TQuery.Create(Self) ;
  with qryJobOrder do
  begin
```

```
    SessionName := sesJobOrder.SessionName ;
    DatabaseName := dbsJobOrder.DatabaseName ;
    SQL.Clear ;
    SQL.Add('Select * from TBL_JOB_ORDER') ;
  end ;
end;

procedure TfrmJobOrder.cboJobOrderChange(Sender: TObject);
begin
  dmlJobOrders.qryJobOrder.Locate('JOB_ORDER_NUMBER',
  cboJobOrder.Text,[loCaseInsensitive]);
end;

procedure TfrmJobOrder.FormCloseQuery(Sender: TObject;
  var CanClose: Boolean);
begin
  Closing := true ;
  if dmlJobOrders.qryJobOrder.UpdatesPending then
    mniJApplyAaClick(Self) ;
  CanClose := true ;
end;

procedure TfrmJobOrder.FormDestroy(Sender: TObject);
begin
  try
    qryJobOrder.Free ;
    dbsJobOrder.Free ;
    sesJobOrder.Free ;
  except
    on E : exception do
    ;
  end ;
end;

end.
```



File Name: ChildBudgetPage.pas

```
{**************************************************************************
 D-Books 0.1 Prototype; Naval Postgraduate School      Begun 07/11/96
 Written by Rob Cameron and Ken Carrick

Module:
 Notes:
**************************************************************************}
unit ChildBudgetPage;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, Frmszlmt, ComCtrls, Buttons, ExtCtrls, SendKey, Grids,
  DBGrids, Menus, DBCtrls, StdCtrls, Mask, Calndar, DB, DBTables;

type
  TfrmBudgetPage = class(TForm)
    pnlBudgetPage: TPanel;
    pgcBudgetPage: TPageControl;
    fslBudgetPage: TfslFormSizeLimit;
    skyBudgetPage: TSendKey;
    dngBudgetPage: TDBNavigator;
    tbsList: TTabSheet;
    tbsDetails: TTabSheet;
```

cboTitle: TComboBox;
lblJump: TLabel;
lblTitle: TLabel;
dedTitle: TDBEdit;
lblSerialNumber: TLabel;
dedSerialNumbers: TDBEdit;
lblApproval: TLabel;
dedApprovalDate: TDBEdit;
spbApproval: TSpeedButton;
lblExpirationDate: TLabel;
dedExpirationDate: TDBEdit;
spbExpiration: TSpeedButton;
Label5: TLabel;
dlcDepartment: TDBLookupComboBox;
Label6: TLabel;
dlcSponsor: TDBLookupComboBox;
lblPIEmployee: TLabel;
dlcPIEmployee: TDBLookupComboBox;
lblFiscalYear: TLabel;
dlcFiscalYear: TDBEdit;
tbsFinacial: TTabSheet;
lblJON: TLabel;
lblLaborJON: TLabel;
dlcJON: TDBLookupComboBox;
dlcLaborJON: TDBLookupComboBox;
tbsNotes: TTabSheet;
Label11: TLabel;
DBMemo1: TDBMemo;
lblFacultyauth: TLabel;
dedFacultyAuth: TDBEdit;
lblSupportauth: TLabel;
dedSupportLabAuth: TDBEdit;
lblOPTARauth: TLabel;
dedOPTARAuth: TDBEdit;
lblTravelAuth: TLabel;
dedTravelAuth: TDBEdit;
lblFacultyCosts: TLabel;
dedFacultyLaborCosts: TDBEdit;
lblSupportCosts: TLabel;
dedSupportLaborCosts: TDBEdit;
lblOPTARCosts: TLabel;
dedOPTARCosts: TDBEdit;
lblTravelCosts: TLabel;
dedTravelCosts: TDBEdit;
lblContractAuth: TLabel;

```
    dedContractAuth: TDBEdit;
    lblContractCosts: TLabel;
    dedContractCosts: TDBEdit;
    mnuBudgetPages: TMainMenu;
    mniBudgetPagesAe: TMenuItem;
    mniBInsertAi: TMenuItem;
    mniBDeleteAd: TMenuItem;
    mniBCanceRecAc: TMenuItem;
    mniBApplyAa: TMenuItem;
    mniBCancelAllAn: TMenuItem;
    mniBShowAs: TMenuItem;
    mniBSStandardAs: TMenuItem;
    mniBSDeletedAd: TMenuItem;
    mniBSEditedAe: TMenuItem;
    mniBSInsertedAi: TMenuItem;
    mniSeperator: TMenuItem;
    mniBCloseAl: TMenuItem;
    dgrBudgetPages: TDBGrid;
    cldBudgetPage: TCalendarDialog;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure mniBInsertAiClick(Sender: TObject);
    procedure mniBDeleteAdClick(Sender: TObject);
    procedure mniBCanceRecAcClick(Sender: TObject);
    procedure mniBApplyAaClick(Sender: TObject);
    procedure mniBCancelAllAnClick(Sender: TObject);
    procedure mniBSStandardAsClick(Sender: TObject);
    procedure mniBSDeletedAdClick(Sender: TObject);
    procedure mniBSEditedAeClick(Sender: TObject);
    procedure mniBSInsertedAiClick(Sender: TObject);
    procedure dngBudgetPageClick(Sender: TObject; Button: TNavigateBtn);
    procedure spbApprovalClick(Sender: TObject);
    procedure spbExpirationClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure cboTitleChange(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure UpdateBudgetPageComboBox ;
    procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

implementation
```

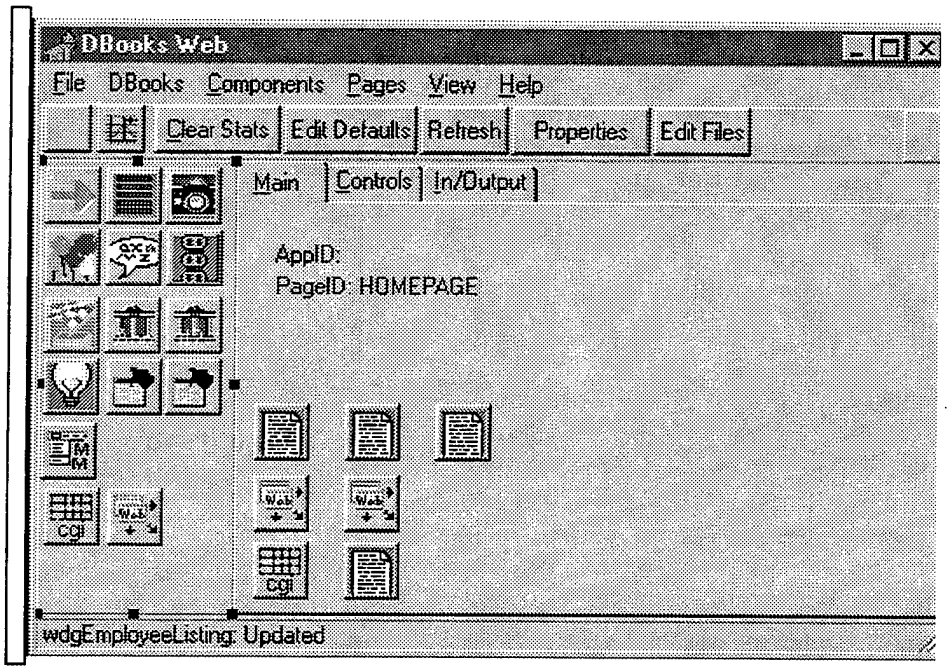
```
uses dmdBudgetPage, dmdDepartment, dmdDbooks, qryThread, dmdEmployees,
  dmdJobOrder ;

{$R *.DFM}

var
  sesBudgetPage : TSession ;
  dbsBudgetPage : TDatabase ;
  qryBudgetPage : TQuery ;
  Closing       : boolean ;
procedure TfrmBudgetPage.UpdateBudgetPageComboBox ;
begin
  TQueryThread.Create(false, qryBudgetPage,'TITLE',cboTitle.Items) ;
end ;

procedure TfrmBudgetPage.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
  Action := caFree ;
end;

procedure TfrmBudgetPage.mniBInsertAiClick(Sender: TObject);
begin
  pgcBudgetPage.ActivePage := tbsDetails ;
  dedTitle.SetFocus ;
  dmlBudgetPages.qryBudgetPage.Insert ;
end;

procedure TfrmBudgetPage.mniBDeleteAdClick(Sender: TObject);
begin
  dmlBudgetPages.qryBudgetPage.Delete ;
end;

procedure TfrmBudgetPage.mniBCanceRecAcClick(Sender: TObject);
begin
  if messagedlg('Are you sure?',mtWarning,[mbYes,mbNo],0) = mrYes then
    dmlBudgetPages.qryBudgetPage.RevertRecord ;
end;

procedure TfrmBudgetPage.mniBApplyAaClick(Sender: TObject);
begin
  with dmlBudgetPages, qryBudgetPage, dmlDBooks do
  begin
    dbsDBooks.StartTransaction ;
```

```
    try
      ApplyUpdates ;
      dbsDBooks.Commit ;
    except
      dbsDBooks.Rollback ;
      raise ;
    end ;
    CommitUpdates ;
    mniBSStandardAsClick(Sender);{Go back to standard view if not there}
    Close ;
    Open ;
    if not Closing then
      UpdateBudgetPageComboBox ;
  end ;
end;

procedure TfrmBudgetPage.mniBCancelAllAnClick(Sender: TObject);
begin
  dmlBudgetPages.qryBudgetPage.CancelUpdates ;
  mniBSStandardAsClick(Sender) ;{Go back to standard view if not there}
end;

procedure TfrmBudgetPage.mniBSStandardAsClick(Sender: TObject);
begin
  with dmlBudgetPages, qryBudgetPage do
    UpdateRecordTypes := [rtModified, rtInserted, rtUnmodified] ;
  mniBudgetPagesAe.Items[5].Items[0].Checked := true ;
  mniBudgetPagesAe.Items[2].Caption := '&Cancel Record Updates' ;
  dgrBudgetPages.Font.Color := clWindowText ;
end;

procedure TfrmBudgetPage.mniBSDeletedAdClick(Sender: TObject);
begin
  with dmlBudgetPages, qryBudgetPage do
    UpdateRecordTypes := [rtDeleted] ;
  with Sender as TMenuItem do Checked := true ;
  mniBudgetPagesAe.Items[2].Caption := '&Undelete BudgetPage' ;
  dgrBudgetPages.Font.Color := clRed ;
end;

procedure TfrmBudgetPage.mniBSEditedAeClick(Sender: TObject);
begin
  with dmlBudgetPages, qryBudgetPage do
    UpdateRecordTypes := [rtModified] ;
  with Sender as TMenuItem do Checked := true ;
```

```
  mniBudgetPagesAe.Items[2].Caption := '&Undo Changes' ;
  dgrBudgetPages.Font.Color := clMaroon ;
end;

procedure TfrmBudgetPage.mniBSInsertedAiClick(Sender: TObject);
begin
  with dmlBudgetPages, qryBudgetPage do
    UpdateRecordTypes := [rtInserted] ;
  with Sender as TMenuItem do Checked := true ;
  mniBudgetPagesAe.Items[2].Caption := '&Delete New BudgetPage' ;
  dgrBudgetPages.Font.Color := clNavy ;
end;

procedure TfrmBudgetPage.dngBudgetPageClick(Sender: TObject;
  Button: TNavigateBtn);
begin
  if Button = nbInsert then
  begin
    pgcBudgetPage.ActivePage := tbsDetails ;
    dedTitle.SetFocus ;
  end ;
end;

procedure TfrmBudgetPage.spbApprovalClick(Sender: TObject);
begin
  with dmlBudgetPages do
  begin
    cldBudgetPage.Date :=        {Initialize with current Date }
      FormatDateTime('mm/dd/yyyy',
                qryBudgetPageAPPROVAL_DATE.asDateTime) ;
    if cldBudgetPage.Execute then
    begin
      if not (qryBudgetPage.State in [dsEdit,dsInsert]) then
      begin
        qryBudgetPage.Edit ;
      end ;
      qryBudgetPageAPPROVAL_DATE.AsString := cldBudgetPage.Date ;
    end ;   {Store value into field}
  end;
end;

procedure TfrmBudgetPage.spbExpirationClick(Sender: TObject);
begin
  with dmlBudgetPages do
  begin
```

```
    cldBudgetPage.Date :=        {Initialize with current Date }
     FormatDateTime('mm/dd/yyyy',
                qryBudgetPageEXPIRATION_DATE.asDateTime) ;
    if cldBudgetPage.Execute then
    begin
     if not (qryBudgetPage.State in [dsEdit,dsInsert]) then
     begin
      qryBudgetPage.Edit ;
     end ;
     qryBudgetPageEXPIRATION_DATE.AsString := cldBudgetPage.Date ;
    end ;  {Store value into field}
  end;
end;

procedure TfrmBudgetPage.FormCreate(Sender: TObject);
begin
  pgcBudgetPage.ActivePage := tbsList ;
  Closing := false ;
  sesBudgetPage := TSession.Create(Self) ;
  sesBudgetPage.SessionName := 'sesBudgetPage1' ;
  dbsBudgetPage := TDatabase.Create(Self) ;
  with dbsBudgetPage do
  begin
    AliasName := 'DBooksOracle' ;
    DatabaseName := 'dbsBudgetPage1' ;
    LoginPrompt := false ;
    Params := dmlDbooks.dbsDbooks.params ;
    SessionName := sesBudgetPage.SessionName ;
  end ;
  qryBudgetPage := TQuery.Create(Self) ;
  with qryBudgetPage do
  begin
    SessionName := sesBudgetPage.SessionName ;
    DatabaseName := dbsBudgetPage.DatabaseName ;
    SQL.Clear ;
    SQL.Add('Select * from TBL_BUDGET_PAGE') ;
  end ;
end;

procedure TfrmBudgetPage.FormShow(Sender: TObject);
begin
  Left := 0 ;
  Top := 0 ;
  UpdateBudgetPageComboBox ;
end;
```

```
procedure TfrmBudgetPage.cboTitleChange(Sender: TObject);
begin
  dmlBudgetPages.qryBudgetPage.Locate('TITLE',cboTitle.Text,
    [loCaseInsensitive]);
end;

procedure TfrmBudgetPage.FormDestroy(Sender: TObject);
begin
  qryBudgetPage.Free ;
  dbsBudgetPage.Free ;
  sesBudgetPage.Free ;
end;

procedure TfrmBudgetPage.FormCloseQuery(Sender: TObject;
  var CanClose: Boolean);
begin
  Closing := true ;
  if dmlBudgetPages.qryBudgetPage.UpdatesPending then
    mniBApplyAaClick(Self) ;
  CanClose := true ;
end;

end.
```



File Name: Main.pas

```
{*************************************************************************
 D-Books 0.1 Prototype; Naval Postgraduate School      Begun 07/11/96
 Written by Rob Cameron and Ken Carrick

Module: Dbooks Web main form.
 Notes:
*************************************************************************}
unit main;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ADEMO3, WebBase, WebCore, WebSend, WebApp, htWebApp, tWebDemo,
WebTypes,
  WebVars, TpApplic, CGiVarS, APiStat, ApiBuilt, ApiCall, WebCall,
  WebBrows, HtmlBase, HtmlCore, HtmlSend, CGiServ, WebServ, UpdateOk,
  WebIniFL, WebInfo, Restorer, RestEdit, GridRest, IniLink, ebutton,
  Combobar, WebMemo, StdCtrls, TpMemo, Buttons, Toolbar, Grids, TxtGrid,
  ComCtrls, ExtCtrls, DB, Menus, TpMenu, WebMenu, DBTables, WdbLink,
  WdbScan, WdbGrid, WebLink, WdbSorce, WebPage, WebPHub, htDbWApp, weblist ;

type
  TfrmMain = class(ThtWebDemoForm)
    DoMenuBarFile: TtpDfmMenuItem;
    DoExit: TtpDfmMenuItem;
```

```
    DoMenuBarHelp: TtpDfmMenuItem;
    DoContents: TtpDfmMenuItem;
    DoTopicSearch: TtpDfmMenuItem;
    DoHowtouseHelp: TtpDfmMenuItem;
    DoTtpDfmMenuItem: TtpDfmMenuItem;
    DoAbout: TtpDfmMenuItem;
    DoActionComponents: TtpMenuItem;
    DoWebPages: TtpMenuItem;
    DoMenuItemView: TtpMenuItem;
    DoMenuItemTool: TtpMenuItem;
    DoMenuItemVerb: TtpMenuItem;
    DoMenuItemIdle: TtpMenuItem;
    DoMenuItemStatus: TtpMenuItem;
    WebMenu1: TWebMenu;
    DBooks_BUDGETPAGE: TWebPage;
    wdgBudgetPage: TWebDataGrid;
    wdsBudgetPage: TWebDataSource;
    wdsJobOrderDetails: TWebDataSource;
    DBooks_JOBORDERDETAILS: TWebPage;
    wdgEmployeeListing: TWebDataGrid;
    wdsEmployeeListing: TWebDataSource;
    DBooks_EMPLOYEELISTING: TWebPage;
    DBooks_VALIDATED: TWebPage;
    procedure wdgDBooksHotField(Sender: TWebDataScan; aField: TField;
      var s: string);
    procedure Table1AfterOpen(DataSet: TDataSet);
    procedure FormShow(Sender: TObject);
    procedure WebAppDbooksEventMacro(Sender: TWebOutputApp; const aMacro,
      aParams, aID: string);
    procedure wdgBudgetPageHotField(Sender: TWebDataScan; aField: TField;
      var s: string);
    procedure DBooks_JOBORDERDETAILSSection(Sender: TObject;
      Section: Integer; var Chunk, Options: string);
    procedure DBooks_EMPLOYEELISTINGSection(Sender: TObject;
      Section: Integer; var Chunk, Options: string);
    procedure DBooks_VALIDATEDSection(Sender: TObject; Section: Integer;
      var Chunk, Options: string);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmMain: TfrmMain;
```

```
implementation

uses dmdDbooksWeb, dmdBudgetPage, dmdJobOrder, dmdDepartment;

{$R *.DFM}

procedure TfrmMain.wdgDBooksHotField(Sender: TWebDataScan; aField: TField;
 var s: string);
begin
 inherited;
 if CompareText(aField.FieldName, 'Species Name') = 0 then
 begin
     s := '%=JUMP|detail,' + aField.DataSet.Fields[0].AsString + '|' +
         aField.DataSet.FieldByName('Species Name').AsString + '=%' ;
 end ;
end;

procedure TfrmMain.Table1AfterOpen(DataSet: TDataSet);
begin
 inherited;
  with TTable(Dataset) do
     FieldByName('Species Name').Tag := HotFieldTag ;

(*  Dataset['Species Name'].Tag := HotFieldTag ;   *)
end;

procedure TfrmMain.FormShow(Sender: TObject);
begin
 inherited;
   dmlDbooksWeb.OpenDM ;
end;

procedure TfrmMain.WebAppDbooksEventMacro(Sender: TWebOutputApp;
  const aMacro, aParams, aID: string);
  var
  sltDepartments :  TStringList ;
  strTemp: string ;
  wltTemp : TWebList ;
  begin
  inherited;
  if CompareText(aMacro,'gridbuttons') = 0 then
  begin
    wdsBudgetPage.Activate ;
```

```
 if wdsBudgetPage.DataSet.RecordCount < 5 then
   wdgBudgetPage.ButtonsWhere := dsbNone ;
end ;
 if CompareText(aMacro,'employeebuttons') = 0 then
begin
 wdsEmployeeListing.Activate ;
 WebAppDBooks.literal['RecordNo']:=inttostr(
wdsEmployeeListing.dataset.recordcount) ;
 if wdsEmployeeListing.DataSet.RecordCount < 5 then
   wdgEmployeeListing.ButtonsWhere := dsbNone ;
end ;
With WebAppDbooks.WebOutput do
begin
 if CompareText(aMacro,'getfield') = 0 then
 begin
   Send(dmlJobOrders.qryJobOrder.FieldByName(aParams).AsString) ;
 end ;
end ;
if comparetext (amacro, 'DeptSearch')=0 then
 begin
  sltDepartments := TStringlist.create() ;
  dmlDepartments.qryDepartment.first ;
  While not dmlDepartments.qryDepartment.eof do
  begin
    strTemp :=  dmlDepartments.qryDepartment.fieldbyname('Name').AsString + '=' ;
    strTemp :=  strTemp +
dmlDepartments.qryDepartment.fieldbyname('Department_ID').AsString ;
    sltDepartments.Add(strTemp) ;
    dmlDepartments.qryDepartment.next ;
  end ;
  wltTemp:=Tweblist.create(self);
  with webappoutput do
  begin
   filldropdown(wltTemp,sltDepartments,'','DeptSearch',
   '',ddmvalueasvalue);
   sendstringlist(wltTemp,false);
  end;
   wltTemp.free;
   sltDepartments.free ;
  end
end;

procedure TfrmMain.wdgBudgetPageHotField(Sender: TWebDataScan;
 aField: TField; var s: string);
begin
```

```
 inherited;
 if CompareText(aField.FieldName, 'Job_Order_id') = 0 then
 begin
     s := '%=JUMP|JobOrderDetails,' +
aField.DataSet.FieldByName('JOB_ORDER_ID').AsString + '|' +
         aField.DataSet.FieldByName('Job_Order_id').AsString + '=%' ;
 end ;
end;

procedure TfrmMain.DBooks_JOBORDERDETAILSSection(Sender: TObject;
  Section: Integer; var Chunk, Options: string);
begin
 inherited;
 With dmlJobOrders,qryJobOrder do
 begin
   Close ;
   Open ;
 end ;
end;

procedure TfrmMain.DBooks_EMPLOYEELISTINGSection(Sender: TObject;
  Section: Integer; var Chunk, Options: string);
begin
  inherited;
  with dmlDBooksWeb do
  begin
    qryemployeelisting.open ;
  end ;
end;

procedure TfrmMain.DBooks_VALIDATEDSection(Sender: TObject;
  Section: Integer; var Chunk, Options: string);
begin
  inherited;
  dmlDbooksWeb.qrylogin.close ;
  dmlDbooksWeb.qrylogin.open;
  if dmlDbooksWeb.qryLogin.recordcount=0 then
  begin
    WebAppoutput.send('%=errMessage=%') ;
    WebAppDBooks.weboutput.close ;
    exit ;
  end ;
 end;

 end.
```

APPENDIX D. DATA MODEL SPECIFICATION

```
CREATE DATABASE "F:\DATA\THESIS\D-Books32\Database\Interbase\D-
BOOKS32.GDB" PAGE_SIZE 4096
;


/* Table: TBL_ACTIVITY_GROUP, Owner: SYSDBA */
CREATE TABLE TBL_ACTIVITY_GROUP (ACTIVITY_GROUP_ID INTEGER
NOT NULL,
    ACTIVITY_GROUP VARCHAR(10),
PRIMARY KEY (ACTIVITY_GROUP_ID));

/* Table: TBL_BUDGET_PAGE, Owner: SYSDBA */
CREATE TABLE TBL_BUDGET_PAGE (BUDGET_PAGE_ID INTEGER NOT
NULL,
    TITLE VARCHAR(50),
    DEPARTMENT_ID SMALLINT,
    SPONSOR_ID INTEGER,
    PI_EMP_ID SMALLINT,
    APPROVAL_DATE DATE,
    EXPIRATION_DATE DATE,
    JOB_ORDER_ID INTEGER,
    LABOR_JOB_ORDER_ID INTEGER,
    FISCAL_YEAR INTEGER,
    NOTES BLOB SUB_TYPE TEXT SEGMENT SIZE 80,
    TOTAL_FACULTY_LABOR_COSTS DOUBLE PRECISION,
    TOTAL_SUPPORT_LABOR_COSTS DOUBLE PRECISION,
    TOTAL_OPTAR_COSTS DOUBLE PRECISION,
    TOTAL_CONTRACT_COSTS DOUBLE PRECISION,
    TOTAL_TRAVEL_COSTS DOUBLE PRECISION,
    FACULTY_LABOR_AUTHORIZED DOUBLE PRECISION,
    SUPPORT_LABOR_AUTHORIZED DOUBLE PRECISION,
    OPTAR_AUTHORIZED DOUBLE PRECISION,
    TRAVEL_AUTHORIZED DOUBLE PRECISION,
    CONTRACT_AUTHORIZED DOUBLE PRECISION,
    SERIAL_NUMBERS VARCHAR(12),
PRIMARY KEY (BUDGET_PAGE_ID));

/* Table: TBL_COST_CENTER, Owner: SYSDBA */
CREATE TABLE TBL_COST_CENTER (COST_CENTER_ID INTEGER NOT
NULL,
    COST_CENTER VARCHAR(10),
PRIMARY KEY (COST_CENTER_ID));
```

```
/* Table: TBL_DBOOKS32_DICTIONARY, Owner: SYSDBA */
CREATE TABLE TBL_DBOOKS32_DICTIONARY (OBJID INTEGER,
    VERSION SMALLINT,
    NAME VARCHAR(31),
    ALIASNAME VARCHAR(31),
    TYPEID INTEGER,
    MISCINFO1 INTEGER,
    MISCINFO2 INTEGER,
    SRCOBJID INTEGER,
    SRCVERSION SMALLINT,
    DESTOBJID INTEGER,
    DESTVERSION SMALLINT,
    OUTOFDATE VARCHAR(1),
    CREATEDATE DATE,
    LASTUPDATE DATE,
    INFOBLOB BLOB SUB_TYPE 0 SEGMENT SIZE 80,
    MISCINFO3 INTEGER);

/* Table: TBL_DEPARTMENT, Owner: SYSDBA */
CREATE TABLE TBL_DEPARTMENT (DEPARTMENT_ID SMALLINT NOT
NULL,
    NAME VARCHAR(30),
    DIST_CODE VARCHAR(6),
    MAIL_STOP VARCHAR(10),
    UIC VARCHAR(10),
    ORGANIZATIONAL_CODE VARCHAR(10),
    COST_CENTER_ID INTEGER,
    SUB_COST_CENTER_ID INTEGER,
    ACTIVITY_GROUP_ID INTEGER,
    SUB_ACTIVITY_GROUP_ID INTEGER,
PRIMARY KEY (DEPARTMENT_ID));

/* Table: TBL_DEPT_POC, Owner: SYSDBA */
CREATE TABLE TBL_DEPT_POC (DEPT_POC_ID SMALLINT NOT NULL,
    DEPARTMENT_ID SMALLINT,
    EMPLOYEE_ID SMALLINT,
PRIMARY KEY (DEPT_POC_ID));

/* Table: TBL_EMPLOYEE, Owner: SYSDBA */
CREATE TABLE TBL_EMPLOYEE (EMPLOYEE_ID SMALLINT NOT NULL,
    DEPARTMENT_ID SMALLINT,
    SSN VARCHAR(9),
    LAST_NAME VARCHAR(20),
    FIRST_NAME VARCHAR(15),
```

```
    MI VARCHAR(1),
    DATE_ONBOARD DATE,
    DATE_TERMINATED DATE,
    TITLE_ID SMALLINT,
    TYPE_ID SMALLINT,
    SERVICE_ID SMALLINT,
    STATUS_ID SMALLINT,
    REMARKS BLOB SUB_TYPE TEXT SEGMENT SIZE 80,
    TENURE_ID SMALLINT,
PRIMARY KEY (EMPLOYEE_ID));

/* Table: TBL_EMP_CONTACT_INFO, Owner: SYSDBA */
CREATE TABLE TBL_EMP_CONTACT_INFO (EMP_CONTACT_INFO_ID
SMALLINT NOT NULL,
    EMP_LOCATION_ID SMALLINT,
    CONTACT_TYPE_ID SMALLINT,
    PHONE_NUMBER_ VARCHAR(10),
    E_MAIL_ADDRESS VARCHAR(40),
    PRIORITY SMALLINT,
PRIMARY KEY (EMP_CONTACT_INFO_ID));

/* Table: TBL_EMP_CONTACT_TYPE, Owner: SYSDBA */
CREATE TABLE TBL_EMP_CONTACT_TYPE (CONTACT_TYPE_ID SMALLINT
NOT NULL,
    CONTACT_TYPE VARCHAR(20),
PRIMARY KEY (CONTACT_TYPE_ID));

/* Table: TBL_EMP_LOCATION, Owner: SYSDBA */
CREATE TABLE TBL_EMP_LOCATION (EMP_LOCATION_ID SMALLINT NOT
NULL,
    EMPLOYEE_ID INTEGER,
    EMP_LOCATION_TYPE_ID SMALLINT,
    PRIORITY SMALLINT,
    BLDG VARCHAR(12),
    ROOM VARCHAR(6),
    STREET_ADDRESS VARCHAR(20),
    CITY VARCHAR(15),
    STATE_ID VARCHAR(2),
    ZIP_CODE VARCHAR(10),
PRIMARY KEY (EMP_LOCATION_ID));

/* Table: TBL_EMP_LOCATION_TYPE, Owner: SYSDBA */
CREATE TABLE TBL_EMP_LOCATION_TYPE (LOCATION_TYPE_ID
SMALLINT NOT NULL,
    LOCATION VARCHAR(20),
```

```
PRIMARY KEY (LOCATION_TYPE_ID));

/* Table: TBL_EMP_PAY_HISTORY, Owner: SYSDBA */
CREATE TABLE TBL_EMP_PAY_HISTORY (EMP_PAY_HISTORY_ID
SMALLINT NOT NULL,
    EMPLOYEE_ID SMALLINT,
    OCCUPATIONAL_CODE_ID SMALLINT,
    BEGIN_DATE DATE,
    END_DATE DATE,
    HOURLY_RATE DOUBLE PRECISION,
    DAILY_RATE DOUBLE PRECISION,
    ANNUAL_RATE DOUBLE PRECISION,
    OVERTIME_RATE DOUBLE PRECISION,
    REGULAR_ACCEL_RATE DOUBLE PRECISION,
    OVERTIME_ACCEL_RATE DOUBLE PRECISION,
    REMARKS BLOB SUB_TYPE TEXT SEGMENT SIZE 80,
    GRADE_ID INTEGER,
    STEP_ID INTEGER,
PRIMARY KEY (EMP_PAY_HISTORY_ID));

/* Table: TBL_EMP_STATUS, Owner: SYSDBA */
CREATE TABLE TBL_EMP_STATUS (EMP_STATUS_ID SMALLINT NOT NULL,
    STATUS VARCHAR(15),
PRIMARY KEY (EMP_STATUS_ID));

/* Table: TBL_EMP_TITLE, Owner: SYSDBA */
CREATE TABLE TBL_EMP_TITLE (EMP_TITLE_ID SMALLINT NOT NULL,
    TITLE VARCHAR(30),
PRIMARY KEY (EMP_TITLE_ID));

/* Table: TBL_EMP_TYPE, Owner: SYSDBA */
CREATE TABLE TBL_EMP_TYPE (EMP_TYPE_ID SMALLINT NOT NULL,
    TYPE VARCHAR(12),
PRIMARY KEY (EMP_TYPE_ID));

/* Table: TBL_FUND, Owner: SYSDBA */
CREATE TABLE TBL_FUND (FUND_ID SMALLINT NOT NULL,
    NAME VARCHAR(20),
    AUTHORIZATION VARCHAR(20),
    BEGIN_DATE DATE,
    END_DATE DATE,
    TYPE_ID SMALLINT,
    STATUS_ID SMALLINT,
    INITIAL_BALANCE DOUBLE PRECISION,
    TOTAL_CHARGES DOUBLE PRECISION,
```

```
PRIMARY KEY (FUND_ID));

/* Table: TBL_FUND_STATUS, Owner: SYSDBA */
CREATE TABLE TBL_FUND_STATUS (FUND_STATUS_ID SMALLINT NOT
NULL,
    STATUS_TEXT VARCHAR(20),
PRIMARY KEY (FUND_STATUS_ID));

/* Table: TBL_FUND_TYPE, Owner: SYSDBA */
CREATE TABLE TBL_FUND_TYPE (FUND_TYPE_ID SMALLINT NOT NULL,
    TYPE VARCHAR(15),
PRIMARY KEY (FUND_TYPE_ID));

/* Table: TBL_GRADE, Owner: SYSDBA */
CREATE TABLE TBL_GRADE (GRADE_ID INTEGER NOT NULL,
    GRADE VARCHAR(8),
PRIMARY KEY (GRADE_ID));

/* Table: TBL_HOURS_TYPE, Owner: SYSDBA */
CREATE TABLE TBL_HOURS_TYPE (HRS_TYPE_ID SMALLINT NOT NULL,
    HOURS_TYPE VARCHAR(10),
PRIMARY KEY (HRS_TYPE_ID));

/* Table: TBL_JOB_ORDER, Owner: SYSDBA */
CREATE TABLE TBL_JOB_ORDER (JOB_ORDER_ID INTEGER NOT NULL,
    STATUS_ID SMALLINT,
    TYPE_ID SMALLINT,
    FUND_ID INTEGER,
    JOB_ORDER_NUMBER VARCHAR(20),
    EXPIRATION_DATE DATE,
    TOTAL_CHARGES DOUBLE PRECISION,
    INITIAL_BALANCE DOUBLE PRECISION,
PRIMARY KEY (JOB_ORDER_ID));

/* Table: TBL_JON_STATUS, Owner: SYSDBA */
CREATE TABLE TBL_JON_STATUS (JON_STATUS_ID SMALLINT NOT NULL,
    STATUS VARCHAR(20),
PRIMARY KEY (JON_STATUS_ID));

/* Table: TBL_JON_TYPE, Owner: SYSDBA */
CREATE TABLE TBL_JON_TYPE (JON_TYPE_ID SMALLINT NOT NULL,
    TYPE VARCHAR(20),
PRIMARY KEY (JON_TYPE_ID));

/* Table: TBL_LABOR_RECORD, Owner: SYSDBA */
```

```
CREATE TABLE TBL_LABOR_RECORD (LABOR_RECORD_ID INTEGER NOT
NULL,
    TIMECARD_ID INTEGER,
    JON_ID INTEGER,
    HRS_TYPE_ID SMALLINT,
    NUMBER_OF_HOURS FLOAT,
PRIMARY KEY (LABOR_RECORD_ID));

/* Table: TBL_OCCUPATIONAL_CODE, Owner: SYSDBA */
CREATE TABLE TBL_OCCUPATIONAL_CODE (OC_CODE_ID SMALLINT NOT
NULL,
    CODE VARCHAR(10),
PRIMARY KEY (OC_CODE_ID));

/* Table: TBL_SERVICE, Owner: SYSDBA */
CREATE TABLE TBL_SERVICE (SERVICE_ID SMALLINT NOT NULL,
    SERVICE VARCHAR(10),
PRIMARY KEY (SERVICE_ID));

/* Table: TBL_SPONSOR, Owner: SYSDBA */
CREATE TABLE TBL_SPONSOR (SPONSOR_ID INTEGER NOT NULL,
    NAME VARCHAR(30),
    TYPE_ID SMALLINT,
    STATUS_ID SMALLINT,
PRIMARY KEY (SPONSOR_ID));

/* Table: TBL_SPONSOR_STATUS, Owner: SYSDBA */
CREATE TABLE TBL_SPONSOR_STATUS (SPONSOR_STATUS_ID SMALLINT
NOT NULL,
    STATUS VARCHAR(20),
PRIMARY KEY (SPONSOR_STATUS_ID));

/* Table: TBL_SPONSOR_TYPE, Owner: SYSDBA */
CREATE TABLE TBL_SPONSOR_TYPE (SPONSOR_TYPE_ID SMALLINT NOT
NULL,
    TYPE VARCHAR(20),
PRIMARY KEY (SPONSOR_TYPE_ID));

/* Table: TBL_STATE, Owner: SYSDBA */
CREATE TABLE TBL_STATE (STATE_ID VARCHAR(2) NOT NULL,
    STATE VARCHAR(30),
PRIMARY KEY (STATE_ID));

/* Table: TBL_STEP, Owner: SYSDBA */
CREATE TABLE TBL_STEP (STEP_ID INTEGER NOT NULL,
```

```
    STEP VARCHAR(8),
PRIMARY KEY (STEP_ID));

/* Table: TBL_SUB_ACTIVITY_GROUP, Owner: SYSDBA */
CREATE TABLE TBL_SUB_ACTIVITY_GROUP (SUB_ACTIVITY_GROUP_ID
INTEGER NOT NULL,
    SUB_ACTIVITY_GROUP VARCHAR(10),
PRIMARY KEY (SUB_ACTIVITY_GROUP_ID));

/* Table: TBL_SUB_COST_CENTER, Owner: SYSDBA */
CREATE TABLE TBL_SUB_COST_CENTER (SUB_COST_CENTER_ID INTEGER
NOT NULL,
    SUB_COST_CENTER VARCHAR(10),
PRIMARY KEY (SUB_COST_CENTER_ID));

/* Table: TBL_TENURE, Owner: SYSDBA */
CREATE TABLE TBL_TENURE (TENURE_ID SMALLINT NOT NULL,
    TENURE VARCHAR(11),
PRIMARY KEY (TENURE_ID));

/* Table: TBL_TIMECARD, Owner: SYSDBA */
CREATE TABLE TBL_TIMECARD (TIMECARD_ID INTEGER NOT NULL,
    EMP_PAY_HIST_ID INTEGER,
    BEGIN_PAY_DATE DATE,
    END_PAY_DATE DATE,
    NOTES BLOB SUB_TYPE TEXT SEGMENT SIZE 80,
PRIMARY KEY (TIMECARD_ID));

/* Index definitions for all user tables */
CREATE INDEX OALIASNAME ON TBL_DBOOKS32_DICTIONARY(TYPEID,
ALIASNAME);
CREATE UNIQUE INDEX OBJID ON TBL_DBOOKS32_DICTIONARY(OBJID,
VERSION);
CREATE INDEX ONAME ON TBL_DBOOKS32_DICTIONARY(TYPEID, NAME);
CREATE INDEX RELDEST ON TBL_DBOOKS32_DICTIONARY(DESTOBJID,
DESTVERSION, TYPEID, SRCOBJID, SRCVERSION);
CREATE INDEX RELSRC ON TBL_DBOOKS32_DICTIONARY(SRCOBJID,
SRCVERSION, TYPEID, DESTOBJID, DESTVERSION);
CREATE INDEX TYPEID ON TBL_DBOOKS32_DICTIONARY(TYPEID);
ALTER TABLE TBL_EMPLOYEE ADD FOREIGN KEY (DEPARTMENT_ID)
REFERENCES TBL_DEPARTMENT(DEPARTMENT_ID);
ALTER TABLE TBL_EMPLOYEE ADD FOREIGN KEY (TITLE_ID) REFERENCES
TBL_EMP_TITLE(EMP_TITLE_ID);
ALTER TABLE TBL_EMPLOYEE ADD FOREIGN KEY (TYPE_ID) REFERENCES
TBL_EMP_TYPE(EMP_TYPE_ID);
```

```
ALTER TABLE TBL_EMPLOYEE ADD FOREIGN KEY (SERVICE_ID)
REFERENCES TBL_SERVICE(SERVICE_ID);
ALTER TABLE TBL_EMPLOYEE ADD FOREIGN KEY (STATUS_ID)
REFERENCES TBL_EMP_STATUS(EMP_STATUS_ID);
ALTER TABLE TBL_DEPT_POC ADD FOREIGN KEY (DEPARTMENT_ID)
REFERENCES TBL_DEPARTMENT(DEPARTMENT_ID);
ALTER TABLE TBL_DEPT_POC ADD FOREIGN KEY (EMPLOYEE_ID)
REFERENCES TBL_EMPLOYEE(EMPLOYEE_ID);
ALTER TABLE TBL_EMP_LOCATION ADD FOREIGN KEY (EMPLOYEE_ID)
REFERENCES TBL_EMPLOYEE(EMPLOYEE_ID);
ALTER TABLE TBL_EMP_LOCATION ADD FOREIGN KEY
(EMP_LOCATION_TYPE_ID) REFERENCES
TBL_EMP_LOCATION_TYPE(LOCATION_TYPE_ID);
ALTER TABLE TBL_EMP_CONTACT_INFO ADD FOREIGN KEY
(EMP_LOCATION_ID) REFERENCES
TBL_EMP_LOCATION(EMP_LOCATION_ID);
ALTER TABLE TBL_EMP_CONTACT_INFO ADD FOREIGN KEY
(CONTACT_TYPE_ID) REFERENCES
TBL_EMP_CONTACT_TYPE(CONTACT_TYPE_ID);
ALTER TABLE TBL_EMP_PAY_HISTORY ADD FOREIGN KEY (EMPLOYEE_ID)
REFERENCES TBL_EMPLOYEE(EMPLOYEE_ID);
ALTER TABLE TBL_EMP_PAY_HISTORY ADD FOREIGN KEY
(OCCUPATIONAL_CODE_ID) REFERENCES
TBL_OCCUPATIONAL_CODE(OC_CODE_ID);
ALTER TABLE TBL_TIMECARD ADD FOREIGN KEY (EMP_PAY_HIST_ID)
REFERENCES TBL_EMP_PAY_HISTORY(EMP_PAY_HISTORY_ID);
ALTER TABLE TBL_FUND ADD FOREIGN KEY (TYPE_ID) REFERENCES
TBL_FUND_TYPE(FUND_TYPE_ID);
ALTER TABLE TBL_FUND ADD FOREIGN KEY (STATUS_ID) REFERENCES
TBL_FUND_STATUS(FUND_STATUS_ID);
ALTER TABLE TBL_JOB_ORDER ADD FOREIGN KEY (TYPE_ID) REFERENCES
TBL_JON_TYPE(JON_TYPE_ID);
ALTER TABLE TBL_JOB_ORDER ADD FOREIGN KEY (STATUS_ID)
REFERENCES TBL_JON_STATUS(JON_STATUS_ID);
ALTER TABLE TBL_JOB_ORDER ADD FOREIGN KEY (FUND_ID) REFERENCES
TBL_FUND(FUND_ID);
ALTER TABLE TBL_LABOR_RECORD ADD FOREIGN KEY (TIMECARD_ID)
REFERENCES TBL_TIMECARD(TIMECARD_ID);
ALTER TABLE TBL_LABOR_RECORD ADD FOREIGN KEY (JON_ID)
REFERENCES TBL_JOB_ORDER(JOB_ORDER_ID);
ALTER TABLE TBL_LABOR_RECORD ADD FOREIGN KEY (HRS_TYPE_ID)
REFERENCES TBL_HOURS_TYPE(HRS_TYPE_ID);
ALTER TABLE TBL_SPONSOR ADD FOREIGN KEY (TYPE_ID) REFERENCES
TBL_SPONSOR_TYPE(SPONSOR_TYPE_ID);
```

```
ALTER TABLE TBL_SPONSOR ADD FOREIGN KEY (STATUS_ID) REFERENCES
TBL_SPONSOR_STATUS(SPONSOR_STATUS_ID);
ALTER TABLE TBL_BUDGET_PAGE ADD FOREIGN KEY (DEPARTMENT_ID)
REFERENCES TBL_DEPARTMENT(DEPARTMENT_ID);
ALTER TABLE TBL_BUDGET_PAGE ADD FOREIGN KEY (SPONSOR_ID)
REFERENCES TBL_SPONSOR(SPONSOR_ID);
ALTER TABLE TBL_BUDGET_PAGE ADD FOREIGN KEY (PI_EMP_ID)
REFERENCES TBL_EMPLOYEE(EMPLOYEE_ID);
ALTER TABLE TBL_BUDGET_PAGE ADD FOREIGN KEY (JOB_ORDER_ID)
REFERENCES TBL_JOB_ORDER(JOB_ORDER_ID);
ALTER TABLE TBL_BUDGET_PAGE ADD FOREIGN KEY
(LABOR_JOB_ORDER_ID) REFERENCES TBL_JOB_ORDER(JOB_ORDER_ID);
ALTER TABLE TBL_EMPLOYEE ADD FOREIGN KEY (TENURE_ID)
REFERENCES TBL_TENURE(TENURE_ID);
ALTER TABLE TBL_EMP_PAY_HISTORY ADD FOREIGN KEY (GRADE_ID)
REFERENCES TBL_GRADE(GRADE_ID);
ALTER TABLE TBL_EMP_PAY_HISTORY ADD FOREIGN KEY (STEP_ID)
REFERENCES TBL_STEP(STEP_ID);
ALTER TABLE TBL_DEPARTMENT ADD FOREIGN KEY
(ACTIVITY_GROUP_ID) REFERENCES
TBL_ACTIVITY_GROUP(ACTIVITY_GROUP_ID);
ALTER TABLE TBL_DEPARTMENT ADD FOREIGN KEY
(SUB_ACTIVITY_GROUP_ID) REFERENCES
TBL_SUB_ACTIVITY_GROUP(SUB_ACTIVITY_GROUP_ID);
ALTER TABLE TBL_DEPARTMENT ADD FOREIGN KEY (COST_CENTER_ID)
REFERENCES TBL_COST_CENTER(COST_CENTER_ID);
ALTER TABLE TBL_DEPARTMENT ADD FOREIGN KEY
(SUB_COST_CENTER_ID) REFERENCES
TBL_SUB_COST_CENTER(SUB_COST_CENTER_ID);

CREATE GENERATOR DEPARTMENT_ID_GEN;
CREATE GENERATOR EMPLOYEE_ID_GEN;
CREATE GENERATOR BUDGET_PAGE_ID_GEN;
CREATE GENERATOR DEPT_POC_ID_GEN;
CREATE GENERATOR EMPLOYEE_LOCATION_ID_GEN;
CREATE GENERATOR EMPLOYEE_CONTACT_INFO_ID_GEN;
CREATE GENERATOR FUND_ID_GEN;
CREATE GENERATOR JOB_ORDER_ID_GEN;
CREATE GENERATOR SPONSOR_ID_GEN;
CREATE GENERATOR SCREENING_CAND_ID_GEN;
CREATE GENERATOR EMP_PAY_HIST_GEN;
CREATE GENERATOR TIMECARD_ID_GEN;
CREATE GENERATOR LABOR_RECORD_GEN;

COMMIT WORK;
```

```
SET AUTODDL OFF;
SET TERM ^ ;

/* Stored procedures */
CREATE PROCEDURE GET_NEXT_TIMECARD_ID AS BEGIN EXIT; END ^

ALTER PROCEDURE GET_NEXT_TIMECARD_ID RETURNS
(NEW_TIMECARD_ID INTEGER)
AS

BEGIN
  NEW_TIMECARD_ID = GEN_ID(TIMECARD_ID_GEN,1) ;
END
 ^
SET TERM ; ^
COMMIT WORK ;
SET AUTODDL ON;
SET TERM ^ ;

/* Triggers only will work for SQL triggers */
CREATE TRIGGER CREATE_EMPLOYEE_CONTACT_INFO_ID FOR
TBL_EMP_CONTACT_INFO
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  NEW.EMP_CONTACT_INFO_ID = GEN_ID
(EMPLOYEE_CONTACT_INFO_ID_GEN, 1) ;
END
 ^
CREATE TRIGGER CREATE_EMPLOYEE_ID FOR TBL_EMPLOYEE
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  NEW.EMPLOYEE_ID = GEN_ID (EMPLOYEE_ID_GEN, 1) ;
END
 ^
CREATE TRIGGER CREATE_DEPARTMENT_ID FOR TBL_DEPARTMENT
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  NEW.DEPARTMENT_ID = GEN_ID (DEPARTMENT_ID_GEN, 1) ;
END
 ^
CREATE TRIGGER CREATE_BUDGET_PAGE_ID FOR TBL_BUDGET_PAGE
ACTIVE BEFORE INSERT POSITION 0
```

```
AS
BEGIN
  NEW.BUDGET_PAGE_ID = GEN_ID (BUDGET_PAGE_ID_GEN, 1) ;
END
 ^
CREATE TRIGGER CREATE_DEPT_POC_ID FOR TBL_DEPT_POC
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  NEW.DEPT_POC_ID = GEN_ID (DEPT_POC_ID_GEN, 1) ;
END
 ^
CREATE TRIGGER CREATE_EMPLOYEE_LOCATION_ID FOR
TBL_EMP_LOCATION
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  NEW.EMP_LOCATION_ID = GEN_ID (EMPLOYEE_LOCATION_ID_GEN, 1) ;
END
 ^
CREATE TRIGGER CREATE_FUND_ID FOR TBL_FUND
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  NEW.FUND_ID = GEN_ID (FUND_ID_GEN, 1) ;
END
 ^
CREATE TRIGGER CREATE_JOB_ORDER_ID FOR TBL_JOB_ORDER
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  NEW.JOB_ORDER_ID = GEN_ID (JOB_ORDER_ID_GEN, 1) ;
END
 ^
CREATE TRIGGER CREATE_SPONSOR_ID FOR TBL_SPONSOR
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  NEW.SPONSOR_ID = GEN_ID (SPONSOR_ID_GEN, 1) ;
END
 ^
CREATE TRIGGER CREATE_EMP_PAY_HIST FOR TBL_EMP_PAY_HISTORY
ACTIVE BEFORE INSERT POSITION 0
AS BEGIN
  NEW.EMP_PAY_HISTORY_ID = GEN_ID (EMP_PAY_HIST_GEN, 1) ;
```

```
END
^
CREATE TRIGGER CREATE_LABOR_RECORD_ID FOR TBL_LABOR_RECORD
ACTIVE BEFORE INSERT POSITION 0
AS BEGIN
                NEW.LABOR_RECORD_ID = GEN_ID (LABOR_RECORD_GEN, 1);
        END
^
COMMIT WORK ^
SET TERM ; ^
```

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5000

3. Professor James C. Emery, Code 05 2
   Naval Postgraduate School
   Monterey, CA 93943-5000

4. Professor Barry Frew, Code SM/Fw 1
   Naval Postgraduate School
   Monterey, CA 93943-5000

5. Judy Harr, Code 01B4 1
   Naval Postgraduate School
   Monterey, CA 93943-5000

6. LCDR Dale Courtney, Code 05 1
   Naval Postgraduate School
   Monterey, CA 93943-5000

7. Mr. R. Jay, Code 21 1
   Naval Postgraduate School
   Monterey, CA 93943-5000

8. Megan Reilly, Code 21D 1
   Naval Postgraduate School
   Monterey, CA 93943-5000

9. Michael Nichols, Code 2231 1
   Naval Postgraduate School
   Monterey, CA 93943-5000

10. LT Warren Yu 1
    1275 Leahy Roadl
    Monterey, CA 93940-4872

11. LT Robert A. Cameron 2
    3711 Azalea Drive
    Philadelphia, PA 19136

12. CPT Kenneth G. Carrick 2
    331 Metz Road
    Seaside, CA 93955