NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AUTOMATING SUBMARINE TRAINING AND READINESS USING WEB-ENABLED APPLICATIONS

by

Kevin S. Anderson
David J. Murphy

June 2001

Thesis Principal Advisor: William J. Haga
Thesis Associate Advisor: Douglas E. Brinkley

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2001	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Automating Submarine Training and Readiness Using Web-Enabled Applications			**5. FUNDING NUMBERS**
6. AUTHOR(S) Anderson, Kevin S. and Murphy, David J.			
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) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author 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)***

This thesis examines the limits to on-board training and readiness imposed upon the submarine community by manual data collection and record systems.

It proposes an integration of web-based applications under the Balanced Scorecard management approach as a solution. Specifications are included for fiber optic LAN Infrastructure design, network client workstations, network servers and IT-21 compliant network operating systems, producing applications and web augmentation tools. A description of relational database design is provided that encompasses database objects, data types and table relationships. Web design issues and network security issues are examined. Complete code for a prototype of the system in Microsoft Access is appended.

Recommendations are made for using an n-tier network architecture to automate the submarine training and readiness processes in small project modules using an incremental Prototype development approach. An n-tier architecture supports custom applications and commercial-off-the-shelf (COTS) components, supports component reuse and encapsulation, provides a consistent, secure and auditable access to data and will allow faster production at less risk. Recommendations are made for using Microsoft products that support n-tier architecture and are consistent with the Navy's IT-21 standards.

14. SUBJECT TERMS Automation, Local Area Network, n-Tier, Readiness, Submarine, Training, Web-Based, Web-Enabled			**15. NUMBER OF PAGES** 188
			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

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

AUTOMATING SUBMARINE TRAINING AND READINESS USING WEB-ENABLED APPLICATIONS

Kevin S. Anderson
Lieutenant, United States Navy
B.S., Jacksonville University, 1994

David J. Murphy
Major, United States Marine Corps
B.S., Syracuse University, 1985

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
June 2001

Authors: _____
Kevin S. Anderson

_____
David J. Murphy

Approved by: _____
William J. Haga, Principal Advisor

_____
Douglas E. Brinkley, Associate Advisor

_____
Dan Boger, Chairman
Information Systems Academic Group

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis examines the limits to on-board training and readiness imposed upon the submarine community by manual data collection and record systems.

It proposes an integration of web-based applications under the Balanced Scorecard management approach as a solution. Specifications are included for fiber optic LAN Infrastructure design, network client workstations, network servers and IT-21 compliant network operating systems, producing applications and web augmentation tools. A description of relational database design is provided that encompasses database objects, data types and table relationships. Web design issues and network security issues are examined. Complete code for a prototype of the system in Microsoft Access is appended.

Recommendations are made for using an n-tier network architecture to automate the submarine training and readiness processes in small project modules using an incremental Prototype development approach. An n-tier architecture supports custom applications and commercial-off-the-shelf (COTS) components, supports component reuse and encapsulation, provides a consistent, secure and auditable access to data and will allow faster production at less risk. Recommendations are made for using Microsoft products that support n-tier architecture and are consistent with the Navy's IT-21 standards.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF ACRONYMS

BPR	Business Process Re-engineering
CD-ROM	Compact Disk Read-Only Memory
CJCS	Chairman of the Joint Chiefs of Staff
CO	Commanding Officer
DBMS	Database Management System
DDBMS	Distributed Database Management System
DH	Department Head
DO	Division Officer
DVD	Digital Video Disk
ERP	Enterprise Resource Planning
HTTP	Hypertext Transfer Protocol
IT-21	Information Technology for the 21st Century
JO	Junior Officer
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LRTP	Long Range Training Plan
MII	Mitre Information Infrastructure
NTFS	Network File System
NUWC	Naval Undersea Warfare Center
POD	Plan of the Day
POW	Plan of the Week

PRP	Personnel Reliability Program
QPO	Qualified Petty Officer
SAT	Systems Approach to Training
SOBT	School Of the Boat Training
SRTP	Short Range Training Plan
SSL	Secure Socket Layer
TDU	Trash Disposal Unit
WAN	Wide Area Network
Windows NT	Windows New Technology
WTR	Weekly Training Report
XML	Extensible Markup Language
XO	Executive Officer

I. INTRODUCTION

A. THE PROBLEM

Submarines are equipped with the IT infrastructure to leverage information superiority as mandated by our leaders, but until now they have failed to do so. Currently, computers are used as electronic typewriters and not as portals to information superiority. Training and readiness data is often hand recorded or stored on personal hard drives thus limiting access to the information making data correlation and trend analysis impossible. In addition, real-time status of the ships training and readiness posture is not available to those who make the decisions. Additional time is also required to collect, tabulate and format the data for presentation to inspection teams during externally generated command inspections. The authors believe that the current methods are inefficient, are filled with redundant efforts, fail to identify significant and/or recurring deficiencies and fail to capture or identify all sources of information available for decision making.

B. PROPOSED SOLUTION AND VISION

The authors propose that submarines should capitalize upon existing information technology infrastructure by integrating web-based applications to efficiently capture, manipulate, correlate, store, share and present information related to training and readiness so that personnel requiring access to this data have it available to them when and where they need it. Furthermore, they hope to demonstrate how web-based applications would enable submarine crews to save time, eliminate redundancy, improve commander's decision-making capabilities, provide wider dissemination of information, correlate results between the various information sources and provide real-time information and trend analysis to decision makers. Integrating these current technologies

into existing infrastructure will increase efficiency throughout the chain of command and enable submarine crews to operate at higher levels of combat readiness.

C. CONSEQUENCES OF INACTION

By not taking full advantage of existing information technologies, the submarine force will fail to maintain its competitive advantage through information superiority as outlined by the Chairman of the Joint Chiefs of Staff (CJCS - 2000) in Joint Vision 2020. The CJCS (2000) advocates that our military advantage through technology infrastructure will decline in the future and that our best competitive advantage will effectively be in the form of an imbalance in our favor by successfully translating information into superior knowledge and decisions. Furthermore, failing to properly utilize their information resources, the submarine force could deny themselves the benefits realized in the Pacific Fleet as testified by Admiral Clemins (1999) which include: increased productivity by reducing the time required to complete tasks, doing the same or more work with less personnel, significantly eliminating redundancy in information management processes and finally, increased troop morale which ultimately translates to better retention and recruitment. The authors believe that web-based information technologies can result in a reduction of incidents and accidents at sea, better trained crews with higher levels of knowledge and more efficient work processes as a result of enhanced training programs provided by real-time information, situational trend analysis and increased access to information for all crewmembers.

D. BACKGROUND

The Joint Chiefs of Staff warn in Joint Vision 2020 (2000) that the increased availability of commercial satellites, digital communications and the public Internet will give adversaries new access and capabilities to the global commercial industrial base and

much of the same technology as the US military at a relatively low cost. The United States therefore, will not necessarily sustain a wide technological advantage over our adversaries. The advantage must come from people, leadership, doctrine, organizations and training that enable us to take advantage of technology to achieve superior warfighting effectiveness. Admiral Clemins (1999) testified before the House Armed Services Committee that the Navy's vision has moved beyond "Network-Centric Warfare" to that of a "Knowledge-Centric" environment. His vision is that people will conduct both warfighting and business missions in a virtual environment via a common tabletop using a web browser interface and other technologies like data mining, data warehousing and relational databases to share information and IT applications. The Navy's goals are to provide all users rapid and secure access to the same information in order to improve existing operational processes of command and control.

E. METHODOLOGY

This research effort includes a literature review of books, periodicals and other Internet resources that relate to information systems, the Navy's policies and vision regarding information systems, web-based technologies and intranet topologies. Benefits of using integrated web-based information technologies are documented and specific information processes, methodologies and application tools used onboard submarines in support of training and readiness are identified. The authors relied upon personal knowledge as well as that from leaders in the fleet when examining current submarine information technologies and identifying those processes that should be included in a standardized web-enabled application.

F. KEY ASSUMPTIONS BEHIND THIS THESIS

1. All submarines currently are or will soon be equipped with the SNAP III intranet using a fiber optic backbone or an equivalent infrastructure.
2. Shipboard intranets are configured with client workstations and servers consisting of hardware and software meeting or exceeding those prescribed by the Navy's IT-21 standards.

II. LITERATURE REVIEW

A literature review of current and previous organizational professional research on web-based application tools that may benefit the submarine force was conducted. Criteria focused on Department of Defense (DoD) and industry technologies that would save time as well as improve commanders' decision-making processes. Defense and industry sources researched include journals, texts, Internet resources as well as NPS faculty.

Training is a key requirement for achieving combat readiness. Having sailors and Marines equipped with the right skills when they go aboard ship is absolutely key to fleet readiness. Currently, the U.S. Navy's submarine force does not use a standardized, automated tool for recording and correlating the results between the various evaluation methods to provide timely trend analysis to the chain of command in support of it's Training and Readiness (T&R) requirements. The DoD and industry are information driven. The need to access and analyze organizations own information in real time, update databases, perform trend analysis and operate in a 24 x 7 environment is changing the demands placed on information technology (IT) infrastructures. Quality of service, scalability and availability demands are also pushing the limits of existing IT infrastructures. Military organizations in particular need to retool and take advantage of flexible n-tier architecture. With the ability to scale both horizontally and vertically, n-tier architectures can increase the flexibility of the DoD's IT infrastructure. (Sun Microsystems, Inc. 2001)

Ten years after its introduction, the client/server application architecture has toppled the mainframe to become the application architecture of choice. Tiers are used to describe the logical partitioning of an application across clients and servers. (Edwards, Jeri 1999) A tier is a functionally separated hardware and software component that performs a specific function. Communications between tiers is accomplished through standard protocols such as HTTP, RMI and XML. (Sun Microsystems, Inc., 2001)

The traditional client/server is a 2-tier architecture because it splits the processing load between a server and a client computer, but the majority of the application logic runs on the client. Gould (2000) explains that in the majority of 2-tier systems, the only service provided by the server is that of a database server. The client is then responsible for data access, applying business logic, converting the results into a format suitable for display and accepting user input and displaying the intended interface to the user in a graphical user interface (GUI). This architecture is called *fat client* because the biggest chunk of the application runs on the client side of the equation. (Edwards, Jeri 1999)

Jeri Edwards, (1999) states that while 2-tier architectures are simple and great for creating applications quickly with visual builder tools, they are not good for mission-critical applications because they are not scalable. Additionally, because the application logic is buried inside the user interface on the client, the client must know how the data are organized and stored on the server side in order to access the data. Tomsen (1998) states that because fat clients contain the business logic for the application, they have a large footprint; each client requires a separate database connection since the client contains the application and the sever hosts the data. Multiple users mean multiple

connections and this degrades the performance and scalability of the application. Scaling is therefore expensive for both server resources and your budget with 2-tier architectures. Sun Microsystems, Inc. (2001) notes that while the use of fewer nodes by 2-tier architectures simplifies manageability, change management is difficult because it requires servers to be taken offline for repair, upgrades, and new applications deployments. In addition, deploying new applications and enhancements in fat-client environments is complex and time consuming, resulting in increased risk and reduced availability. Lastly, Gould (2000) reports that while the client/server architecture is generally easy to deploy at first, they are difficult to upgrade or enhance. They also make reuse of business and presentation logic difficult, if not impossible. Most importantly, because 2-tier applications are typically not very scalable, they are not well suited to the Internet.

Jeri Edwards, (1999) explains that as applications become more complex, 2-tier applications become exponentially harder to develop. 3-tier architecture has therefore become the new growth area for client/server computing because it meets the requirements of large-scale Internet and intranet applications and they are easier to manage and deploy on the network. The 3-tier architecture splits the application logic from the GUI and the database into a middle tier. The 3-tier architecture processing load is therefore split between 1) clients that run the GUI logic (called the front-end), 2) the application server running the business logic and 3) the database (called the back-end). Because 3-tier moves the application logic to the server, it is also referred to as a *fat server* or a *thin client* architecture.

One benefit of the 3-tier architecture over 2-tier is that it provides better security by not exposing the database schema to the client and by enabling more fine-grained

authorization on the server (Jeri Edwards, 1999). Tomsen (1998) explains that 3-tier applications improve scalability by moving the business logic out of the client and onto a separate tier. The 3-tier architecture clients are thin clients and only need to contain the logic necessary to call the business logic appropriately. Thin clients can be browsers or standard applications. Modifying an application under the 3-tier architecture only requires component modification in the middle tier, therefore you do not have to change and redeploy your clients unless you change the GUI. Coleman-Martin (1998) lists the following benefits of the 3-tier architecture:

- Three-tier is at least as scalable as the client-server model.
- Three-tier system management is more centralized, therefore less costly, than client-server management.
- The three-tier model incorporates legacy systems in the data tier, which allows the legacy investment to be recaptured.
- Three-tier client cost (hardware, software, and maintenance) is lower than that of the client-server client.
- Industry focus is on the production of tools for the development and deployment of n-tier system designs.

The Gartner Group (1999) recommends using 3-tier if your application has any of the following characteristics:

- Many application services or classes-more than 50.
- Applications programmed in different languages or written by different organizations.
- Two or more heterogeneous data sources-such as two different DBMSs or a DBMS and a file system.
- An application life that is longer than three years-especially if you expect many modifications or additions.
- The expectation that the application will grow over time so that one of the previous conditions will apply.

Tomsen (1998) reports that the n-tier architecture, which separates your application into as many tiers as you need, is the next evolution of distributed

applications. Sun Microsystems, Inc. (2001) explains that the n-tier architecture was designed to enable applications to be distributed as needs dictate. The n-tier application architecture is characterized by the functional decomposition of applications, service components, and their distributed deployment, which provides improved scalability, availability, manageability, and resource utilization. There is no prescribed structure or set of tiers for an n-tier architecture, however the Presentation, Application Server, and Data tiers are standards for all n-type architectures. Because the application of one application server tier of the n-tier architecture can call upon the applications of other application server tiers to share data, there is no limit to the amount of inter-application calling that can occur. Jeri Edwards, (1999) explains that the n-tier architecture is really a 3-tier one that does not implement the application tier as a monolithic program. The n-tier divides up the application tier into a collection of components that are used in a variety of client-initiated business transactions where each component automates a relatively small business function. Components can call other components to fulfill a client request providing application reuse. In addition, components can act as gateways that encapsulate legacy applications running on mainframes. He reports that the benefits of component-based applications (n-tier) over a monolithic application (3-tier) are:

- Can develop big applications in small steps (small projects). Allows for faster production of applications. Reduces risk. Good for small-team, incremental development approach.
- Applications can reuse components – as binary "black boxes" recombining them in different ways depending upon the application.
- Clients can access data and functions easily and safely without needing to know which database is being accessed or how it is being accessed. Encapsulation provides consistent, secure, auditable access to data and eliminates random, uncontrolled updates coming from many applications at once.
- Custom applications can incorporate off-the-shelf components.

- Component environments don't get older – they get better. Component-based systems easily grow into a suite of applications. New applications can be assembled quickly by adding a few new middle-tier components and reusing a number of existing ones. You can update components without changing your clients. You can add new capabilities as you need them.

In 2001, within the world of Internet and intranet applications, n-tier is the odds-on favorite. It lets you start small in both scale and function and then grow your application to huge proportions. The message from the experts is that client/server has grown up and proven itself worthy of being entrusted with key mission-critical applications.

The following case studies from DoD and industry exhibit the benefits of using web-based applications built with 3-tier or n-tier architectures. Each one demonstrates a potential benefit for submarine crews to operate at a higher level of combat readiness.

Finegan (1997) presents the U.S. Atlantic Command's (ACOM) successful implementation of Internet technology to deploy an interactive intranet using web-based solutions to eliminate to stove piped structure which divided and compartmentalized ACOM's eight directorates. Individual departments hoarded information, resulting in officers forced "to work in a virtual vacuum, without the benefit of the expertise residing elsewhere in the staff. It was inefficient, creating across the command an air of isolation from overall mission and ignorance about what was happening and why." ACOM sought to produce "highly informed intellectual products quickly enough to influence events of interest." Their goal was to develop a way to systematically push smart, synthesized information to the decision makers, so they could act upon it.

Web-based solutions were chosen for their cost effectiveness and ease of implementation. Similar to n-tier design philosophy, ACOM developed one new feature at a time, steadily building the functionality they required. Applications built include news summaries, intelligence updates, feature buttons to specific sites such as schedules and staff links, message boards, automated message handling, collaboration sites, e-mail interface, graphics achieve, weapons reference material and sophisticated search engines. Benefits realized by ACOM's efforts include doing more work with fewer people, less redundancy, "dramatically reduced" phone conversations, speed and synergy created around the command and the ability to draw upon world wide expertise through collaboration sites (Finegan, 1997).

The Chief of Naval Operations (2001) states World Wide Web and Intranet technologies provide an opportunity to change business processes in order to create, manage, access, share and exchange information effectively so that decision makers and warfighters are able to get the right information at the right time and at the right place. This is vital to our ability to operate as a cohesive warfighting team.

Paige (1996) acknowledged before the Armed Forces Communications and Electronics Association that in order for our armed forces to maintain battlespace dominance, enabling technologies enhancing C^4ISR must be leveraged via knowledge-based warfare. He explained that knowledge-based warfare uses information technologies to lift the "fog of war" by providing the right information, with the right degree of detail, to the right decision maker at the right time by driving "technology to provide the data collection, analysis, fusion and processing, and information dissemination systems" in order to achieve a common picture of the battlespace. Paige

further explained that both warfighters and decision makers must have access to a distributed, multimedia, information database via a simple and consistent interface regardless of their location or where the information resides. Critical enablers cited by Paige included concerns for security and interoperability at all levels from office applications to seamless information sharing across global networks independent of service, agency, function, discipline or system boundaries without additional translation, remapping or needless duplication.

John Edwards, (1999) observed that information contained in enterprise resource planning (ERP) and data warehouse systems would be invaluable to executives if they could easily access and view the material anytime, anywhere. Providing the application that enables executives to essentially conduct daily operational audits regardless of their location is the goal of Harmony. The Web is an ideal distribution medium for executives who hit the road. Despite their many advantages, Web-based applications also create security and operational concerns. Desisto, as quoted by Edwards (1999), warns that the biggest threat to Web-based applications doesn't come from hackers but from organizations that improperly assign access rights. Desisto notes that all Web-based applications come with tools that allow organizations to accurately direct information. The trick lies in using these tools and not getting sloppy.

Kalin (1998) documents two successful Navy projects to save time and money. The first project Kalin discussed was an extranet connecting Boeing and Navy personnel, called GOSNET. GOSNET allows engineering and other data to be shared across state and organizational lines so that engineers and technicians have access to the most up-to-date design, version, changes, schematics and other information. In addition GOSNET

provides the ability to upload daily aircraft status reports thus eliminating past requirements of faxing and e-mailing information on over 75 aircraft to over 30 different people. GOSNET's biggest obstacle was security for which SSL encryption was used to overcome. Internet technology was chosen over a client/server system because of the ease of using a browser as the user interface thus eliminating the need to load special client software on every workstation. In addition, the client/server solution would have been slow and introduced compatibility issues.

The second project to which Kalin paid tribute was the intranet deployed by the Naval Undersea Warfare Center (NUWC) Newport Division. NUWC developed an intranet to automate the ordering of consumables via an "IntraMart", provide online training, information databases and an executive business information system. NUWC was able to cut 25 jobs saving $600,000 annually and reduce the time it takes to order and receive supplies. NUWC considered a client/server solution, but chose to use Internet technology to avoid having to deploy client software on every client workstation and to capitalize on existing computer infrastructure and browser software already in use.

Young (2000) reports on the tangible and intangible benefits realized by Mitre Corp's implementation of their corporate intranet called Mitre Information Infrastructure (MII). Using MII, Mitre changed their corporate culture from one that fostered internal rivalry and information safeguarding to one with an accessible and shared corporate knowledge base. Operating costs were reduced by shifting a number of tasks toward self-service over MII, resulting in fewer human resource (HR) staff requirements and allowing the remaining HR staff to focus on more sophisticated problems. Staff productivity was improved by shifting mundane activities to MII, such as document

management, time card submission and purchasing tasks. An example cited was Mitre's implementation of a bar-coded equipment tracking system using a database on MII that saved hundreds of man-hours during bi-annual government audits. Mitre increased its knowledge base and sharing of expertise by converting all corporate documents to HTML and establishing a distributed file system. Mitre employees now have access to all corporate knowledge through research tools, captured statistics on projects, department and division collaboration and new data via information correlation. In addition, the usage of lessons-learned in Mitre's MII knowledge repositories resulted in time savings and delivery of improved products. Other benefits Mitre has witnessed as a result of MII include a lower employee turnover rate, lower job frustration, increases in productivity and an improved work environment. (Young 2000)

Compaq (2001) promotes web-enabled applications on their web site stating that they can enhance both new and current technologies. Furthermore, they state that web-based applications benefit by allowing legacy data to be easily and securely accessed from a familiar browser interface. They also state that the web-based environment is more easily deployed and efficiently maintained as it supports platform and operating system independence and network standards. Lastly they argue that web-based applications can be created faster, at less cost and with less risk.

Fabris (1999) reports on Bay Networks' methods of changing their stove-piped corporate database into an Information Management and Delivery System (IMDS). Prior to IMDS employees suffered from information overload via email and voice mail to make up for their inadequate information technology implementation. Numerous employees from various departments published web pages with pertinent product

information, articles, white papers and corporate strategies, but no central search mechanism linked them together. As a result, time and corporate knowledge were wasted.

Bay Networks' solution was dynamic HTML that allowed web browsers to compile documents straight away and eliminate the development and maintenance by programmers creating pages listing related documents. In order to make IMDS a successful tool, Bay Networks studied how information flowed throughout the organization and then created aliases under multiple categories so that multiple links would lead to the same document. To overcome access privileges, Bay Networks used a Lightweight Directory Access Protocol (LDAP) functioning as a central password clearinghouse to automate security access privileges, thus dynamically restricting the documents IMDS presented to individual users. Bay Networks took their browser interface designs from popular Internet sites, built multiple indexes for thorough search capabilities, presented high level categories with drill down features and feedback buttons for online critiques and correction inputs. IMDS significantly improved Bay Networks' productivity and increased their corporate knowledge base.

USMC Standards Branch (2001) Systems Approach to Training (SAT) is a model of recognized standards governing the instructional process in the private sector and within the DoD. This model was adopted by the Marine Corps for use in developing and conducting all of their training and education. The Marine Corp's goal is to develop effective and efficient instruction that promotes transfer of learning from the instructional setting to the job. This tool has the potential to assist the submarine force in developing instruction that meets the Navy's goals.

Navy Office of Training Technology (OTT) (2001) hosts a web site for Seamless Product Information, Data Exchange & Repository (OTT SPIDER). OTT works to improve Navy training programs using Internet and intranet technologies. The OTT serves as a broker of training technology data and information, and coordinates training technology issues, opportunities, initiatives, programs, feasibility studies, policy, guidance and standards. Some of the technologies used in the Fleet include, Automated Electronic Classrooms, Interactive Multimedia Instruction, Training Management Systems and Interactive Electronic Technical Manuals.

The submarine force has a comprehensive training and qualification program. Submarine crews are dependent on teamwork for all facets of operations. OTT (2001) states that a team is two or more people who interact, dynamically, interdependently, and adaptively toward a common and valued goal/objective/mission, who have been assigned specific roles or functions to perform, and who have a limited life span of membership.

Jeri Edwards (1999) states that tactical decision making teams in the modern warfare environment are faced with situations characterized by unfolding events, multiple plausible hypotheses, high information ambiguity, severe time pressure, and serious consequences for errors. The accurate diagnosis of performance shortfalls, and the tailoring of subsequent training toward correcting these shortfalls for teams and individual team members are contingent upon systematic performance assessment from multiple perspectives. A key factor toward ensuring a team's success in a cognitively complex and stressful task environment such as a submarine underway, is training that incorporates explanation, demonstration, practice, and feedback. However, evaluating

team member task work and teamwork skills, and providing meaningful performance feedback real time are complex and demanding tasks for the combat systems trainers.

In conclusion, the challenge remains for the submarine force to capitalize upon and integrate the use of intelligent real time technologies to effectively implement a system to monitor training and readiness resources. This thesis will determine to what degree the submarine force could potentially benefit by implementing automated training and readiness decision support tools throughout the fleet.

THIS PAGE INTENTIONALLY LEFT BLANK

III. ANALYSIS OF PAPER DRIVEN PROCESSES

A. PAPER DRIVEN PROCESSES

Paper driven processes are those that use paper as a medium to communicate between individuals and command units. They include blank and preprinted forms filled in by handwritten ink, typewriters and computer printers. They can be records maintained as documental evidence regarding training, equipment maintenance and qualification requirements as well as correspondence, notices, bulletins or summary and status reports. Reinholt (2000) noted the following problems with paper records in his proposal of automating aviation training records:

1. Documentation Accuracy, Redundancy and Authenticity

Handwritten entries can be confusing or completely illegible due to poor penmanship or the smudging and smearing of ink. Also, recurring corrections clutters the page and places the authenticity of the data into question. Paper records also document redundant information on several separate forms and records, for example a crew member's name and address. Redundancy wastes space, is easily susceptible to discrepancies between like information stored in separate locations and makes maintenance of data laborious and difficult to maintain in sync (Pratt and Adamski, 1991). Initials or a signature are normally used to authenticate paper records and can be easily forged or become illegible as described above. Furthermore, paper records are not easily standardized. The forms, templates and the format of recorded data itself can vary between divisions, departments and/or commands. (Reinholt, 2000)

2. Maintenance, Storage, Security and Cost

Maintaining paper records requires folders, binders and filing cabinets. In addition, depending on the classification of the records, the ability to secure the space in which the records are stored can be a concern. Constant handling of records causes folders and binders to break down, fasteners to break and pages to become torn or lost.

The authors also note that space is wasted in dry storage rooms filled with paper supplies and maintenance materials in support of paper records. Space is a precious commodity aboard cramped submarines and could be used for spare parts. Supplies used in support of paper processes add weight to the submarine and consume funds allocated to the ship's consumables operating budget. Finally, destroying paper resources greatly impacts the operation of submarines at sea. Paper must be shredded, compacted and placed in Trash Disposal Unit (TDU) cans, creating additional work for junior crew members. Operating the TDU to dispose of the trash leaves the boat vulnerable because it is a noisy evolution and places speed and depth restrictions upon the submarine while being performed.

3. Data Sharing, Availability and Portability

Paper driven processes record information in a single location; therefore the data contained within them are not readily shared between users. Since only one person can look at the document at a time, people must either travel to where the document is stored or the document must be routed to the users via their in and out boxes while navigating the chain of command. Disseminating several copies of short-life items, such as bulletins or handouts, wastes paper and generates more trash. Distributing copies of long-term documents, such as instructions or procedures, introduces the problems associated with

redundancy. Data recorded on paper are not readily shared between other data sources. Searching for selected portions of a record or correlating like data from many records requires a manual search. The data must then be manually correlated, recorded and transferred. Lastly, paper records and information contained within them are not easily available to others beyond the submarine. Mailing documents is time consuming and expensive. Faxing large documents is also time consuming and can be expensive if a long distance phone calls are required. (Reinholt, 2000) (Pratt and Adamski, 1991)

B. AUTOMATING PAPER DRIVEN PROCESSES

Reinholt (2000) noted the following benefits to automating paper records in his proposal of automating aviation training records:

1. Documentation Accuracy, Redundancy and Authenticity

Input forms will standardize data entry, reduce documentation errors and increase data integrity by performing entry verification and spell checking upon form submission and by using list boxes to limit input selections to predetermined choices when appropriate. In addition, all entries will be a standard typed font which is legible and easily corrected if an input error is made. Using a relational database system that centrally stores information eliminates redundancy. Controlling access to files and folders using a LDAP, operating system restrictions and digital signatures guarantees authenticity, prevents unauthorized access to sensitive information and prevents forgery.

2. Maintenance, Storage, Security and Cost

Automating paper processes eliminates the space, weight and cost of storing paper records and supplies. In addition, it eliminates the need to repair worn, torn or broken record pages, fasteners and folders. Also, records will no longer get lost or misplaced and regular backups of database records will minimize the impact in the event of a

hardware failure. The security of data will be assured through using a LDAP and operating system file and directory access permissions, which require a password at system login.

3. Data Sharing, Availability and Portability

Probably the most important benefit of automating paper driven processes is that the digital data is easily shared and readily available to simultaneous multiple users no matter their location. Documents will no longer be lost or misplaced in routing boxes while navigating the chain of command. Disseminating multiple copies of short or long term documents and instructions will no longer be required as one centrally stored copy will be easily accessible from many remote stations via a browser. In addition, updates will be faster and easier because they are only made once to the centrally stored electronic data and the updates are available to everyone at the same time. Searching for selected portions of a record or correlating like data from many records will be fast, comprehensive and automatic. Once forms with new data are submitted, subsequent generated reports will automatically include the most recent data available without any further crew member intervention. Data will also be more readily available to others beyond the command via the internet or wireless transmission from a satellite connection. Individual personnel and training records can also be transported on a floppy disk or other digital storage medium making it even faster and more efficient during their subsequent check-in at their following command.

IV. ANALYSIS OF ADMINISTRATIVE PROCESSES

Reinholt (2000) stated that the key element of automating aviation training records was the underlying database. For his purpose the relational database model controlled by a database management system (DBMS) was appropriate. Much like the mobile aviation unit requirements he discussed, Trident submarines operate with two crews and would require a distributed database management system (DDBMS) model in order to sync up data generated separately on the boat and in the off-crew office by the two separate crews.

Many of the administrative processes aboard a submarine fall victim to the generic inefficiencies associated with paper driven processes described in chapter 3. Other processes have moved beyond paper to store data electronically, but function as stove-piped applications. As a result, they do not share data or prevent data redundancy either.

A. PERSONNEL RECORD MAINTENANCE AND OVERLAPPING DATA

1. Problem

People checking into the command give the same data over and over throughout the check-in process. Some data, such as their social security number and date of birth will never change. Other data, such as the division to which they are assigned, their rank and their qualifications completed may change periodically. Despite this fact, redundant data are stored and maintained separately by ship's office personnel, Division Officers and Department Heads, the corpsman, Personnel Reliability Program (PRP) Officer, Nuclear Weapons Radiological Controls (NWRC) Officer, Training Petty Officers, Qualifications Petty Officers and more. Recording and maintaining redundant data waste

space and time. In addition, discrepancies can develop in the data stored at the duplicated locations. (Pratt and Adamski, 1991)

2. Solution

Storing data in a relational database will prevent redundancy, thus minimizing wasted space and eliminated discrepancies between like information stored in separate locations. The entry and maintenance of data will be less time consuming because it will only be entered once and follow-on editing will take place to only one set of centrally stored data (Pratt and Adamski, 1991). For example, the tables on the left in Figure 1 demonstrates how separate programs record redundant data in both the Personnel table and the Training Lecture table. In this case, if a crewmember's named changed due to marriage or their division changed for what ever reason, this data would be required to be

Stove Piped Applications with Redundant Data

Personnel

Last Name	First Name	SSN	Rank	Div	Date Reported	PRD	EOAS
Johnson	Rudy	111111111	E6	M	1/5/97	Jan-02	Jan-01
Smith	Scott	222222222	E4	RC	5/8/96	May-01	Jul-02
Jones	Dale	333333333	E3	TM	6/29/00	Jun-04	Mar-03
Ferris	Tim	444444444	E5	A	1/15/99	Jan-03	Jul-05

Lecture Topics

Lecturer Name	Div	Title	Date
Johnson	M	Flooding	5/6/01
Martin	NAV	Dead Reckoning	5/6/01
Mills	E	Switchboard Fires	5/12/01

Tables Related in Database with no Redundancy

Personnel

Crew Num	Last Name	First Name	SSN	Rank	Div	Date Reported	PRD	EOAS
1	Johnson	Rudy	111111111	E6	M	1/5/97	Jan-02	Jan-01
2	Smith	Scott	222222222	E4	RC	5/8/96	May-01	Jul-02
3	Jones	Dale	333333333	E3	TM	6/29/00	Jun-04	Mar-03
4	Ferris	Tim	444444444	E5	A	1/15/99	Jan-03	Jul-05

Lecture Topics

Crew Num	Title	Date
1	Flooding	5/6/01
18	Dead Reckoning	5/6/01
11	Switchboard Fires	5/12/01

Figure 1. Data Redundancy

updated in both places. In addition, it is likely that a single crew member would give more than one training lecture during their tour – in this case any changes would have to made to more than one record within the Lecture Topics table as well. On the other hand, the tables on the right demonstrate how a relational database records the relation of two tables using primary and secondary keys; in this example the crew member number. As

such, any change to the members name or division would only be recorded once in the Personnel table, and any reference to this person from the Lecture Topics table would automatically relate back to the correct data.

There are currently many programs and applications that can be used as a front-end to access relational databases, and many more programming languages that could be used to develop a specific application in house. However, using a web-based solution will prevent having to install separate client software on every remote computer – which is faster, less expensive, less risky and more efficient to operate and maintain (Compaq, 2001). Other benefits of the web-based approach include increased compatibility across multiple platforms, unlimited scalability, the ability to personalize the interface for groups or individuals, painless deployment of server-based management, universal access, reduced development time and cost due to the availability of tools and authoring applications and reduced training costs due to most persons being familiar with using a browser interface (Edwards, 1999).

Automating administrative processes could yield additional benefits separate from those associated with eliminating redundancy. For example, The U.S. Atlantic Command (ACOM) reported that they increased their command efficiency, speed and synergy, while at the same time they reduced redundancy and their dependency upon phone communications by deploying an interactive intranet using web-based solutions (Finegan, 1997). In another example, the Mitre Corp. increased their productivity by allowing their employees to update their own automated resource records and research the answers to routine questions online. Ultimately, they reduced the number of administrative staff

previously required and the remaining staff were free to deal with more challenging issues. Mitre saved over 16 million in 4 years in labor and material cost (Young, 2000).

3. Suggested Functionality

The following suggested processes or programs do not necessarily require a separated database table for each. For example, some data used by the medical corpsman may be physically stored in the same data table as the information required by the PRP Officer. For each program the user would need the ability to have a personalized view of the data pertinent to their job regardless of what table the data are stored in, the ability to input, edit and delete records, search for and display individual or a group of records based upon specified criteria and the ability to define and generate reports. Recommended areas of administrative automation include:

- Personnel Office for personnel records
- Department Head/Division Officer Notebook
- Personnel Reliability Program (PRP)
- Medical Corpsman records
- Nuclear Weapons Radiological Controls (NWRC) program
- Fitness Reports, Evaluations and Brag Sheets
- Small Arms Inventory
- Sensitive Keys Program
- Fuel, Oil and Water Report data
- Primary and Secondary Maintenance programs

Other processes supporting such things as Training Petty Officers and Qualification Petty Officers will be discussed separately in follow-on chapters.

B. ROUTING, CHOPPING, UPDATING AND APPROVING DOCUMENTS

1. Problem

Paper records and documents must either be routed to the users or users must seek the document – both ways are inefficient and impractical. Documents being chopped have twice the inefficiencies, as they must flow up and down the chain of command with recommendations and changes occurring frequently. In addition, many records have specific review and renewal requirements, which are not readily identified by using paper records. Documents are normally approved with ink signatures, which are susceptible to accuracy and authenticity issues.

2. Solution

Documents should be generated using a word processor or other automated office tools and stored in a digital format providing centralized access through a common 'table top' as argued by Admiral Clemins (1999). Allowing users to go to the documents vise the other way around is more efficient and allows greater access to the data.

> But the site's core payoff remains in replacing sneakerware-the old in-box/out-box system-with the delayering of information delivery. Time was when every person in the chain of command got information from the next-higher level. They decided what was important to pass down, and it was a much-reduced version by the time it reached the bottom, days or even weeks later. Now everybody knows what's going on and can take action more quickly. "That creates speed and synergy around the command," says Dorohovich (Finegan, 1999).

Documents being chopped through the chain of command can easily be highlighted with different colors of text and have notes attached to them. Digital documents are also more readily edited and benefit from other tools, such as spell checkers and grammar checkers. In addition, centrally stored documents improve information flow by relieving bogged down processes suffering from information

overflow (Fabris, 1999). Centrally stored documents would be available from more than one location and could be viewed by several people at the same time. Those documents with specific review and renewal requirements could have e-mail reminders sent to the document owners if they have not been updated in the specified timeframe. Finally, stored documents benefit from the database rules and functionality concerning multi-user and network access of records. Therefore, the documents record attributes could be set to prevent unauthorized changes, thus giving the capability of "signing" or "approving" stored documents and preventing forgery (GenTN002, 2001).

C. PUBLICATIONS

1. Problem

Submarines keep several types of procedures and manuals for the various systems on board. Reactor Plant Manuals, for example, has up to six volumes each with several parts to each volume depending upon the reactor plant type. Each part is maintained in a separate binder and there are normally five copies of each volume maintained on board. It is not uncommon for submarine returning from deployment to have up to 10 boxes of changes and revisions to insert into the various publications retained on board. The authors estimate that an average of 100 hours a month is spent by crew members inventorying, updating and maintaining the total compliment of publications held on board a submarine.

Crew members reference these publications for standard operating procedures and casualty procedure actions, as well as to prepare lectures, study for qualifications and do research on infrequent repairs. With the exception of staged operating and casualty procedures, crew members must track down the manuals they require. Some problems associated with this process include: the required publication is already in use,

publication is improperly stowed or misplaced or publication is not up to date with the current revision or change number.

2. Solution

As many publications as possible should be maintained in electronic format to minimize the time, weight, and cost of supporting paper copies. Most submarines will need to maintain paper copies of frequently used operating and casualty procedures as they are not equipped to view electronic versions at specific watch stations. However, those volumes dealing with information and infrequently used operations should be delivered to commands on CD-ROM or DVD and viewable over the submarines intranet using a browser. One disk can contain all revisions and changes for a particular system. Business rules at the individual commands can then filter the data viewed by crew members based upon the current equipment configuration of their submarine. Updating publications for new equipment configurations would then be as easy as changing the business rules controlling the display filters or changing out the CD-ROM/DVD when a new one is distributed to the fleet. The navy would save money because it is cheaper to produce and distribute CD-ROMs/DVDs than it is to distribute boxes of paper. The inventory, update and maintenance time of the publications program would be significantly reduced while the accuracy and availability of publications would be increased. The problems with improperly stored or misplaced publications would be eliminated.

D. PERIODIC REVIEWS

1. Problem

In addition to normal maintenance, some publications, manuals and other generated records, such as the Small Arms Locker Inventory records, require periodic

documented reviews (audits) at specific intervals by specific personnel. These reviews and signatures can be overlooked due to the manual process of tracking these requirements.

2. Solution

Reviews should be recorded electronically on-line to eliminate the problems associated with paper records. Additionally, business rules can be used to specify which records require periodic reviews, the periodicity of the review and who is required to conduct it. Therefore, automatic email messages can be sent at a specified interval reminding those individuals to conduct the review until they record it as completed. This approach would also aid new Division Officers during their transition period, to learn all of the publications and binders that they are responsible for.

E. LESSONS LEARNED

1. Problem

Lessons-learned used by submarine crews originate from various sources, including individual ships, squadrons, fleet and navy-wide levels. They cover topics regarding general safety, nuclear reactors, weapons systems, ship handling, maintenance programs, mishaps, accidents, incidents, best business practices and many more. Personnel train on those lessons-learned that apply to their specialty as they are released. The lessons-learned are then maintained in a binder by the applicable department yeoman for future reference. The paper system makes cataloging and maintaining these lessons-learned cumbersome and inefficient. This system is also time consuming when researching for specific lessons-learned relating to upcoming training or briefings regarding infrequent operations or difficult tasks. The required manual search for specific lessons-learned is inefficient in that pertinent ones can be overlooked. Difficulty

in finding pertinent lessons-learned could result in repeating mishaps and accidents that may have otherwise been avoided.

2. Solution

A standardized record format with all required fields should be used for lessons-learned navy wide. Lessons-learned from the fleet should be distributed to all commands on CD-ROM or DVD to update local databases. Emergent lessons-learned reported by message format could be entered into the central database at the individual commands, as well as any lessons-learned at their command. Each lesson-learned would have associated fields identifying the applicable date, origin, ship class, department, division, system, type (procedure, documentation or equipment) such that an electronic query would identify all pertinent records for a given search or application.

Automating lessons-learned would save time by eliminating the manual maintenance, tracking and storage associated with paper driven processes. Money would be saved because it is cheaper to send and store electronic bits than it is paper documents and books. In addition, the fleet could potentially operate safer as it would have faster and more comprehensive access to past lessons-learned prior to performing difficult or infrequent operations thereby avoiding mishaps and accidents. By preventing mishaps and accidents the navy would save additional money and would potentially operate at a higher level of readiness.

THIS PAGE INTENTIONALLY LEFT BLANK

V. ANALYSIS OF TRAINING AND READINESS PROCESSES

A. TRAINING PLANNING AND REPORTING

1. Process

Training is an integral part of a submarine crews survival. They must maintain proficiency in frequent and infrequent operating procedures, maintenance procedures, casualty procedures, stealth procedures, safety of ship procedures, various battle scenarios and much more. To meet this challenge submarine crews attend training sessions at the divisional, departmental, wardroom, senior watch station and watch section levels each week. Specific training requirements, including the topics and their periodicity are spread across multiple training manuals and instructions.

a. Short and Long Term Training Plans

Coordination and scheduling of training topics are controlled at the department level using the long range and short range training plans (LRTP and SRTP) submitted by the Department Head and approved by the Executive Officer. The LRTP and SRTP follow standardized formats throughout the fleet. The SRTP is derived from the LRTP and is used as the working document for scheduling training lectures for the current training cycle. Watch station qualifications, monitored evolutions, drill sets and written examinations are methods of obtaining, measuring and maintaining crew proficiency and are processes addressed separately below.

b. Scheduling

Coordination and scheduling of training sessions, evolutions and drills are controlled by ship's weekly routine established by the XO and published in the Plan of the Week (POW) and the Plan of the Day (POD). Commanding Officer's (CO)

permission and scheduling of maintenance and evolutions to be performed at night is normally disseminated in the CO's Night Orders. The POW and POD are posted at various locations throughout the ship and all hands are responsible for reading them. The CO's Night Orders are located in the Control room and are required to be read and initialed by all senior watch standers prior to assuming their watch.

c. Reports

The status of the ship's training and readiness is reported to the CO via the chain of command in the form of a Weekly Training Report (WTR). Training reports are routed on a weekly or bi-weekly basis by department depending upon the command. Training reports include copies of the SRTP, the status of training lectures completed vs. training lectures scheduled, training monitor evaluation sheets, percent qualifications completed vs. qualification goals assigned, all test results, evolution monitor comment sheets and drill set comment sheets.

2. Process Evaluation

a. Short and Long Term Training Plans

The SRTP and LRTP are currently a paper driven process. Because they are used as planning tools they are not modified or updated on a frequent basis, and are therefore adequate for their purpose. Converting them to an electronic format as part of an integrated system could generate additional benefits for decision makers though.

b. Scheduling

Posting of paper copies of the POW and the POD around the ship to get the word out to the crew accomplishes their intended goal, but at a high cost. Not only is it a waste of paper, but having to shred that paper creates additional work for junior crew members. In addition, disposing of them through the TDU impacts the ship's mission as

described in chapter 3. Another problem with posting the PODs and POW throughout the extremely narrow submarine passageways is that it creates congestion, which hinders movement throughout the ship. Similarly, significant congestion occurs in the control room when personnel line-up to review the CO's Night Orders prior to assuming the watch.

c. Reports

The WTR used to access training and readiness is inadequate in that the information it provides decision makers is time-late and does not reflect the real-time status of the ships training and readiness posture. In addition, it fails to identify significant or recurring deficiencies and fails to capture or identify all sources of information that could be provided if the related data from the various information sources were somehow associated. The preparation of the WTR is inefficient because the information is derived manually and there is redundancy in the process. Specific examples will be cited below under processes that provide inputs to the WTR.

3. Web-Based Solution

All data resulting from training and readiness processes should be stored in a distributed relational database.

a. Short and Long Term Training Plans

A web-based application should provide an authoring screen to develop the LRTP by adding, editing and arranging the topics for a specified time frame. There should be a submit button so that the proposed plan can be verified against established business rules to ensure no mandatory topics or periodicities have been overlooked. For example, one business rule may state that flooding training must be performed at least every six months. Business rules should be managed separately and reflect current

requirements from training manuals – they could even be provided to submarines on CD-ROM. Once the LRTP passes verification the DH and XO should be able to approve the proposed plan on-line. Once approved by the XO, the new LRTP should take effect. Because the SRTP is derived from the LRTP, the computer should automatically generate it from the LRTP when requested by a user for a specified time period. When a training topic is performed and recorded as described below, the SRTP should automatically update to reflect the date the lecture was completed.

b. Scheduling

The POW and the POD should be generated by the ship's office using a word processor as it always has, but stored in a pre-approval folder until approved by the XO on-line. Once approved it should automatically be placed in the active folder where it can be linked to from within a browser via the common portal home page. The CO's Night Orders should have similarly functionality, but approving authority would obviously be the CO.

c. Reports

The WTR should become a report that is dynamically generated automatically at the same time each week and electronically pushed to the chain of command where a hyper-linked notice would appear on their homepage the next time they logged in. In addition, the WTR should be able to be dynamically generated whenever desired by a member of the chain of command as well as other reports and queries providing real time status and information. The user should be able to specify the scope, parameters and output format i.e. tabulated, pie chart or bar graph.

B. TRAINING SESSIONS

1. Process

a. Training Materials

Divisional and departmental training is usually in a lecture format presented by senior qualified personnel and carried out in accordance with the SRTP as described above. Seminar formats and walk-through scenarios are also used. Training aids often include power point presentations, overhead projectors, actual ship's equipment and the various procedure manuals.

b. External Training Materials Provided to the Submarine

Individuals qualifying for their dolphins also participate in School of the Boat Training in both group and individual formats. Individual training is normally conducted using Submarine On Board Training (SOBT) provided on laptop computers using interactive training modules on CD-ROM.

c. Documenting Training

All training lectures are monitored by a senior crew member and their comments as well as a list of those who attended the training are recorded on a training evaluation sheet. Training records are maintained in paper format as evidence of complying with requirements during various ship's internal and external inspections. Divisional and departmental representatives maintain their respective training records as a collateral duty. Training records and summaries are routed via the chain of command to the XO/CO as part of the Department's WTR.

2. Process Evaluation

a. Training Materials

Overall the quality of submarine training is excellent and one of the strengths of our submarine force. Technical experts present quality information to other

crew members in order to maintain our forces at the highest level of proficiency and readiness. However, there is a lot of duplication of effort, missed opportunities and inefficiencies built into the current system. Even though many training topics repeat, it is common practice that a different senior person presents a given topic each time. Frequently, training lessons are generated from scratch, instead of starting with a previous lecture or presentation as a foundation. Some commands seldom use structured training syllabi or reuse training materials. Lectures and Power Point presentations are often stored on work center computers or in personal folders on a network drive making the resources unavailable to those who were on watch during the training. These resources are not accessible by individuals seeking additional training materials while undergoing qualifications either. Members who happen to be on watch during training often do not make up that particular lecture. Usually more experienced personnel assume the watch so that junior personnel can attend the training. This can result in the same personnel missing several lectures throughout a given training cycle.

b. External Training Materials Provided to the Submarine

The SOBT materials provide individuals an excellent training resource via interactive modules on CD-ROM. They currently offer 517 training products. However, on average each submarine only has four laptop computers and two copies of each CD-ROM training topic for this program. Equipment failures and insufficient copies of high use topics have failed to meet the demand for this training opportunity on several occasions in the past. Currently, there are plans to increase the number of laptops to eight over the next three years, however they will still be of standalone architecture as the SOBT training materials are not designed to operate over a LAN and can only run on the Windows 95/98 operating system. Furthermore, the check-in/check-out process, the

inventory control and program upkeep of these materials distract watch standers that could be spending their time more productively. (SOBT, 2001)

In addition to SOBT training materials, there are hundreds of prepared training topics available to submarine commands ranging from general military topics to specific specialty training over the Internet. Most commands are not aware of them or do not have access to them while at sea. There are 29 links to related sites with training sources that can be accessed via the Fleet Training Toolbox. This includes links to training resources from the Chief of Naval Education and Training (CNET).

c. Documenting Training

Further inefficiencies within the training process stem from maintaining paper records. Paper forms are used to document the names of crew members present for training. These forms must be modified and printed each time a person transfers in or out of the department usually resulting in unused forms being thrown out. Also, it is time consuming for the Training Petty Officer (TPO) to manually maintain the training records, summarize and transfer the information to the WTR and update the SRTP once training has been conducted. Finally, because of the nature of the manual system, real-time information is not readily available or easily correlated with other pertinent training information from other evaluation sources to provide decision makers with the most complete picture with regards to the command's training and readiness goals.

3. Web-Based Solution

a. Training Materials

Required training should have a hyper-link from the SRTP directly to a structured syllabus for those preparing training. Syllabi should list the minimum requirements that must be covered. Department Heads should be encouraged to add

requirements to cater to their command's special needs. All training materials should be centrally stored and accessible on-line. Each training object, whether it is a Word document or Power Point presentation should be a part of a record in the database. Other fields in the record should be used to index the training topics and aid search features in generating a list of training topics. Examples of some fields that should be included are topic, category, department, division, workstation and date. The category field should indicate the area of interest, such as flooding, fires, communications, reactor safety or first aid. The date field should indicate the last date the material was modified.

The ability to search for pertinent training materials would provide more resources to those preparing future training and could save them time. Personnel who were on watch during training could be provided the link to the training they missed and be required to review the materials on-line after watch. Everyone could link to the resource for further clarification or to brush up before a divisional or departmental exam. In addition, those qualifying would have quick access to a vast amount of additional resources to prepare for their check-outs and exams potentially increasing the commands overall level of knowledge.

b. External Training Materials Provided to the Submarine

All prepared training materials available to submarine commands via the Internet should be provided to the submarines on CD-ROM or DVD and made a part of their training database so that submarine crews underway have use of these resources. Additionally, these materials should be accessible via a browser over the LAN. Accessing these resources via the LAN makes use of existing hardware and eliminates the inventory and maintenance overhead of standalone laptops. In addition, more people

would have access to all the training materials and the check-in/check-out procedure would be eliminated.

c. Documenting Training

All training documentation should be automated to eliminate the inefficiencies associated with paper process. Input forms could be used to record those present for training and the auditor's comments. When the form is submitted, subsequent references to the SRTP and other pertinent reports would automatically reflect the training completion and who was present. In addition, reports could automatically reflect the status of remedial training completion for those who were on watch during the training session. Automating training records would enable the automatic transfer of information used in generating the WTR and updating the SRTP saving the Training Petty Officer time. In addition, information entered into the database would be readily available providing real-time information that is easily correlated with other pertinent training information from other evaluation sources to provide decision makers with the most complete picture with regards to the command's training and readiness goals.

C. DRILLING AND MONITORED EVOLUTIONS

1. Process

Drills and monitored evolutions are both a tool for training and a method for evaluating how watch section teams respond to casualties and perform frequent and infrequent operations. Drills and evolutions fall under four primary areas: Forward, Aft, Weapons and Security. Forward ship's drills and evolutions include ship control surfaces, fires, flooding and loss of hydraulics. Aft ship's drills and evolutions are all those that occur in the engine room. For example, casualties to or operation of the nuclear primary, secondary or auxiliary systems, fire, flooding or loss of hydraulics. Weapons

drills and evolutions include all those related to the ship's weapons systems, including torpedoes and missile systems. Security drills and evolutions include all those related to security alert scenarios and establishing, operating and disbanding security exclusion areas. The specific requirements and frequency of required drills and evolutions are spread across various training manuals and instructions and can be difficult to keep up with.

DHs usually develop a matrix of required evolutions to be performed by each watch section for the current operating cycle. Each respective DH then schedules the evolutions and updates the matrix as they are performed. Evolutions may be performed (or walked through) by watch section teams or as a scheduled monitored event as part of the weeks training agenda. Evolutions are frequent or infrequent procedures and performed in accordance with operating procedure manuals. The senior person in charge of the evolution fills out a comment sheet based upon the watch section's performance on following procedures, the condition of the equipment used and communications maintained between the watch standers. He also assigns a grade and debriefs the watch standers involved following the evolution. Statistics on evolutions performed are maintained in binders as evidence of their completion for internal and externally generated command inspections.

Drill scenarios are maintained in the form of drill guides. Each area (Forward, Aft, Weapons and Security) have specific scenarios mandated that must be performed. Each drill guide normally has several anomalies and initiation methods in order to train and evaluate on numerous possible scenarios. The respective DH and CO must approve

all drill guides. Most drill guides are generated using a word processor and originals are printed and maintained as signed paper copies in thick binders.

Usually a fully qualified junior officer (JO) functions as the drill coordinator. Fully qualified JOs are first term officers who have completed all of their qualifications, and therefore have the most experience among the JOs on board. The drill coordinator is in charge of submitting proposed drill guides, training and briefing his drill team and initiating and conducting the drills. Inputs on which drills to perform are normally provided to the drill coordinator from the various DHs whom maintain a matrix for drill requirements similar to that explained for evolutions above. During the conduct of a training drill, drill-team members observe the watch section's actions and record comments on their performance. They also record comments about the drill teams performance with regards to drill initiation, casualty simulation, training aids and safety (ship and personnel) while running the drill. During follow-on drill debriefs, individual team member's comments are incorporated into a single written list on a drill evaluation sheet and a grade is assigned. The responsible DH usually debriefs the watch section and drill comments are posted for the entire crew to read. Drill evaluation sheets are maintained in a binder. In addition, a record of the most recurring deficiencies is maintained so that those issues can be addressed during future training sessions.

2. Process Evaluation

The current process for scheduling and tracking evolutions and drills has been in place for a long time and essentially gets the job done. However, required evolutions or drills have occasionally been overlooked, especially during transitional periods for newly assigned DHs or senior enlisted assistants. Even with experienced DHs, evolution and

drill requirements have been missed when they were improperly scheduled, marked as completed when only two out of three watch sections completed them, or when a watch section received a below average performance grade for a specific requirement and failed to be retested. Such incidents stem from the inefficient paper system used for scheduling, tracking and documenting evolutions and drills, which is spread throughout a dozen binders. The maintenance of the drill guides suffers from these same inefficiencies.

Drill sets and monitored evolutions are not only a tool for training, but also a method for evaluating how watch section teams perform frequent and infrequent operations and respond to casualties, many of which can be life threatening. Therefore the system for capturing comments, both pro and con, is critical to the readiness and survival of the crew. Based upon the author's personal experience, the current method of recording drill comments on separate sheets of paper for each drill is ineffective for this purpose. On average, three to four drill comment sheets are filled out each day, but there is no easy way to correlate recurring deficiencies between this recorded data. For example, it is possible that 35% of the deficiencies within the engine room come from one watch station, or that 25% of firefighting deficiencies come from improperly donned firefighting equipment. Another scenario could be that 15% of deficiencies come from flooding drills and that 45% of the crew recently failed written examination questions regarding this same topic. This information could alert decision makers that additional training is required for a particular watch station or that more training on a specific topic is required for the entire crew. However, the current system has no way to correlate or present this information to decision makers. Additionally, there are requirements for DHs to maintain a list of the highest recurring deficiencies. Under the current system,

this is a laborious process of constantly shuffling through stacks of drill and evaluation comment sheets to determine which deficiencies are the most frequent.

Finally, it is not enough to simply identify and record deficiencies in crew knowledge and performance, there must be an effective way to disseminate this information to the crew so that they will learn and improve. Currently, the evaluated watch section is normally debriefed after watch by the drill coordinator or the responsible DH. This offers a great learning environment and allows watch standers to ask specific questions. Problems arise when disseminating drill comments (lessons-learned) to the rest of the crew. Several methods used for this in the past include debriefing the drill over the 1MC, posting copies of the comment sheets on bulkheads throughout the ship or inserting a copy of the comment sheets into the CO's Night Orders as required reading. All of these methods are problematic though. A 1MC announcement is a noisy evolution and disrupts those sleeping for the mid-watch, posting numerous copies on bulkheads creates the same problems as mentioned previously associated with the POW and POD and inserting them in the CO's Night Orders creates even more congestion in control during watch turnover and fails to reach 100% of the crew.

3. Web-Based Solution

Evolution and drill matrix requirements should be governed by business rules and tracked for the specified training period (patrol). Drill and evolution comments should be entered into a form and the data submitted to the database. The status of drills and evolutions completed would then automatically reflect the date and grade received as each watch section completed them. It would also indicate if retesting was required for a

failing grade. Automatic tracking of drill and evolution completions would aid DH's in ensuring all requirements are meet and that all required retest were performed. In addition, the inefficiencies associated with paper systems would be eliminated and information would be available to decision makers real time. Advanced search and query features could also be used to correlate data obtained from these and other data sources to provide additional real-time information and reports not previously available alerting decision makers to the need for additional training for a particular watch station or for the entire crew on a specific topic. Additionally, recurring deficiencies could be tracked automatically, eliminating the manual search methods currently used. The following is a list of required functions that would be required:

- Enter, edit and delete evolution and drill matrix requirements (business rules)
- Enter evolutions and drills performed, recording the following information in record fields:
  - Date
  - Evolution or Drill title
  - Watch section – should include names of key watch standers
  - Grade
- Enter specific drill and evolution comments, recording the following information in record fields:
  - Category – recommended choices are Procedure I/A, Procedure S/A, Supervisory Control, Communications, Damage Control, Drill Team and Miscellaneous
  - Sub-category – example sub-categories for Procedure I/A are RPM-CP1, RPM-CP8, SEPM-CP101. etc…
  - Watch station associated with the comment
  - Comment
- Display reports and allow queries of data

Additionally, some fields have been identified that would be required for productive indexing and correlation with data from other training sources.

Evaluated watch sections should continue to be debriefed in person after watch by the drill coordinator and the responsible DH. However, drill and evolution comments should be disseminated to the entire crew via a link in their browser window. The command could require that all members read this information at specified intervals and use computer generated logs to determine who was not complying. Providing this information over the LAN prevents noisy 1MC announcements, eliminates congestion in the passageways and ensure that 100% of the crew is reached.

D. TESTING

1. Process

Written examinations are performed at the division and department levels as well as part of the watch qualifications process. As such, numerous test banks are maintained separately by a variety of personnel. For instance, each division and each department has a Training Petty Officer (TPO) responsible for their exams. In addition, several other billets such as the Diving Officer or NWRC Officer maintain exam banks for qualifications under their responsibility. Exam banks are maintained in a variety of ways depending upon who is responsible. Some simply rotate through different versions of the same exam and rarely add or change the questions. Some frequently change the questions by making up new exams, but manually cut and paste between word processor documents. Still others use an in-house exam generator that randomly selects from available questions in a database. All exams are forwarded for approval prior to the exam being administered, usually to the DH, XO or CO depending upon the exam's purpose. Regardless of how exams are generated, the exam banks are usually maintained on an

individual hard drive or stored on a network hard drive under a specific user's profile, making them unavailable for other training uses such as online self-testing or remedial purposes. In addition, the grading and recording of exams is extremely time consuming. Most often, TPOs manually calculate individual scores, question averages and the overall exam average and then manually transfers the results to an exam summary sheet. Grade summary sheets are submitted through the chain of command as part of the WTR. Individual failures and over-all failed questions are usually required to be retested and reported via the WTR as well.

2. Process Evaluation

Exam banks should be centrally located and exam generation should be a standardized procedure. The current system does not require either. Exams that are rotated through and simply reused have a high risk of compromise and jeopardizes the entire process. Cutting and pasting to develop new exams meets the goals of legitimate testing, however this method is tedious, time consuming and inefficient. In addition, many exams often require a certain number of questions from specific topic areas, which can easily be overlooked using the cut and paste method. None of the methods mentioned makes the question banks available for self-study or remedial testing. In addition, the manual routing process for exam approval can be time consuming and inefficient as exams are sometimes lost or misplaced. By far, the largest shortcoming of the current process is the manual scoring and reporting of exam results. An estimate of 30 hours a week per submarine are wasted by TPOs calculating test scores and averages, which could easily be automated by a computer and completed in seconds. In addition, routing test results in the WTR provides time-late data that is not easily correlated with other evaluation methods. For example, there could be failed exam questions regarding

fire fighting and adverse performance comments by fire fighting teams on recent drills sets. This data correlated and presented together could indicate a command wide deficiency and the need for additional training. The current reporting system does not provide that information to decision makers.

3. Web-Based Solution

Exam banks should centrally store a large number of questions in a database that is accessible over the LAN. Exams should be automatically generated for specified purposes based upon business rules. Business rules are entered separately and ensure the correct percentage of questions from all required topic areas are met. Users should also be able to customize the automatically generated exams and then submit them for verification to ensure they still fulfill all topic area requirements. Lastly, TPO's should have the option to build their own exams from a queried list of questions for a specific topic area.

Once exams are generated they should be capable of online approval eliminating the inefficiencies of manual routing. After exams have been administered, the results should be submitted using an input form for each person who took the exam. Subsequent WTRs or other reports and queries should automatically calculate and display exam results by question or by person eliminating all manual calculations by TPOs. Exam information would be available to decision makers real time and advanced search and query features could also be used to correlate data obtained from other data sources to provide additional real-time information and reports not previously available. Decision makers could then be alerted to the need for additional training for a particular watch station or for the entire crew on a specific topic. The following is a list of required

functions that would be required. Additionally, some fields have been identified that would be required for productive indexing and correlation with data from other training sources, such as drills and evolutions.

- Enter, edit and delete specific exam business rules
- Enter, edit and delete exam questions, recording the following information in record fields:
    - Date question last modified
    - Question
    - Question Answer (key)
    - Total points possible
    - Department
    - Division
    - Watch station or Qualification
    - Category – same as for drill comments
    - Sub-Category – same as for drill comments
- Generate, customize, build, verify and submit exams for approval
- Record exam results by group or individuals taking the exam, recording the following information in record fields:
    - Date exam administered
    - Exam taken (which links to specific questions from question bank)
    - Name of exam taker (link to personnel ID in database)
    - Points earned on each question
- Calculate exam results, display reports and allow queries of data
- Provide method of retesting failed questions
- Provide interface for self testing

Based upon the author's experience, the written examination process is an effective learning tool. Therefore, the exam banks should be set up for people to test themselves. As most exam questions are in essay format, users should be able to specify an area of interest (watch station, qualification or casualty type) and be provided random

questions from the question bank in a flashcard format. Once they have read the question and formulated their answer, they should be able to display the correct answer in order to verify their knowledge. This should not jeopardize the exam process provided the question bank is sufficiently large as crew members will have no way of knowing the exact questions they may be given on future written examinations. Performing self-testing using legitimate questions allows crew members to focus on the areas that leadership feels is the most critical and should contribute to a higher level of knowledge at the command.

E. QUALIFICATIONS

1. Process

First time submariners spend approximately 30 hours a week for up to 18 months qualifying for their Dolphins and continue to qualify in more senior in-rate watch stations for up to five years depending upon their rate. Additional senior qualifications are also assigned as personnel increase in rank and become eligible to stand more senior watches. Most qualifications are good for three years at which time personnel are required to re-certify. Senior personnel spend up to ten hours a week either qualifying or re-qualifying for one watch station or another. The qualification process itself is standardized throughout the Navy. Qualification requirements for specific watch stations, such as Below Decks Watch or Engine room Middle Level Watch, and craftsmen qualifications, such as Primary Valve Operator, are distributed to the Fleet from the responsible agency in the form of qualification cards within a governing publication or training manual. The qualification cards contain between 50 and 150 signature blocks representing knowledge areas or practical factors that are required to become qualified to stand a particular watch or become proficient at a specific skill. Each signature block contains a summary of the

knowledge required, a list of applicable references and specifies the required qualifications of those authorized to sign it as completed.

Chiefs and DOs assign individuals their qualifications so that the ship can maintain the required number of persons qualified on each watch station for both inport and sea operations. Individuals are given a schedule designating the number of signatures that must be obtained each week and the final date by which that qualification must be completed. Individuals then complete their qualification by studying the reference materials or performing hands-on training and seeking out a Qualified Petty Officer (QPO) authorized to give them a checkout. The QPO signs the qualifying crew member's qualification card when they have demonstrated a sufficient level of knowledge or performed the practical factor with sufficient skill.

The progress of the ship's overall qualification process is determined by collecting all qualification cards, counting the signatures obtained and then comparing those numbers to the assigned total for the given date for each qualification card for each crew member. Normally, each division has one person assigned this collateral duty who tracks the divisional progress in the Divisional Qualifications Book. A summary of each division's qualification progress is routed via the chain of command to the XO/CO as part of the Department's WTR.

2. Process Evaluation

Information provided to the chain of command is not real time. It is difficult and cumbersome to correlate the number of qualified watch standers compared to the number of individuals in the process of qualifying (especially when the watch station includes individuals from more than one division) in order to maintain the required support for

inport and at sea watch bills. Furthermore, the current system fails to give the 'big picture' on qualification status at one glance for long term planning. Another shortcoming is that an individual's progress is only checked on a weekly basis and counseling can be overlooked for a couple of weeks in a row if WTR is submitted bi-weekly. The paper qualification cards are easily lost, damaged or inadvertently destroyed. More importantly, the current paper system is susceptible to forgery. The process of counting signatures every week, scheduling, tracking and reporting processes are laborious and time consuming. These processes fail to reflect the most up-to-date status and make inter-week supervision of an individual's progress difficult their cards are only collected at the end of each week. In addition, qualifying individuals wastes on average of three hours a week tracking down reference manuals, which are sometimes checked out, misfiled or not current. Qualifying individuals are not normally aware of other related or enhanced training materials in addition to those listed on their qualification card unless told by a more senior crew member.

3. Web-Based Solution

Qualification card templates should be available digitally via the LAN to assign to personnel with a specified timeframe for completion. Personnel should then be able to access their qualification card online to track their real-time status. Signature blocks could provide hyperlinks to the suggested publications or training manuals to be viewed in their browser, eliminating the wasted time for tracking them down. Publications should be maintained in a CD-ROM/DVD library as previously discussed, giving personnel unabated access to the most current revisions of manuals and publications. In addition, personnel would be able to conduct advanced searches of other training

materials and lectures as discussed above, given them access to a broader range of knowledge and a variety of training formats.

A list of Qualified Petty Officers (QPO) should be defined using business rules for each qualification. When a crewmember successfully completes a checkout, the QPO would access their qualification card online and electronically sign and date it. Only those personnel approved as a QPO for any given check-out would have access to the signature feature.

The proposed solution prevents forgery of signatures or the loss or inadvertent destruction of qualification cards. It saves approximately 40 hours a week per submarine by eliminating the paper process inefficiencies and allowing the computer to count signatures and generate reports on the qualification progress. More importantly it provides real-time qualification status to decision makers making it easier to keep tabs on delinquents. Qualification Petty Officers would spend less time maintaining records and more time on mentorship for those behind in qualifications. In addition, graphical summaries of correlated data could easily display the bigger picture to determine if the ship in maintaining readiness goals, including such data as the number required to man a particular watch station and the current number of persons qualified verses the number of persons and percent complete of those currently working towards qualification.

F. COMMAND INSPECTIONS

1. Process

Submarines undergo inspections of different operational areas. All inspections have common elements, consisting of an extensive scrutiny of the submarines written records, maintenance of operating manuals and publications, crew performance via drill

sets and monitored evolutions, crew level of knowledge via oral interviews, seminars and written examinations and the functional condition and cleanliness of equipment and spaces. Most inspections require up to 200 publications, manuals and binders containing training and maintenance records to be pre-staged in a central location where members of the inspection team spend between one to two days scrutinizing them. The inspection team also provides the chain of command a list of evolutions and casualty scenarios they will expect to see executed in order to evaluate the crew's performance. Often the drill sets transpire in a simulated battle environment. All planning, preparation and execution of evolution and drill sets are the same as explained above, but in a compressed timeframe and under the scrutiny of outside observers. In addition, a combination of written examinations brought aboard by the inspection team and new exams generated from the ship's exam bank are administered to the crew. The exam generation, proctoring, scoring and recording are extremely time challenging events. In addition to written examinations, submarine crews are often evaluated on their ability to conduct meaningful seminars on a given training scenario. The final element of most inspections is a series of oral interviews of randomly selected individuals. Many questions asked during these interviews will relate to highly missed exam questions or to deficiencies noted in the crew's performance during the monitored evaluations, drill sets and seminars.

2. Process Evaluation and Web-Based Solution

In the author's experience, the one to two days spent pouring through publications, manuals and binders is an unacceptable waste of time. The proposed migration to a web-enabled electronic database system as described throughout this thesis could reduce this to three hours of verifying digital records and the few paper copies of

operating procedures that remain on board. The verification of other publications and manuals would be a simple check to verify the most current CD-ROM/DVD was on board and that the business rules used for filtering information correctly represents the equipment configuration on board. Quick and efficient electronic searches and report generation would replace the manual search of operational data regarding evolutions, drills, examinations and qualifications. All the benefits previously argued of the proposed web-based solutions regarding training preparation, evolutions, drills and testing would apply during the compressed and challenging schedule of the command inspection. In addition, crew members could learn from their mistakes and correct their knowledge deficit during the inspection process by reviewing comments and deficiencies via an on-line message board. The authors judge that the time saved via the automated processes would allow the inspection process to be safer, less time consuming, less stressful and more thorough.

VI. IMPROVING COMMAND MANAGEMENT WITH WEB-BASED TECHNOLOGIES

A. BALANCED SCORECARD

In addition to the benefits previously identified of web-based automation of business, readiness and training processes, we now consider the power of using web-based technologies stipulated by Admiral Clemins (1999) in conjunction with the Balanced Scorecard (BSC) organizational management approach advocated by the Procurement Executives' Association (PEA) (1998).

The BSC is a strategic management concept developed by Kaplan and Norton (1992) that has been adopted by a large percentage of Fortune 1000 companies and government agencies, including the PEA. The BSC provides a conceptual framework for translating an organization's strategic objectives into clear measurable performance indicators that define success at every level of an organization. Publicizing specified goals provides focus and direction throughout the organization, strengthens systems and functions and promotes cross-functional coordination that breaks down "stovepipes" within the organization. In addition to using performance measurement information to establish performance goals, the BSC's uses performance measurement information to provide sound data on which to allocate and prioritize resources, aid decision makers in determining to change or maintain current policies and operations and to report on the status of meeting goals. Performance indicators therefore provide a tool for assessing, managing, and improving the overall health and success of the business systems, as well as measure both an organization's current performance and its efforts to improve processes, motivate and educate employees, and enhance information systems enabling

the organization to manage its activities and its resources more effectively. The PEA recommends using the BSC approach across all functional departments and provides a guide for government organizations to assist them in implementing it. (PEA 1998, Balance Scorecard Institute 2001, and SAS Institute Inc. 2001)

B. COMBINING BALANCED SCORECARD WITH WEB PORTAL

Implementing the BSC approach via a browser knowledge portal interface would link the command's vision and strategic goals to individual and group performance making strategy everyone's job as well as provide the whole command the same information so that it can align itself toward a common solution on the same key issues. The ultimate success of the command would become more personal to all crew members. The portal's home page would provide operational data via a message board, such as OPCON condition, weather, Rules of Engagement (ROEs), speed and depth restrictions and authorized maintenance as well as links to the commands vision, mission statement, goals, web-enable applications and required readings, such as drill set comments. The LRTP, SRTP, POW, POD and CO's night orders would be a click away keeping everyone informed of both long-term goals and objectives, as well as the short-term schedule for achieving them. All web-enable applications supporting administration, training and readiness processes would be accessible from this 'common table top' as recommended by Admiral Clemins (1999). The underlying database and supporting information technologies would provide a organization-wide network for sharing data (and knowledge) unlocking and focusing hidden assets and data in order to make the most of our people's energies, talents, and training.

VII. CONCEPTUAL MODEL FOR READINESS AND TRAINING WEB-BASED APPLICATION

A. VISION

The authors' acknowledge that automating all of the proposed submarine training and readiness processes is a large project and therefore not easily completed by one person or as a single application. They envision using relational database tables, similar to the scaled prototype developed using Microsoft Access as shown in appendix A, to store data that will be accessed via a web browser. Each submarine training and readiness process should be developed in a modular approach (using n-tier architecture) to limit the scope of each individual project and promote application component reuse (Edwards, Jeri 1999).

One author developed and field-tested a 2-tier client/server scaled prototype in Microsoft Access (documented in appendixes A, B and C) during his previous tour of duty aboard the USS Tennessee (SSBN 734-B). He established that inefficiencies associated with paper-driven processes could be reduced or eliminated through BPR and automation. The authors' intend that this thesis will facilitate the replacement of the current paper-driven processes that are redundant, inefficient and time-consuming to maintain. Their goal is that this thesis will spark a development effort to field a standardized web-based information system to the submarine fleet which will improve information access and dissemination, provide real-time information and situational trend analysis, facilitate better trained crews with higher levels of knowledge and enhance

communications, crew coordination, productivity and goal alignment on key issues through the implementation of the BSC management approach.

B. LAN INFRASTRUCTURE AND DESIGN

One of the authors' assumptions for this thesis was that all submarines are, or will soon be, equipped with the SNAP III intranet using a fiber optic backbone extending from the forward compartment (FC) through all watertight bulkheads and into the engine room (ER) to service all major compartments as shown in figure 2.

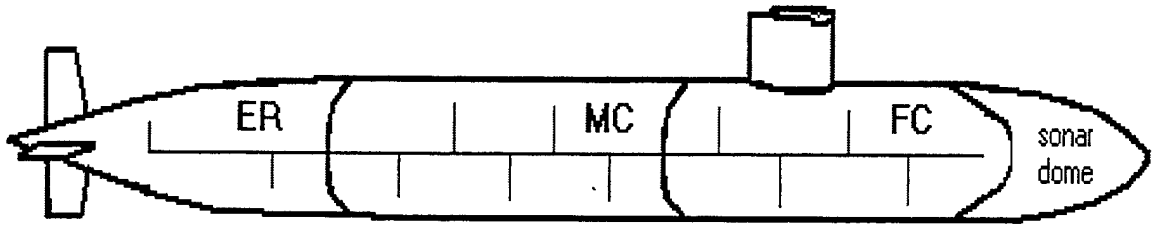

Figure 2. Submarine Fiber Optic Layout (Trident compartments displayed)

Within each compartment, hubs should distribute the fiber optic backbone to client stations in all major work and training spaces, offices, chief's quarters, lounges, mess decks, wardroom and officer staterooms. The LAN client and server workstations should meet or exceed the following Navy IT-21 minimum desktop PC and commercial off-the shelf (COTS) software configuration standards (CNET, 1997):

Client Workstations:

- 133Mhz Intel Pentium processor
- 32MB RAM
- PCI video card with 2MB RAM
- 2.0GB hard drive
- 17" .28mm monitor (1280 x 1024 resolution)
- 3.5" floppy drive
- Dual PCMCIA card reader

- Windows 95 compatible keyboard
- 12x CD-ROM drive
- 16 bit SoundBlaster compatible sound card stereo speakers (set)
- 6 outlet surge suppressor

Server Stations

- Dual Pentium Pro 200 Mhz (Intel) processor
- 128 MB ECC/Parity memory
- Two 4.5GB Ultra SCSI-3 internal hard drives
- Ultrawide PCI SCSI-3 controller
- SEAGATE 4/8gb DAT internal Tape Drive
- 3.5" Floppy Drive
- Dual PCMCIA Card Reader
- S3-based, 64bit w/2MB RAM
- 14" digital .28mm Monitor
- ps/2 Mouse
- ps/2 Win NT compatible keyboard
- 10/100MB PCI Ethernet card
- 700VA or greater UPS

The authors' acknowledge that these minimum standards are dated and recommend exceeding them to ensure that the hardware will have the capability to operate current and future software and operating systems that may be fielded. They recommend the following upgraded minimum hardware configurations:

- 500Mhz Intel Pentium III processor (duel for server)
- 128MB RAM (512MB for server)
- PCI video card with 16MB RAM
- 10GB hard drive (three SCSI 40GB hard drives for server)
- 32x CD-ROM drive
- 100 disk CD-ROM/DVD drive stack for server

The individual LAN hardware component specifics are beyond the scope of this thesis. For further research in this area the author's recommend reading masters thesis authored by Terronez (2000), Sizemore (2000) and Baltzis (2000).

C. PROJECT DESIGN METHODOLOGY

1. The Waterfall Model

This style of development is effective as an organizing tool or when milestone progress tracking is important. The disadvantage to the waterfall approach is its rigid format of completing one phase prior to starting the rest using little user feedback. The following six phases are used in the project development life cycle, with the beginning of each phase dependent on completion of the prior phase:

- Project Identification and Selection
- Project Initiation and Planning
- Analysis Phase
- Design Phase
- Implementation Phase
- Maintenance Phase

The waterfall approach often leads to high costs, extensive maintenance, long development periods and produces a product that no longer meets the users' needs. (Patrou, 1998)

2. The Prototype Model

The prototype approach to the development process concentrates on building a scaled-down version of the desired system, so that the user can better determine their requirements. Prototyping is a means to provide a scaled-down version of a particular desired system that addresses a system need. Prototyping allows the user to visually see and interact with a working version of the program so that all user requirements and

needs are understood. This process is also known for discovering new ideas and solutions to current problems because of the interaction gained through prototyping and feedback between users and developers. (Patrou, 1998)

The primary phases that make up the prototype process are: 1) identifying basic requirements, 2) developing a prototype, 3) users operating the prototype, and 4) prototype revisions and enhancements. When the users are satisfied with all prototype functions and performance, a full-scale version can be designed using the prototype as a model. The benefits to prototyping include, speed, user interaction and feedback, and user satisfaction with the final product. User interaction and feedback provide tremendous beneficial side effects because the user is involved in the development process. Product satisfaction is almost guaranteed because the user and developer work close together throughout the process. (Patrou, 1998)

D. DATABASE DESIGN

A relational database is a collection of related tables that share information to provide dynamic information management and eliminate data redundancy as much as possible (Friedrichsen, 2001). Database tables break down into records (moving horizontally across the table) and fields (moving vertically down the table). Each table contains one or more records. Each field is defined to hold a specific type of data. (Ladd and O'Donnell, 2001)

1. Database Terminology

Friedrichsen (2001) defines the following key database terms:

- Field - a category of information about an item in the database.
- Key Field - a field that contains unique information for each record.
- Record - a group of related fields for an item.

- Table - a collection of records for a single subject.
- Datasheet - a logical view of the fields and records of a single table or query, in which a field appears as a column and a record as a row.
- Database - a broad collection of data associated with a topic.

2. Common Database Objects

Friedrichsen (2001) defines the following database objects contained in Microsoft Access of which the authors' have experienced to be universal for most relational databases:

- Table - contains the raw data within the database, organized by fields and records in a spreadsheet-like view called a datasheet. Tables can be linked by a common field to share information and therefore minimize data redundancy.
- Query - creates a datasheet that displays a subset of fields or records from one or more tables. Queries are created by users to question the database about its data and used to create calculated fields and summarize information.
- Form - provides an easy-to-use data entry screen that generally shows only one record at a time or to display records of information to the screen.
- Report - provides a professional output of data, printed or to the screen, and can contain enhancements such as headers, footers, calculated fields and record or summary groupings with subtotals.
- Page - creates Web pages from objects and provides Web page connectivity features to the database.
- Macro - stores a collection of keystrokes or commands to automate a task such as printing several reports or displaying a toolbar when a form opens.
- Module - stores programming code that extends functions and sub-procedures to automate processes.

3. Common Data Types

Friedrichsen (2001) defines the following data types contained in Microsoft Access of which the authors' have experienced to be universal for most relational databases:

- Text – text characters, numbers not used in calculations or a combination of text and numbers up to 255 characters.
- Memo - text entries beyond 255 characters in text.
- Number - numeric data that may be used with mathematical calculations.
- Date/Time - date and time values.
- Currency - numeric data that represents money.
- AutoNumber - an automatic field entry that increments by one with each new record. They are typically used for primary key fields. Users cannot type into an AutoNumber field.
- Yes/No - referred to as *Boolean* by most programming languages and are used when only one of two values are possible (yes/no, true/false or on/off).
- OLE Object - a linked embedded file created outside of Access, such as an Excel spreadsheet, Word document, graphic or sound clip.
- Hyperlink - Web page address or link to other fields or objects.

Data types are further defined by properties, such as field size, caption, format or validation rule. Properties vary depending upon the data type. For example one property for the Number data type defines whether it is a byte, a short integer or long integer or a single or double precision floating point number.

4. Database Table Relationships

Friedrichsen (2001) describes the three relationship types used by relational databases as follows:

- One-to-One – a record in Table X has no more than one matching record in Table Y.
- One-to-Many – a single record in Table X has many records in Table Y.
- Many-to-Many – a record in Table X has many records in Table Y, and a record in table Y has many records in Table X.

Examples of table relations can be seen in Appendix A. Lines between key fields in the related tables represent the table relationships. The "1" denotes the *one* side of a relationship and the "∞" denotes the *many* side of the relationship. Many-to-Many

relationships function by using a third table, called a join table, between them. Each table has a One-to-Many relationship with the join table. (Pratt and Adamski, 1991)

Normalization is the process of designing a relational database and involves determining the appropriate fields, tables and table relationships. Synchronization between tables is accomplished through relationships that are developed using a common linking field between two tables. The common linking field in the table on the *one* side is called the key field (also known as the primary key field). The common linking field in the table on the *many* side is called the foreign key field. Databases allow the developer to choose to enforce referential integrity of relationships. Enforcing reverential integrity will prevent deleting records from the *one* side of a relationship when there are related records in the table on the *many* side. (Friedrichsen, 2001)

5. Database Products Available

a. Microsoft Access

Microsoft Access is a relational database management system that is a part of the Microsoft Office Suite. It is frequently used as a server-side Web database because many people want to Web-enable databases they previously developed on their desktops. "Access performs well on the Web up to a point, but after you have a threshold number of records in the database or too many concurrent users trying to hit your database, things will break down and you should consider upgrading to SQL Server." (Ladd and O'Donnell, 2001) Mercer (2001) recommends using Access to rapidly prototype databases for a particular project and then use a conversion and migration tool "to go from a prototype in Access to a full-fledged, industrial-strength, database-management system when you've stabilized the functionality required."

b. Microsoft SQL Server

Microsoft developed SQL Server to compete with Oracle and Sybase to get database-driven information to the Web. Microsoft SQL Server works well with any Web development tool that complies with Microsoft's OBDC standard. Microsoft intends for their SQL Server database, Microsoft Internet Information Server and their Internet Explorer browser to be used together to provide a single Web environment. (Ladd and O'Donnell, 2001)

c. Oracle

Oracle provides databases for Windows NT and UNIX that have their own tools called the Oracle Web Developer Suite. Oracle is one of the largest database developers in the world. (Ladd and O'Donnell, 2001)

d. Sybase

Sybase's Enterprise Application Server is the cornerstone of their Internet products. It includes tools that can be used to produce dynamic Web pages from a database. Sybase also produces PowerBuilder for developing Web applications that easily integrate with the database to perform advanced queries. (Ladd and O'Donnell, 2001)

e. MySQL

MySQL was developed by T.c.X to handle large amounts of data and provide a robust SQL engine comparable to other major commercial database servers, but at a moderate price. (Ladd and O'Donnell, 2001)

6. Proposed Database Table Design and Relationships

Automating all of the submarine training and readiness processes as proposed by this thesis is a large and demanding project that would require up to 75 database tables or

more. In the Authors' opinion, a project of this size should be completed using a modular format such as the Prototype development model so that users can aid developers in defining the requirements and testing the functionality as the project grows. Figure 3 displays proposed database table fields and relationships that could be used to automate the Personnel, Drills, Evolutions and Qualification processes. The authors recommend

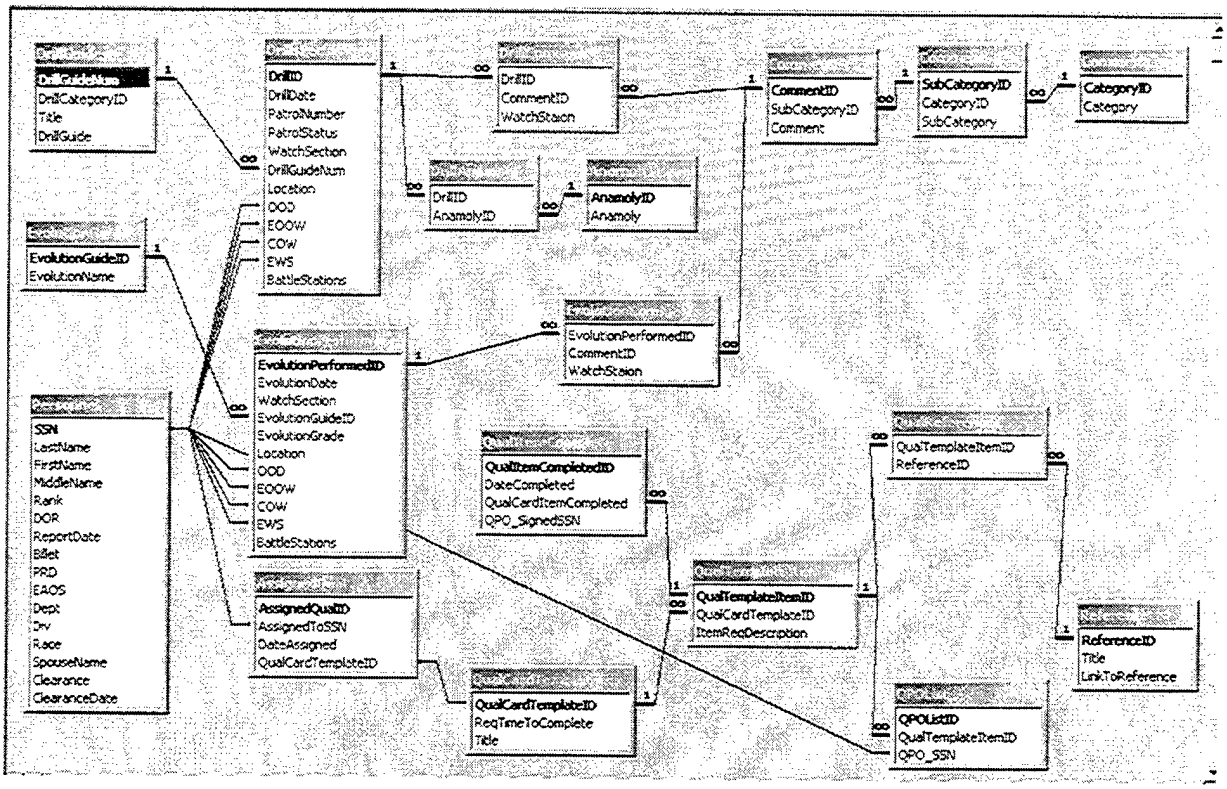

Figure 3. Proposed Database Table Relationships

reading the masters thesis authored by Whitecar (2001) for a thorough database table design and working prototype that automates all functions relating to medical service records. The authors' recommend linking the Web-based application proposed by this thesis with the one developed by Whitecar.

The following list includes other processes this thesis has identified for which database tables, or queries using existing table fields and relationships, would need to be

designed to support the development of a comprehensive submarine training and readiness Web-based application. There is not a one-to-one correspondence between the number of tables required to the number of processes that the authors' have recommended be automated as can be seen by referring to number of tables used to support automating the Drills process in Figure 3 above.

- Training
- Written examination
- Fitness Reports, Evaluations, Counseling and Brag Sheets
- Publications
- Primary and Secondary maintenance
- Document control
- Small arms
- Sensitive keys program
- Periodic Reviews
- Lessons Learned

E. APPLICATION DEVELOPMENT

The authors' research confirms that Web sites are more and more becoming complete online applications. Whereas early Web sites used static pages to display text and graphics, today's dynamic Web sites perform very complex functions, much like traditional application programs that interact with users and respond to their requests. (Ladd and O'Donnell, 2001)

Static Web pages are not created automatically when a user requests a page; they remain the same from one access to another. Their content only changes when the Webmaster uploads a revision. Contrary, dynamic Web pages are automatically created, or have portions that are automatically created, when a page is accessed. Developers can

choose among Client-side tools and Server-side tools to create dynamic Web-applications. (Microsoft, 2000)

1. Client-side Tools used to Build Dynamic Applications

a. JavaScript

JavaScript is a simple but powerful language that does not have to be compiled or linked into applets that are downloaded. JavaScript does not allow access to your computer outside your Web browser. It is not possible to create viruses or Trojan Horses with JavaScript, nor is it possible to extract information about you with JavaScript. JavaScript is best for Web pages that need straightforward animation, scrolling text or other simple programming requirements (Microsoft, 2000).

b. VBScript

VBScript is perfect for intranet environments based on Microsoft operating systems and applications, as it interfaces well with Visual Basic for Applications and ActiveX controls. Like JavaScript, VBScript does not allow access to your machine, so you cannot use it to propagate malicious software or to extract information from your computer (Microsoft, 2000).

c. ActiveX

ActiveX controls are compiled plug-ins that can be automatically downloaded and installed to provide application-like user-interface controls on a Web page. ActiveX technology is most appropriate for intranet or Internet servers where security isn't a major issue and similar software is in use (Microsoft, 2000).

d. Java

Java is a complete interpreted programming language based on the syntax of C++. Java is supported by more Web browsers than any other sort of dynamic content and has relatively few security-related problems (Microsoft, 2000).

2. Server-side Tools used to Build Dynamic Applications

a. Common Gateway Interface (CGI)

The CGI is the standard that started programs on the Web server computer for returning dynamically created HTML documents to the HTTP service for transmission to the remote client. A new instance of the CGI application starts each time a connection is made. CGI can be used for the following functions: e-mail gateways, image-maps, cryptographic security, security restrictions to private Web pages and random selection from text files (Microsoft, 2000).

b. Internet Server Applications Programming Interface (ISAPI)

Microsoft developed the (ISAPI) to make it easier to customize Microsoft's Internet Information Server and to provide a faster interface than CGI for making dynamically generated Web pages (Microsoft, 2000).

c. Active Server Pages (ASP)

Active Server Pages (ASP) are an extension to Internet Information Server that enables you to run server-side scripts written in JavaScript or VBScript; those scripts return dynamically created HTML documents based on user input or other variables. Unlike static Web pages, dynamic or Active Server Pages make it easy to communicate with the server; to start, control and manage the site as a persistent application; and to work with individual users during separate sessions. Because ASP processes output before it gets back to the user, all manner of functionality can be built in, such as

database access, component usage, and the ordinary programmatic functionality available with VBScript, Jscript, or any other scripting language you'd care to use. (Mercer, 2001)

The ActiveX Data Object (ADO) is an ASP component and is the most valuable component ASP can call. ADP objects allow direct and convenient access to databases and other data sources, and this functionality serves many common purposes on application-driven Web sites. ASP also includes other components, and you can buy third-party components online. Components are valuable because they offer pre-programmed functionality that would take many hours of effort to duplicate yourself, even assuming you had excellent programming skills. (Mercer, 2001)

F. DESIGN ISSUES

1. Facets of Web Design

User considerations are of paramount importance, but during the design process, designers should compose a mission statement based on IT-21 architecture requirements. Microsoft's latest version of Internet Explorer 5.5 is recommended due to its compatibility and availability (down-load free from Microsoft).

By balancing end-user considerations with the Navy's objectives, designers will produce a site that has broad appeal and that helps attain the communication goals. Post your mission statement, requirements summary or objective list in a common work space on a white-board so that you and your design team can always be reminded of your goals. (Ladd and O'Donnell, 2001)

Audience characteristics and the Navy's objectives for creating a site are the human factors that go into Web-site design. As you begin to focus on the site itself, two other factors require attention: the information you're presenting and the "look and feel"

of the site. Two approaches for structuring content have emerged during the Web's short history: the drill-down structure (also known as the layered structure) and the flat structure. (Ladd and O'Donnell, 2001)

Many of the notions that go into designing a Web site also go into the design of a single Web page, but some page design considerations are unique. Ideally, page design should follow site design; when you get ready to start a page, you should already have a good sense of what the page needs to accomplish, given its place in your site design. Knowing your audience and designing to that audience requires you to gather as much information about them as possible, including:

- Equipment configuration
- Learning characteristics (how to best present information so they understand it)
- Motivations for visiting the Web site
- Demographic factors

(Ladd and O'Donnell, 2001)

2. Security

A submarine underway is essentially a secure intranet with no contact or threat from the Internet and the World Wide Web. Keeping data on an intranet private is just as easy as keeping any other set of files private. Because intranets run on file servers, network administrators can simply use file system permissions to keep certain Web pages or FTP files private. Users who attempt to access web pages for which they don't have permission will receive an access-denied message.

Submarines in port will be connected to the Intranet over a dedicated line from its network to an Internet Service Provider. That permanent connection is always vulnerable to damaging attacks. Security measures must be implemented to protect a submarine's network and data from this constant threat. Many methods are available to implement security mechanisms for submarine Internet servers in port; the most effective security measures that the authors' recommend include the following:

a. Proxy Server

A server dedicated to the function of receiving Internet Web requests for clients, retrieving the requested pages, and forwarding them to the client. Proxy servers cache retrieved Web pages to improve performance and reduce bandwidth; they also serve the security function of protecting the identity of internal clients. A proxy server also makes firewall and packet filter configuration easier because all external connections to the services mediated by the proxy will go to the proxy server computer. Any other external connections can be explicitly denied. (Microsoft, 2000)

b. Dedicated Internet Servers

Submarine crews in port should not use their file and print server as an Internet server; instead, administrators should place a dedicated machine in a different protection domain to host WWW and FTP services. (Microsoft, 2000)

c. Client and Internet Server Internet Access Only

Network administrators can make it much harder for attackers to get at their file and application servers by not allowing those servers to connect to the Internet (Microsoft, 2000).

d. Separate Network for Internet Access

Submarine crews require access to the Internet while in port, but simply cannot accept the risk of local network intrusion from the Internet. One solution: create a second network for Internet services. The submarine's intranet will be physically isolated and secure from attackers just as if the submarine were underway. (Microsoft, 2000)

e. Windows Security

Regardless of the Internet security implemented at the border between the network and the Internet, administrators should have strong Windows NT domain security in their LAN. After a hacker penetrates your Internet security, the Windows security may still protect your data if you have set up the security properly. It certainly will provide another mechanism that will allow you the chance to see that an intrusion has occurred. (Microsoft, 2000)

f. Windows NT Built-In Filtering

Windows NT Server includes packet-filtering in the TCP/IP interface. Administrators can use this filtering in addition to a strong firewall to control access to individual servers (Microsoft, 2000).

g. Firewalls

When attaching a submarine's LAN directly to the Internet through a router, firewalls will allow administrators to restrict the information flowing through the firewall by type (for instance, only HTTP or FTP information), IP address of the computers and many other factors. Firewalls are the best way to protect networks from intrusion via the Internet. Firewalls are usually used in conjunction with packet filters

and with Proxy servers, which provide even more service-level functionality. (Microsoft, 2000)

h. Filtered Packet Services

Some ISPs provide a firewall service that will automatically filter traffic to and from networks by type and if requested they will also restrict access to networks from certain IP addresses or subnets (Microsoft, 2000).

i. Internet Service Restrictions

Most Internet service software, including Microsoft's Internet Information Server (IIS), includes security mechanisms to prevent functions like listing the files in a Web site or uploading files to an FTP site (Microsoft, 2000).

j. File and Directory Security

Submarine crews in port that set up their own Web server should require users to log on to the server. This method allows administrators to implement regular Windows NT file-level security to keep documents secure. (Microsoft, 2000)

k. Encryption

Administrators can further secure servers through the use of MS-CHAP password and data encryption and the Point-to-Point Tunneling Protocol (PPTP). These two services encrypt the data that flows over the Internet so that even if it were intercepted between two computers, it would be nearly impossible to decode. (Microsoft, 2000)

l. Secure Socket Layer (SSL)

Most browsers and Internet servers support the SSL protocol. The SSL establishes an encrypted link between the Internet server and browser. The SSL is useful

for keeping transactional data from being compromised while submarine crews work on the Web. (Microsoft, 2000)

Maintaining security on a submarine's Internet and intranet site is an ongoing process. None of these methods precludes the use of any others. It is recommended to implement all of them or as many as network administrators can reasonably justify for the level of protection required by their data. Administrators must always be aware of the threats to their Internet sites and Windows NT server that hosts it. Making sure that the right permissions are set on the files and directories is only half the problem. Administrators must also make sure that network intruders attempting to exploit weaknesses in network programs and operating system services have no weaknesses to exploit. (Microsoft, 2000)

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

Adversaries of the United States are gaining new technological capabilities comparable to ours through commercial industrialization at a relatively low cost. United States military leaders warn that our future advantage over our adversaries will come from people, leadership, doctrine, organizations and training that enable us to take advantage of information technology to achieve superior war fighting effectiveness. The Navy's goals are to provide all users rapid and secure access to the same information in order to improve existing operational processes of command and control. Currently, the Navy plans to move beyond "Network-Centric Warfare" to that of a "Knowledge-Centric" environment so that people will conduct both war fighting and business missions in a virtual environment via a common tabletop using a web browser interface.

The submarine force currently has the technology infrastructure to deploy and operate as Naval leaders envision, however the applications in use are mostly paper-driven or stand alone applications and do not leverage the available technology to provide real-time data analysis as required by commanders and other decision makers.

Training is a key requirement for achieving combat readiness. The increased capability of the personal computer enables us to bring enhanced training methods to the desktop through simulation and animation. Comprehension and retention are improved through the use of these teaching aids. It is our responsibility as leaders to continue to leverage these technologies that facilitate the transfer of knowledge to sailors and Marines. This thesis focused on methodologies that utilize an automated training and readiness decision support system that could enable submarine crews to operate at a

higher level of combat readiness. The proposed applications and methodologies could assist submarine crews in assessing and modifying operating schedules further enhancing strategic readiness. Additionally, this thesis demonstrates how web-based applications would enable submarine crews to save time, eliminate redundancy, improve commander's decision-making capabilities, provide wider dissemination of information, correlate results between the various information sources and provide real-time information and trend analysis to decision makers.

B. RECOMMENDATIONS

1. Network Architecture

The authors recommend using the n-tier architecture in order to automate submarine training and readiness processes in small projects using an incremental Prototype development approach. The n-tier architecture is good for small-team development, allows component reuse and supports encapsulation, which provides consistent, secure and auditable access to data. In addition, the n-tier approach supports both custom applications and off-the-shelf components and will allow faster production at less risk.

Microsoft offers products that support n-tier architecture and are consistent with the Navy's IT-21 standards (Tomsen, 1998).

2. Further Thesis

Development of a Web-enabled prototype automating submarine training and readiness processes in order to expedite delivery of a final product to the fleet.

APPENDIX A. ACCESS PROTOTYPE TABLE RELATIONS

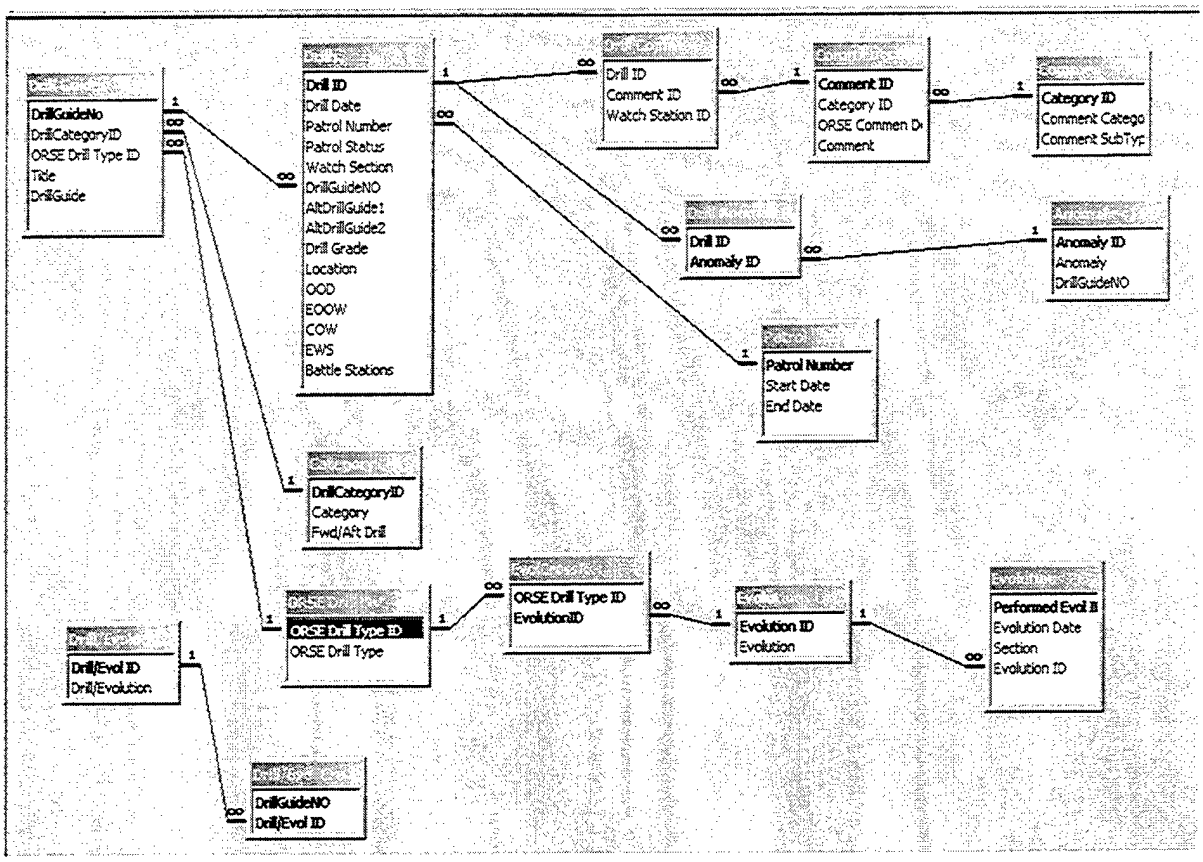

Figure A-1. Access Prototype Database Table Relationships

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. ACCESS PROTOTYPE SCREEN SHOTS

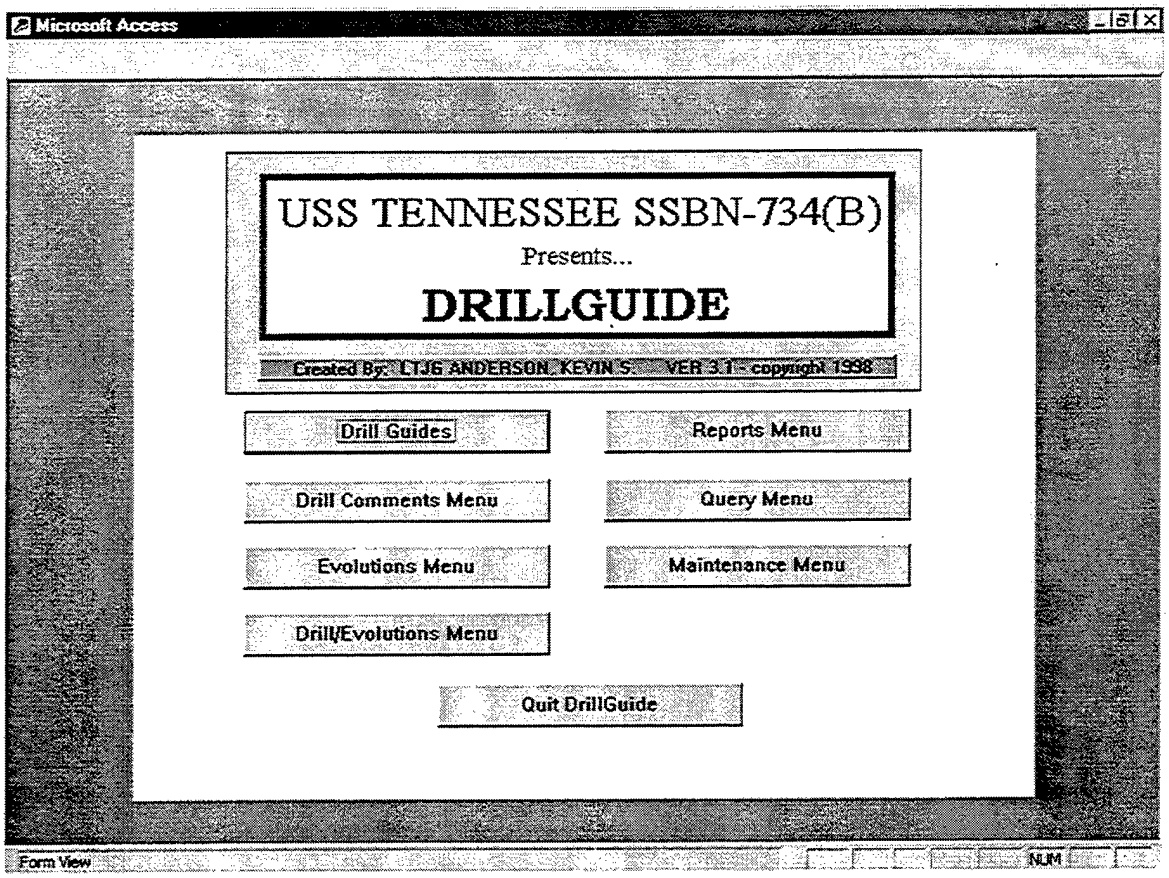

Figure B-1. Prototype Main Switchboard Form

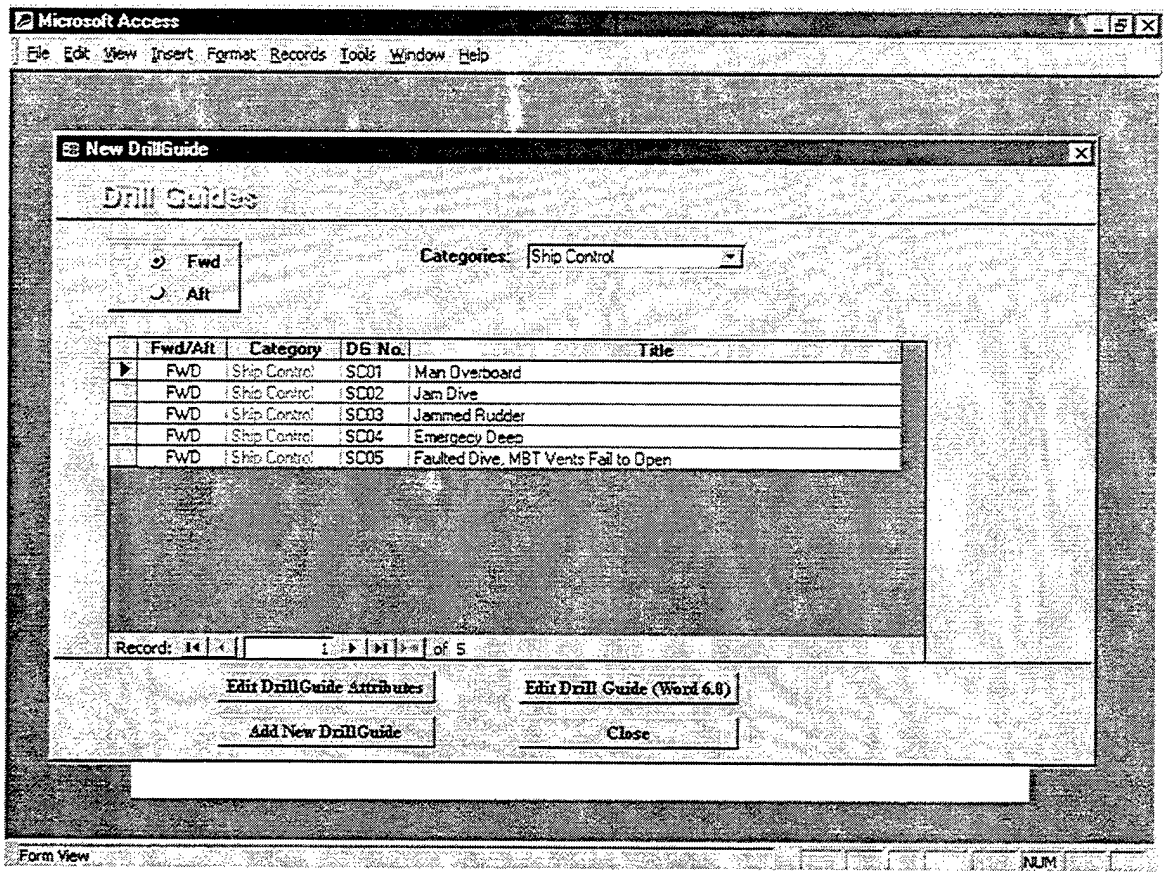

Figure B-2. Drill Guides Maintenance Form

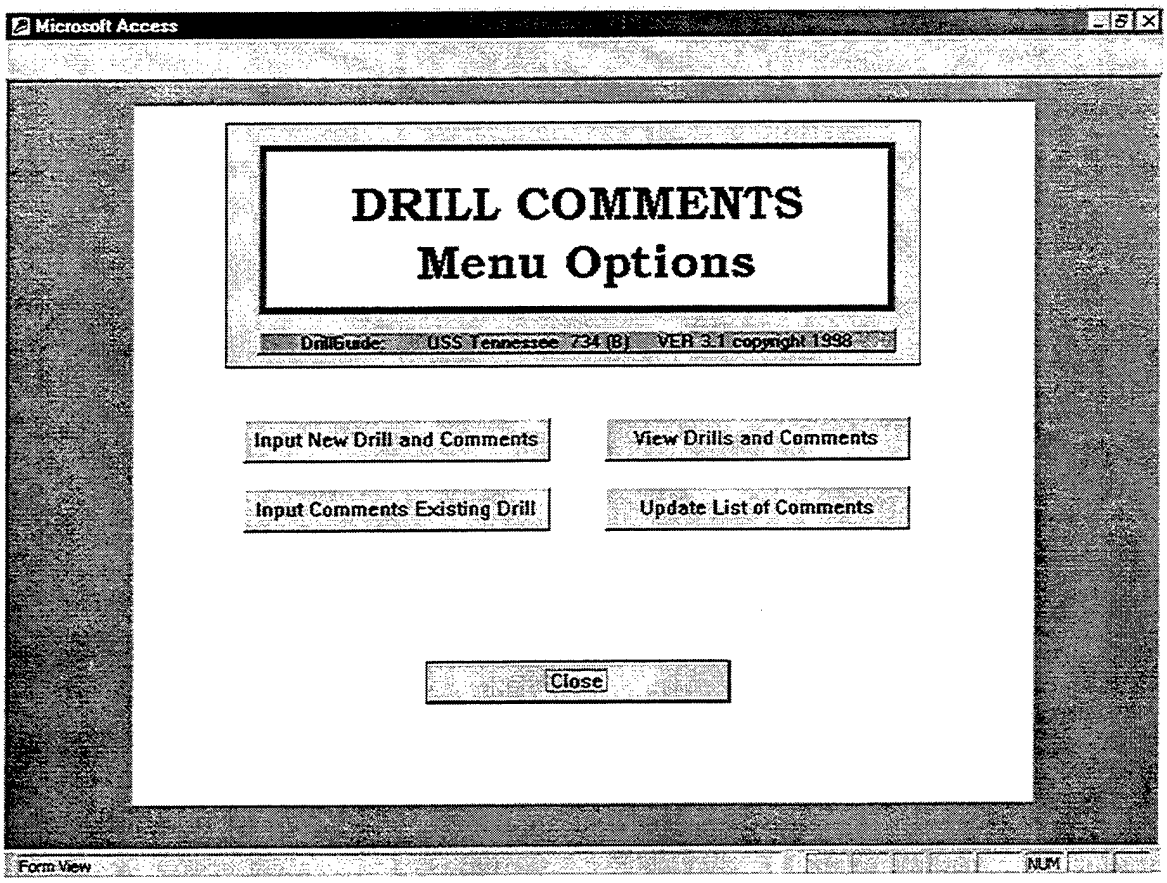

Figure B-3. Drill Comments Switchboard

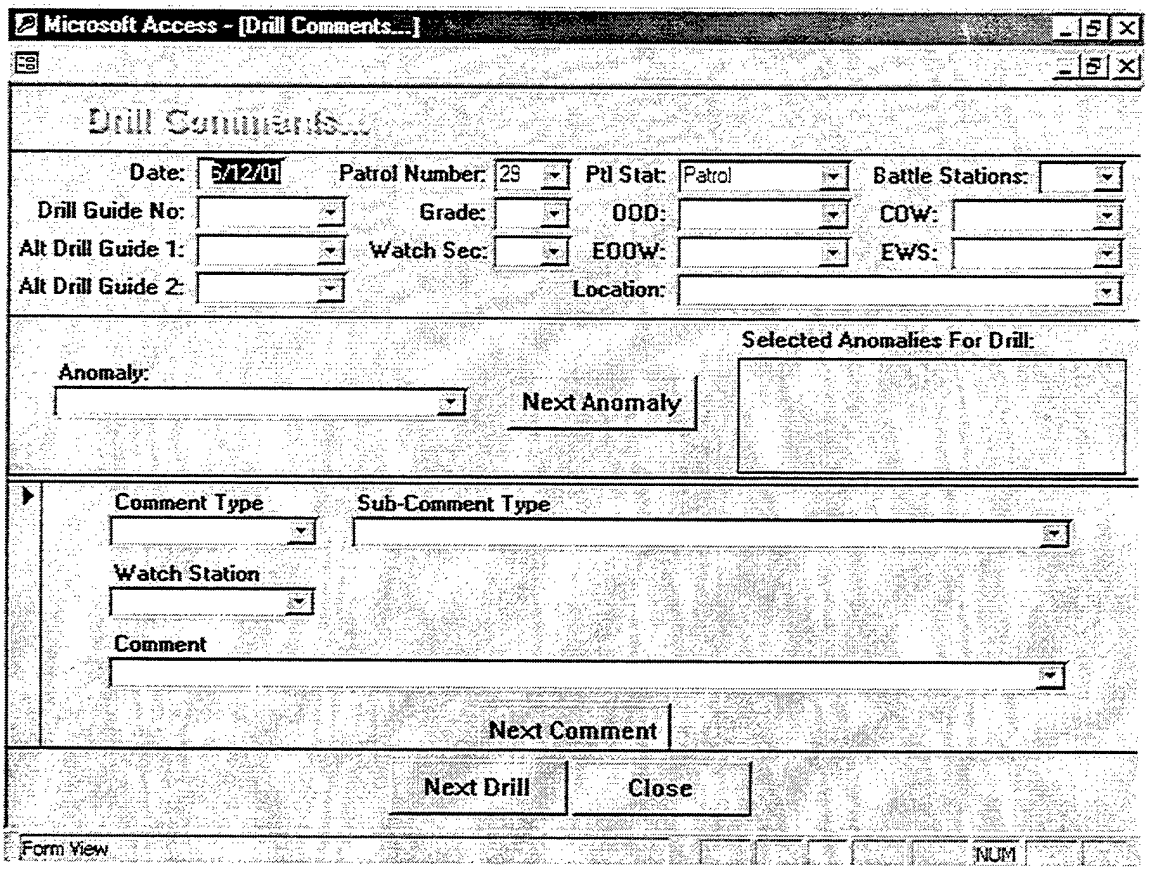

Figure B-4. Drill and Drill Comments Input Form

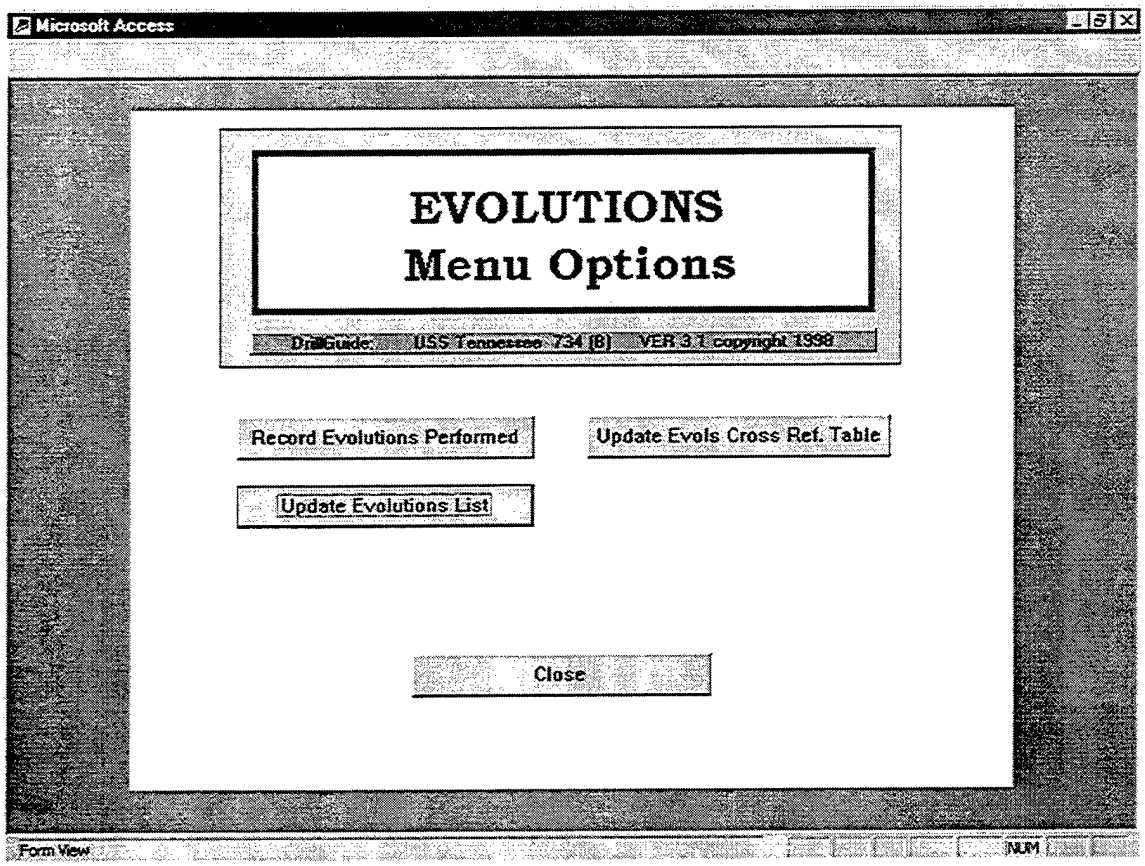

Figure B-5. Evolutions Switchboard

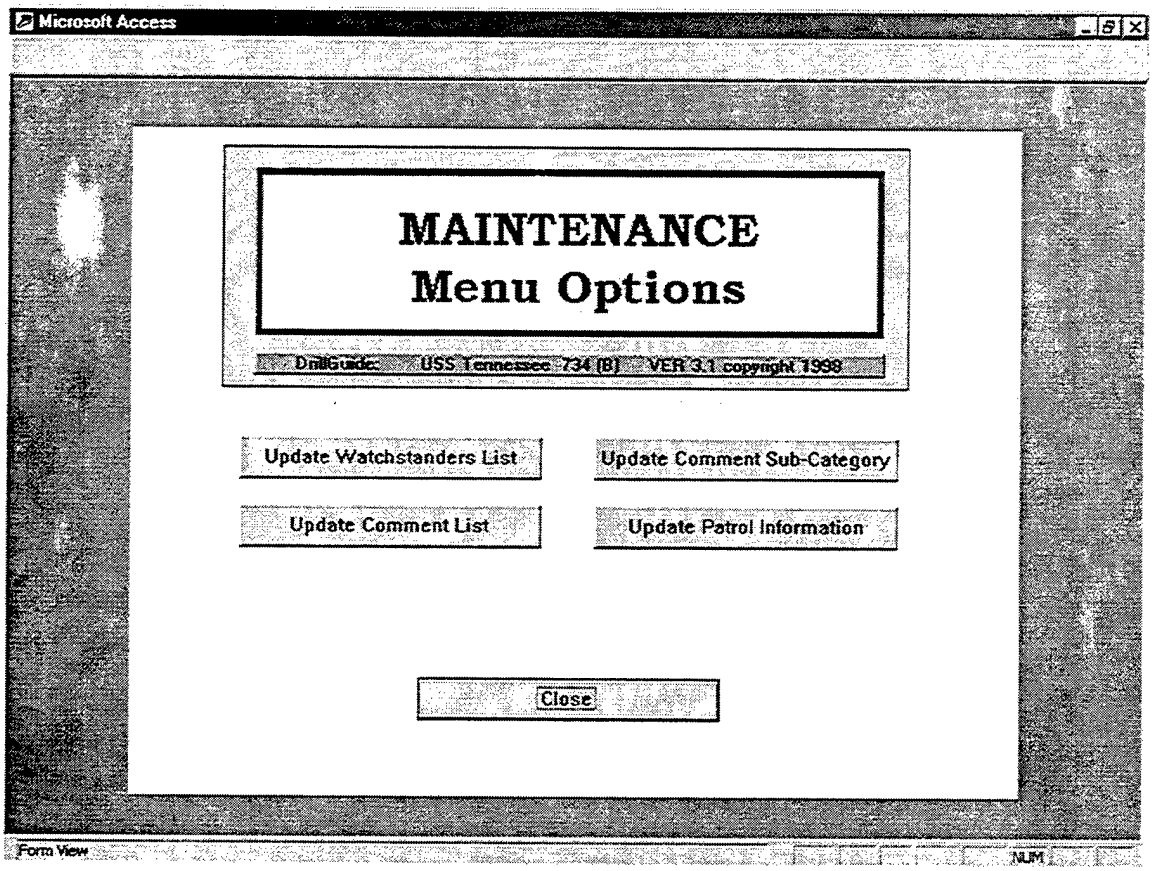

Figure B-6. Maintenance Switchboard

APPENDIX C. ACCESS PROTOTYPE CODE

```
'
'              UTILITY FUNCTIONS
'
'    This module contains useful functions that you
'    can use in expressions on your forms and reports.
'
'
Option Compare Database   'Use database order for string comparisons
Option Explicit

Function IsLoaded(MyFormName)
' Accepts: a form name
' Purpose: determines if a form is loaded
' Returns: True if specified the form is loaded;
'       False if the specified form is not loaded.
' From: User's Guide Chapter 25
   Dim i
   IsLoaded = False
   For i = 0 To Forms.Count - 1
      If Forms(i).FormName = MyFormName Then
         IsLoaded = True
         Exit Function      ' Quit function once form has been found.
      End If
   Next
End Function

Function IsLocked(rs As Recordset, UserName As String, MachineName As String)
' Accepts: a recordset and two string variables
' Purpose: determines if the current record in the recordset is locked,
'       and if so who has it locked
' Returns: True if current record is locked (and sets UserName and MachineName
'       to the user with the lock).  False if the record isn't locked.
' From: Building Applications Chapter 12
   Dim ErrorString As String
   Dim MachineNameStart As Integer
   IsLocked = False
   On Error GoTo IsLockedError
   rs.Edit              'Try to edit the current record in the recordset.
   rs.MoveNext
   rs.MovePrevious
   Exit Function            'No error, so return False.
IsLockedError:
   If Err = 3260 Then       'Record is locked -- parse error string.
```

```
      ErrorString = Error$
      UserName = Mid$(ErrorString, 44, InStr(44, ErrorString, """) - 44)
      MachineNameStart = InStr(43, ErrorString, " on machine ") + 13
      MachineName = Mid$(ErrorString, MachineNameStart, Len(ErrorString) -
MachineNameStart - 1)
      IsLocked = True
   End If
   Exit Function
End Function

Function NullToZero(anyValue As Variant) As Variant
' Accepts: a variant value
' Purpose: converts null values to zeros
' Returns: a zero or non-null value
' From: User's Guide Chapter 17
   If IsNull(anyValue) Then
      NullToZero = 0
   Else
      NullToZero = anyValue
   End If
End Function

Function Proper(anyValue As Variant) As Variant
' Accepts: a text value
' Purpose: converts first letter of each word to uppercase
' Returns: converted text value
' Note: although this function converts most proper names correctly, it converts
'     'McKee' to 'Mckee', 'van Buren' to 'Van Buren', 'John III' to 'John Iii'
   Dim ptr As Integer
   Dim theString As String
   Dim currChar As String, prevChar As String
   If IsNull(anyValue) Then
      Exit Function
   End If
   theString = CStr(anyValue)
   For ptr = 1 To Len(theString)          'Go through string char by char.
      currChar = Mid$(theString, ptr, 1)  'Get the current character.
      Select Case prevChar                'If previous char is letter,
                                          'this char should be lowercase.
         Case "A" To "Z", "a" To "z"
            Mid(theString, ptr, 1) = LCase(currChar)
         Case Else
            Mid(theString, ptr, 1) = UCase(currChar)
      End Select
      prevChar = currChar
   Next ptr
```

```
    Proper = CVar(theString)
End Function
Sub ShowEvent (Event As String)
' Accepts: a text value, the type or name of an object and the name of an event
' Purpose: displays an event history list in the Event History form
    If IsNull(Forms![Event History]![History]) Then
        Forms![Event History]![History] = Event
    Else
        Forms![Event History]![History] = Event & Chr(13) & Chr(10) & Forms![Event
History]![History]
    End If
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub DrillGuideNo_Exit(Cancel As Integer)
    Me![DGFilename] = Trim(Me![DrillGuideNo]) + ".DOC"
End Sub

Private Sub FwdAft_AfterUpdate()
    Me![SelectCategory].Requery
End Sub

Private Sub SelectCategory_AfterUpdate()
    If Me![FwdAft] = 0 Then
        Me![ORSE Drill Type].Enabled = True
    Else
        Me![ORSE Drill Type].Enabled = True
        Me![ORSE Drill Type] = 1
        Me![ORSE Drill Type].Enabled = False
    End If
    Me![ListDgs].Requery
End Sub

Private Sub WORD1_Click()
On Error GoTo Err_WORD1_Click
    Dim x As Integer
    Dim FileName As String
    Dim WordObj As Object
    Set WordObj = CreateObject("Word.Basic")
    FileName = "C:\Drilbase\" + Me![DGFilename]
    WordObj.Fileopen FileName
    WordObj.Bold 1
    WordObj.Insert "USS TENNESSEE"
    WordObj.InsertPara
```

```
    WordObj.Insert "SSBN 734BLUE"
    WordObj.EditSelect
    Response = DATA_ERRCONTINUE
    Exit_WORD1_Click:
    Exit Sub
Err_WORD1_Click:
    MsgBox Error$
    Resume Exit_WORD1_Click
End Sub

Private Sub Word2_Click()
On Error GoTo Err_WORD2_Click
    Dim x As Integer
    Dim FileName As String
    Dim WordObj As Object
    Set WordObj = CreateObject("Word.Basic")
    FileName = "C:\Drilbase\" + Me![DGFilename]
    WordObj.Fileopen FileName
    WordObj.EditSelect
    Response = DATA_ERRCONTINUE
Exit_WORD2_Click:
  Exit Sub
Err_WORD2_Click:
    MsgBox Error$
    Resume Exit_WORD2_Click
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Anomaly_AfterUpdate()
    Forms![Input Comments]![Drill Anomalies].Form![Selected Anomalies].Requery
End Sub

Private Sub Anomaly_NotInList(NewData As String, Response As Integer)
      DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
      DoCmd.OpenForm "NewAnomaly", A_NORMAL, , , A_ADD
      Forms![NewAnomaly]![Anomaly] = NewData
      Response = DATA_ERRCONTINUE
End Sub

Private Sub Button20_Click()
On Error GoTo Err_Button20_Click
    DoCmd.GoToRecord , , A_NEXT
    Forms![Input Comments]![Drill Anomalies].Form![Selected Anomalies].Requery
Exit_Button20_Click:
```

```
    Exit Sub
Err_Button20_Click:
    MsgBox Error$
    Resume Exit_Button20_Click
End Sub

Private Sub Button27_Click()
On Error GoTo Err_Button27_Click
    DoCmd.GoToRecord , , A_NEWREC
    Forms![Input Comments]![Drill Anomalies].Form![Anomaly].SetFocus
Exit_Button27_Click:
    Exit Sub
Err_Button27_Click:
    MsgBox Error$
    Resume Exit_Button27_Click
End Sub
Private Sub Button27_Exit(Cancel As Integer)
 Forms![Input Comments]![Drill Anomalies].Form![Selected Anomalies].Requery
End Sub
```

```
Option Compare Database   'Use database order for string comparisons
Option Explicit
Const ERR_RPT_CANCELED = 2501

Private Sub Cancel_Click()
' Close form.  (This code created by Command Button Wizard.)
On Error GoTo Err_Cancel_Click
    DoCmd.Close
Exit_Cancel_Click:
    Exit Sub
Err_Cancel_Click:
    MsgBox Error$
    Resume Exit_Cancel_Click
End Sub

Private Sub Patrol_AfterUpdate()
    Me![Selected Drill].Requery
End Sub

Private Sub Preview_Click()
' Preview report.
On Error GoTo Err_Preview_Click
    Dim DocName As String
    DocName = "Drill Critique"
    DoCmd.OpenReport DocName$, 2
```

```
Exit_Preview_Click:
   Exit Sub
Err_Preview_Click:
   If Err = ERR_RPT_CANCELED Then
      Resume Exit_Preview_Click
   Else
      MsgBox Error$
      Resume Exit_Preview_Click
   End If
End Sub

Private Sub Print_Click()
' Print report.
On Error GoTo Err_Print_Click

   Dim DocName As String
   DocName = "Drill Critique"
   DoCmd.OpenReport DocName$, 0
Exit_Print_Click:
   Exit Sub
Err_Print_Click:
   If Err = ERR_RPT_CANCELED Then
      Resume Exit_Print_Click
   Else
      MsgBox Error$
      Resume Exit_Print_Click
   End If
End Sub
```

```
Option Compare Database   'Use database order for string comparisons
Option Explicit

Const ERR_RPT_CANCELED = 2501
Private Sub Cancel_Click()
' Close form. (This code created by Command Button Wizard.)
On Error GoTo Err_Cancel_Click
   DoCmd.Close
Exit_Cancel_Click:
   Exit Sub
Err_Cancel_Click:
   MsgBox Error$
   Resume Exit_Cancel_Click
End Sub

Private Sub Preview_Click()
```

```
' Preview report.
On Error GoTo Err_Preview_Click
   Dim DocName As String
   DocName = "Drill/Evolution Report"
   DoCmd.OpenReport DocName$, 2
Exit_Preview_Click:
   Exit Sub
Err_Preview_Click:
   If Err = ERR_RPT_CANCELED Then
      Resume Exit_Preview_Click
   Else
      MsgBox Error$
      Resume Exit_Preview_Click
   End If
End Sub

Private Sub Print_Click()
' Print report.
On Error GoTo Err_Print_Click
   Dim DocName As String
   DocName = "Drill/Evolution Report"
   DoCmd.OpenReport DocName$, 0
Exit_Print_Click:
   Exit Sub
Err_Print_Click:
   If Err = ERR_RPT_CANCELED Then
      Resume Exit_Print_Click
   Else
      MsgBox Error$
      Resume Exit_Print_Click
   End If
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button33_Click()
On Error GoTo Err_Button33_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "UpdateDrill/EvolsList"
   DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button33_Click:
   Exit Sub
Err_Button33_Click:
   MsgBox Error$
```

```
    Resume Exit_Button33_Click
End Sub

Private Sub Button34_Click()
On Error GoTo Err_Button34_Click
    Dim DocName As String
    Dim LinkCriteria As String
    DocName = "Drills/Evol Cross Ref"
    DoCmd.OpenForm DocName, , , LinkCriteria, A_ADD
Exit_Button34_Click:
    Exit Sub
Err_Button34_Click:
    MsgBox Error$
    Resume Exit_Button34_Click
End Sub

Private Sub Button35_Click()
On Error GoTo Err_Button35_Click
    DoCmd.Close
Exit_Button35_Click:
    Exit Sub
Err_Button35_Click:
    MsgBox Error$
    Resume Exit_Button35_Click
End Sub

Private Sub CommentList_Click()
On Error GoTo Err_CommentList_Click
    Dim DocName As String
    DocName = "CommentInput"
    DoCmd.OpenForm DocName, A_NORMAL, , , A_ADD
Exit_CommentList_Click:
    Exit Sub
Err_CommentList_Click:
    MsgBox Error$
    Resume Exit_CommentList_Click
End Sub

Private Sub DrillComments_Click()
On Error GoTo Err_DrillComments_Click
    Dim DocName As String
    DocName = "Input Comments"
    DoCmd.OpenForm DocName, A_NORMAL, , , A_ADD
Exit_DrillComments_Click:
    Exit Sub
Err_DrillComments_Click:
```

```
    MsgBox Error$
    Resume Exit_DrillComments_Click
End Sub

Private Sub DrillGuides_Click()
On Error GoTo Err_DrillGuides_Click
    Dim DocName As String
    DocName = "New DrillGuide"
    DoCmd.OpenForm DocName, A_NORMAL, , , A_ADD
Exit_DrillGuides_Click:
    Exit Sub
Err_DrillGuides_Click:
    MsgBox Error$
    Resume Exit_DrillGuides_Click
End Sub

Private Sub View_Drills_Click()
On Error GoTo Err_View_Drills_Click
    Dim DocName As String
    Dim LinkCriteria As String
    DocName = "View Comments"
    DoCmd.OpenForm DocName, , , A_READONLY
Exit_View_Drills_Click:
    Exit Sub
Err_View_Drills_Click:
    MsgBox Error$
    Resume Exit_View_Drills_Click
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button32_Click()
On Error GoTo Err_Button32_Click
    Dim DocName As String
    Dim LinkCriteria As String
    DocName = "EvolutionsPerformed"
    DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button32_Click:
    Exit Sub
Err_Button32_Click:
    MsgBox Error$
    Resume Exit_Button32_Click
End Sub

Private Sub Button33_Click()
```

```
On Error GoTo Err_Button33_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "UpdateEvolsList"
   DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button33_Click:
   Exit Sub
Err_Button33_Click:
   MsgBox Error$
   Resume Exit_Button33_Click
End Sub

Private Sub Button34_Click()
On Error GoTo Err_Button34_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "UpdateCrossRefTable"
   DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button34_Click:
   Exit Sub
Err_Button34_Click:
   MsgBox Error$
   Resume Exit_Button34_Click
End Sub

Private Sub Button35_Click()
On Error GoTo Err_Button35_Click
   DoCmd.Close
Exit_Button35_Click:
   Exit Sub
Err_Button35_Click:
   MsgBox Error$
   Resume Exit_Button35_Click
End Sub

Private Sub Button36_Click()
On Error GoTo Err_Button36_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "Input Comments"
   DoCmd.OpenForm DocName, A_NORMAL, , , A_ADD
Exit_Button36_Click:
   Exit Sub
Err_Button36_Click:
   MsgBox Error$
   Resume Exit_Button36_Click
```

```
End Sub

Private Sub Button37_Click()
On Error GoTo Err_Button37_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "Select Patrol"
    DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button37_Click:
   Exit Sub
Err_Button37_Click:
   MsgBox Error$
   Resume Exit_Button37_Click
End Sub

Private Sub Button38_Click()
On Error GoTo Err_Button38_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "CommentInput"
   DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button38_Click:
   Exit Sub
Err_Button38_Click:
   MsgBox Error$
   Resume Exit_Button38_Click
End Sub

Private Sub Button39_Click()
On Error GoTo Err_Button39_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "Select Drill"
   DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button39_Click:
   Exit Sub
Err_Button39_Click:
   MsgBox Error$
   Resume Exit_Button39_Click
End Sub

Private Sub Button40_Click()
On Error GoTo Err_Button40_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "UpdateCommentsList"
```

```
    DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button40_Click:
    Exit Sub
Err_Button40_Click:
    MsgBox Error$
    Resume Exit_Button40_Click
End Sub

Private Sub CommentList_Click()
On Error GoTo Err_CommentList_Click
    Dim DocName As String
    DocName = "CommentInput"
    DoCmd.OpenForm DocName, A_NORMAL, , , A_ADD
Exit_CommentList_Click:
    Exit Sub
Err_CommentList_Click:
    MsgBox Error$
    Resume Exit_CommentList_Click
End Sub

Private Sub DrillComments_Click()
On Error GoTo Err_DrillComments_Click
    Dim DocName As String
    DocName = "Input Comments"
    DoCmd.OpenForm DocName, A_NORMAL, , , A_ADD
Exit_DrillComments_Click:
    Exit Sub
Err_DrillComments_Click:
    MsgBox Error$
    Resume Exit_DrillComments_Click
End Sub

Private Sub DrillGuides_Click()
On Error GoTo Err_DrillGuides_Click
    Dim DocName As String
    DocName = "New DrillGuide"
    DoCmd.OpenForm DocName, A_NORMAL, , , A_ADD
Exit_DrillGuides_Click:
    Exit Sub
Err_DrillGuides_Click:
    MsgBox Error$
    Resume Exit_DrillGuides_Click
End Sub

Private Sub View_Drills_Click()
On Error GoTo Err_View_Drills_Click
```

```
    Dim DocName As String
    Dim LinkCriteria As String
    DocName = "View Comments"
    DoCmd.OpenForm DocName, , , A_READONLY
Exit_View_Drills_Click:
    Exit Sub
Err_View_Drills_Click:
    MsgBox Error$
    Resume Exit_View_Drills_Click
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button25_Click()
On Error GoTo Err_Button25_Click
    DoCmd.Close
Exit_Button25_Click:
    Exit Sub
Err_Button25_Click:
    MsgBox Error$
    Resume Exit_Button25_Click
End Sub

Private Sub Button26_Click()
On Error GoTo Err_Button26_Click
    DoCmd.GoToRecord , , A_NEXT
    Me![Selected Drills].Requery
Exit_Button26_Click:
    Exit Sub
Err_Button26_Click:
    MsgBox Error$
    Resume Exit_Button26_Click
End Sub

Private Sub Button27_Click()
On Error GoTo Err_Button27_Click


    DoCmd.DoMenuItem    A_FORMBAR,    A_EDITMENU,    A_UNDOFIELD,    ,
A_MENU_VER20
Exit_Button27_Click:
    Exit Sub
Err_Button27_Click:
    MsgBox Error$
    Resume Exit_Button27_Click
```

```
End Sub

Private Sub Drill_Evol_ID_AfterUpdate()
   Me![Selected Drills].Requery
End Sub

Private Sub Drill_Evol_ID_NotInList(NewData As String, Response As Integer)
      DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
      DoCmd OpenForm "UpdateDrill/EvolsList2", A_NORMAL, , , A_ADD
      Forms![UpdateDrill/EvolsList2]![Drill/Evolution] = NewData
      Response = DATA_ERRCONTINUE
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Close_Click()
On Error GoTo Err_Close_Click
   DoCmd.Close
Exit_Close_Click:
   Exit Sub
Err_Close_Click:
   MsgBox Error$
   Resume Exit_Close_Click
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub DelRec_Click()
On Error GoTo Err_DelRec_Click
   DoCmd.DoMenuItem A_FORMBAR, A_EDITMENU, A_SELECTRECORD_V2, ,
A_MENU_VER20
   DoCmd.DoMenuItem    A_FORMBAR,    A_EDITMENU,    A_DELETE_V2,    ,
A_MENU_VER20
   DoCmd.Requery
Exit_DelRec_Click:
   Exit Sub
Err_DelRec_Click:
   MsgBox Error$
   Resume Exit_DelRec_Click
End Sub

Private Sub Evol_Date_Exit(Cancel As Integer)
   Me![StartDate] = Me![Evol Date]
   Me![StopDate] = Me![Evol Date]
```

```
    DoCmd.Requery
End Sub

Private Sub Patrol_Number_Exit(Cancel As Integer)
    Me![StartDate] = Me![Patrol Number].Column(1)
    Me![StopDate] = Me![Patrol Number].Column(2)
    DoCmd.Requery
End Sub

Private Sub ViewBy_AfterUpdate()
If Me![ViewBy] = 1 Then
    Me![Patrol Number].Enabled = True
    Me![Evol Date].Enabled = False
End If
If Me![ViewBy] = 2 Then
    Me![Patrol Number].Enabled = False
    Me![Evol Date].Enabled = True
End If
End Sub
```

```
Option Compare Database   'Use database order for string comparisons
Option Explicit

Const ERR_RPT_CANCELED = 2501
Private Sub Cancel_Click()
' Close form.  (This code created by Command Button Wizard.)
On Error GoTo Err_Cancel_Click
    DoCmd.Close
Exit_Cancel_Click:
    Exit Sub
Err_Cancel_Click:
    MsgBox Error$
    Resume Exit_Cancel_Click
End Sub

Private Sub Form_Load()
    ' Enter parameters in Beginning Date and Ending Date text boxes when
    ' form is loaded.
    Me![StartDate] = #1/2/1993#
    Me![StopDate] = #1/16/1993#
End Sub

Private Sub Patrol_AfterUpdate()
    Me![StartDate] = Me![Patrol].Column(1)
    Me![StopDate] = Me![Patrol].Column(2)
```

```
End Sub

Private Sub Preview_Click()
' Preview report.
On Error GoTo Err_Preview_Click
   Dim DocName As String
   DocName = "EvolutionsPerformed"
   DoCmd.OpenReport DocName$, 2
Exit_Preview_Click:
   Exit Sub
Err_Preview_Click:
   If Err = ERR_RPT_CANCELED Then
      Resume Exit_Preview_Click
   Else
      MsgBox Error$
      Resume Exit_Preview_Click
   End If
End Sub

Private Sub Print_Click()
' Print report.
On Error GoTo Err_Print_Click
   Dim DocName As String
   DocName = "EvolutionsPerformed"
   DoCmd.OpenReport DocName$, 0
Exit_Print_Click:
   Exit Sub
Err_Print_Click:
   If Err = ERR_RPT_CANCELED Then
      Resume Exit_Print_Click
   Else
      MsgBox Error$
      Resume Exit_Print_Click
   End If
End Sub

Private Sub RangeStart_AfterUpdate()
   Me![StartDate] = Me![RangeStart]
   Me![StopDate] = Me![RangeStart]
   Me![RangeStop] = Me![RangeStart]
End Sub

Private Sub RangeStop_AfterUpdate()
   Me![StopDate] = Me![RangeStop]
End Sub
```

```
Private Sub RptDates_AfterUpdate()
If RptDates = 1 Then
   Me![Patrol].Enabled = True
   Me![RangeStart].Enabled = False
   Me![RangeStop].Enabled = False
End If
If RptDates = 2 Then
   Me![RangeStart].Enabled = True
   Me![RangeStop].Enabled = True
   Me![Patrol].Enabled = False
End If
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button32_Click()
On Error GoTo Err_Button32_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "EvolutionsPerformed"
   DoCmd OpenForm DocName, , , LinkCriteria, A_ADD
Exit_Button32_Click:
   Exit Sub
Err_Button32_Click:
   MsgBox Error$
   Resume Exit_Button32_Click
End Sub

Private Sub Button33_Click()
On Error GoTo Err_Button33_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "UpdateEvolsList"
   DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button33_Click:
   Exit Sub
Err_Button33_Click:
   MsgBox Error$
   Resume Exit_Button33_Click
End Sub

Private Sub Button34_Click()
On Error GoTo Err_Button34_Click
   Dim DocName As String
   Dim LinkCriteria As String
```

```
    DocName = "UpdateCrossRefTable"
    DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button34_Click:
    Exit Sub
Err_Button34_Click:
    MsgBox Error$
    Resume Exit_Button34_Click
End Sub

Private Sub Button35_Click()
On Error GoTo Err_Button35_Click
    DoCmd.Close
Exit_Button35_Click:
    Exit Sub
Err_Button35_Click:
    MsgBox Error$
    Resume Exit_Button35_Click
End Sub

Private Sub CommentList_Click()
On Error GoTo Err_CommentList_Click
    Dim DocName As String
    DocName = "CommentInput"
    DoCmd.OpenForm DocName, A_NORMAL, , , A_ADD
Exit_CommentList_Click:
    Exit Sub
Err_CommentList_Click:
    MsgBox Error$
    Resume Exit_CommentList_Click
End Sub

Private Sub DrillComments_Click()
On Error GoTo Err_DrillComments_Click
    Dim DocName As String
    DocName = "Input Comments"
    DoCmd.OpenForm DocName, A_NORMAL, , , A_ADD
Exit_DrillComments_Click:
    Exit Sub
Err_DrillComments_Click:
    MsgBox Error$
    Resume Exit_DrillComments_Click
End Sub

Private Sub DrillGuides_Click()
On Error GoTo Err_DrillGuides_Click
    Dim DocName As String
```

```
    DocName = "New DrillGuide"
    DoCmd.OpenForm DocName, A_NORMAL, , , A_ADD

Exit_DrillGuides_Click:
    Exit Sub
Err_DrillGuides_Click:
    MsgBox Error$
    Resume Exit_DrillGuides_Click
End Sub

Private Sub View_Drills_Click()
On Error GoTo Err_View_Drills_Click
    Dim DocName As String
    Dim LinkCriteria As String
    DocName = "View Comments"
    DoCmd.OpenForm DocName, , , A_READONLY
Exit_View_Drills_Click:
    Exit Sub
Err_View_Drills_Click:
    MsgBox Error$
    Resume Exit_View_Drills_Click
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button26_Click()
On Error GoTo Err_Button26_Click
    DoCmd.GoToRecord , , A_NEWREC
Exit_Button26_Click:
    Exit Sub
Err_Button26_Click:
    MsgBox Error$
    Resume Exit_Button26_Click
End Sub
Private Sub Close_Click()
On Error GoTo Err_Close_Click
    DoCmd.Close
Exit_Close_Click:
    Exit Sub
Err_Close_Click:
    MsgBox Error$
    Resume Exit_Close_Click
End Sub

Private Sub Evolution_NotInList(NewData As String, Response As Integer)
```

```
    DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
    DoCmd.OpenForm "UpdateEvolsList", A_NORMAL, , , A_ADD
    Forms![UpdateEvolsList]![Evolution] = NewData
    Response = DATA_ERRCONTINUE
End Sub

Private Sub Field22_NotInList(NewData As String, Response As Integer)
    DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
    DoCmd.OpenForm "NewComment", A_NORMAL, , , A_ADD
    Forms![NewComment]![Comment] = NewData
    Response = DATA_ERRCONTINUE
End Sub

Private Sub SaveRecord_Click()
On Error GoTo Err_SaveRecord_Click
  DoCmd.DoMenuItem A_FORMBAR, A_FILE, A_SAVERECORD, ,
A_MENU_VER20
Exit_SaveRecord_Click:
  Exit Sub
Err_SaveRecord_Click:
  MsgBox Error$
  Resume Exit_SaveRecord_Click
End Sub
```

```
Option Compare Database  'Use database order for string comparisons
Option Explicit
  Const ERR_RPT_CANCELED = 2501
Private Sub AddToWhere(FieldValue As Variant, FieldName As String, MyCriteria As
String, ArgCount As Integer)
  ' Create criteria for WHERE clause.
  If FieldValue <> "" Then
    ' Add "and" if other criterion exists.
    If ArgCount > 0 Then
      MyCriteria = MyCriteria & " and "
    End If
    ' Append criterion to existing criteria.
    ' Enclose FieldValue and asterisk in quotation marks.
    MyCriteria = (MyCriteria & FieldName & " Like " & Chr(39) & FieldValue &
Chr(42) & Chr(39))
    ' Increase argument count.
    ArgCount = ArgCount + 1
  End If
End Sub

Private Sub Battle_Stations_AfterUpdate()
  ' Disable control in detail section after changing search criterion.
```

```
    DisableControl
End Sub

Private Sub Button43_Click()
On Error GoTo Err_Button43_Click
    DoCmd.Close
Exit_Button43_Click:
    Exit Sub
Err_Button43_Click:
    MsgBox Error$
    Resume Exit_Button43_Click
End Sub

Private Sub Clear_Click()
    ' Clear controls in form header and remove records from subform.
    Dim MySQL As String
    Dim Tmp As Variant
    MySQL = "SELECT * FROM Q_FindComments WHERE False"
    ' Clear search text boxes.
    Me![Patrol] = DLast("[Patrol Number]", "Patrol")
    Me![StartDate] = DLast("[Start Date]", "Patrol")
    Me![StopDate] = DLast("[End Date]", "Patrol")
    Me![Location2].Visible = False
    Me![Location2].Enabled = False
    Me![Location].Visible = True
    Me![Location].Enabled = True
    Me![Location].Requery
    Me![RangeStart] = Me![StartDate]
    Me![RangeStop] = Me![StopDate]
    Me![DGCategory] = Null
    Me![Drill Title] = Null
    Me![Comment Type] = Null
    Me![SubComment] = Null
    Me![Patrol Stat] = Null
    Me![Watch Section] = Null
    Me![Battle Stations] = Null
    Me![Location] = Null
    Me![Location2] = Null
    Me![OOD] = Null
    Me![EOOW] = Null
    Me![COW] = Null
    Me![EWS] = Null

    ' Reset subform's RecordSource property to remove records.
        Me![Find Comments Subform].Form.RecordSource = MySQL
    ' Move insertion point to Look For Company text box.
```

```
    Me![Patrol].SetFocus
End Sub

Private Sub Comment_Type_AfterUpdate()
    Me![SubComment] = ""
    Me![SubComment].Requery
    ' Disable control in detail section after changing search criterion.
    DisableControl
End Sub

Private Sub COW_AfterUpdate()
    ' Disable control in detail section after changing search criterion.
    DisableControl
End Sub

Private Sub DGCategory_AfterUpdate()
    Me![Drill Title] = ""
    Me![Drill Title].Enabled = True
    Me![Drill Title].Requery
    ' Disable control in detail section after changing search criterion.
    DisableControl
End Sub

Private Sub DisableControl()
    ' If enabled, disable control in detail section after changing search criterion.
    Dim Tmp As Variant
    If Me![Find Comments Subform].Enabled Then
    End If
End Sub

Private Sub Drill_Title_AfterUpdate()
    If IsNull(Me![Drill Title]) Then
        Me![Location2].Visible = False
        Me![Location2].Enabled = False
        Me![Location].Visible = True
        Me![Location].Enabled = True
        Me![Location].Requery
    Else
        Me![Location].Visible = False
        Me![Location].Enabled = False
        Me![Location2].Visible = True
        Me![Location2].Enabled = True
       Me![Location2].Requery
    End If
    ' Disable control in detail section after changing search criterion.
    DisableControl
```

```
End Sub

Private Sub EOOW_AfterUpdate()
   ' Disable control in detail section after changing search criterion.
   DisableControl
End Sub

Private Sub EWS_AfterUpdate()
   ' Disable control in detail section after changing search criterion.
   DisableControl
End Sub

Private Sub Form_Load()
  Me![StartDate] = DLast("[Start Date]", "Patrol")
  Me![StopDate] = DLast("[End Date]", "Patrol")
  Me![RangeStart] = Me![StartDate]
  Me![RangeStop] = Me![StopDate]
End Sub

Private Sub Location_AfterUpdate()
   ' Disable control in detail section after changing search criterion.
   DisableControl
End Sub

Private Sub OOD_AfterUpdate()
   ' Disable control in detail section after changing search criterion.
   DisableControl
End Sub

Private Sub Patrol_AfterUpdate()
   Me![StartDate] = Me![Patrol].Column(1)
   Me![StopDate] = Me![Patrol].Column(2)
   Me![RangeStart] = Me![Patrol].Column(1)
   Me![RangeStop] = Me![Patrol].Column(2)
   Me![OOD].Requery
   Me![EOOW].Requery
   Me![COW].Requery
  Me![EWS].Requery
End Sub

Private Sub Patrol_Stat_AfterUpdate()
   ' Disable control in detail section after changing search criterion.
   DisableControl
End Sub

Private Sub RangeStart_AfterUpdate()
```

```
    Me![StartDate] = Me![RangeStart]
    Me![StopDate] = Me![RangeStart]
    Me![RangeStop] = Me![RangeStart]
End Sub

Private Sub RangeStop_AfterUpdate()
    Me![StopDate] = Me![RangeStop]
End Sub

Private Sub Show_Comments_Click()
    ' Create a WHERE clause using search criteria entered by user and
    ' set RecordSource property of Find Comments Subform.
    Dim MySQL As String, MyCriteria As String, MyRecordSource As String
    Dim ArgCount As Integer
    Dim Tmp As Variant

    ' Initialize argument count.
    ArgCount = 0
    ' Initialize SELECT statement.
    MySQL = "SELECT * FROM Q_FindComments WHERE "
    MyCriteria = ""
    ' Use values entered in text boxes in form header to create criteria for WHERE clause.
    If Me![Location].Enabled = True Then
        AddToWhere [Location], "[Location]", MyCriteria, ArgCount
    Else
        AddToWhere [Location2], "[Location]", MyCriteria, ArgCount
    End If
    AddToWhere [Patrol], "[Patrol Number]", MyCriteria, ArgCount
    AddToWhere [DGCategory], "[DrillCategoryID]", MyCriteria, ArgCount
    AddToWhere [Drill Title], "[DrillGuideNO]", MyCriteria, ArgCount
    AddToWhere [Comment Type], "[Comment Category]", MyCriteria, ArgCount
    AddToWhere [SubComment], "[Category ID]", MyCriteria, ArgCount
    AddToWhere [EOOW], "[EOOW]", MyCriteria, ArgCount
    AddToWhere [OOD], "[OOD]", MyCriteria, ArgCount
    AddToWhere [COW], "[COW]", MyCriteria, ArgCount
    AddToWhere [EWS], "[EWS]", MyCriteria, ArgCount
    AddToWhere [Patrol Stat], "[Patrol Status]", MyCriteria, ArgCount
    AddToWhere [Watch Section], "[Watch Section]", MyCriteria, ArgCount
    AddToWhere [Battle Stations], "[Battle Stations]", MyCriteria, ArgCount
    ' If no criterion specifed, return all records.
    If MyCriteria = "" Then
        MyCriteria = "True"
    End If
    ' Create SELECT statement.
    MyRecordSource = MySQL & MyCriteria
    ' Set RecordSource property of Find Comments Subform.
```

```
    Me![Find Comments Subform].Form.RecordSource = MyRecordSource
    ' If no records match criteria, display message.
    ' Move focus to Clear button.
    If Me![Find Comments Subform].Form.RecordsetClone.RecordCount = 0 Then
        MsgBox "No records match the criteria you entered.", 48, "No Records Found"
        Me!Clear.SetFocus
    Else
        ' Enable control in detail section.
        ' Move insertion point to Find Comments Subform.
        Me![Find Comments Subform].SetFocus
    End If
End Sub

Private Sub SubComment_AfterUpdate()
    ' Disable control in detail section after changing search criterion.
    DisableControl
End Sub

Private Sub Watch_Section_AfterUpdate()
    ' Disable control in detail section after changing search criterion.
    DisableControl
End Sub
```

```
Option Compare Database  ' Use database order for string comparisons.
Option Explicit
Private Sub AddToWhere(FieldValue As Variant, FieldName As String, MyCriteria As
String, ArgCount As Integer)
    ' Create criteria for WHERE clause.
    If FieldValue <> "" Then
        ' Add "and" if other criterion exists.
        If ArgCount > 0 Then
            MyCriteria = MyCriteria & " and "
        End If
        ' Append criterion to existing criteria.
        ' Enclose FieldValue and asterisk in quotation marks.
        MyCriteria = (MyCriteria & FieldName & " Like " & Chr(39) & FieldValue &
Chr(42) & Chr(39))
        ' Increase argument count.
        ArgCount = ArgCount + 1
    End If
End Sub

Private Sub Clear_Click()
    ' Clear controls in form header and remove records from subform.
```

```
    Dim MySQL As String
    Dim Tmp As Variant

    MySQL = "SELECT * FROM Customers WHERE False"
    ' Clear search text boxes.
    Me![Look For Company] = Null
    Me![Look For Contact] = Null
    Me![Look For City] = Null
    Me![Look For Country] = Null
    ' Reset subform's RecordSource property to remove records.
    Me![Find Customers Subform].Form.RecordSource = MySQL
    ' Move insertion point to Look For Company text box.
    Me![Look For Company].SetFocus
End Sub

Private Sub DisableControl()
    ' If enabled, disable control in detail section after changing search criterion.
    Dim Tmp As Variant
    If Me![Find Customers Subform].Enabled Then
        Tmp = EnableControls("Detail", False)
    End If
End Sub

Private Sub Form_Activate()
    ' Used by Solutions to show toolbar that includes Show Me button.
    ' Hide built-in Form View toolbar.
    ' Show Custom Form View toolbar.
    DoCmd.ShowToolbar "Form View", A_TOOLBAR_NO
    DoCmd.ShowToolbar "Custom Form View", A_TOOLBAR_YES
End Sub

Private Sub Form_Deactivate()
    ' Used by Solutions to hide toolbar that includes Show Me button.
    ' Hide Custom Form View toolbar.
    ' Show built-in Form View toolbar.
    DoCmd.ShowToolbar "Custom Form View", A_TOOLBAR_NO
    DoCmd.ShowToolbar "Form View", A_TOOLBAR_WHERE_APPROP
End Sub

Private Sub Form_Open(Cancel As Integer)
    ' Move insertion point to Look For Company text box when form is opened.
    Me![Look For Company].SetFocus
End Sub

Private Sub Look_For_City_AfterUpdate()
    ' Disable control in detail section after changing search criterion.
```

```
    DisableControl
End Sub

Private Sub Look_For_Company_AfterUpdate()
    ' Disable control in detail section after changing search criterion.
    DisableControl
End Sub

Private Sub Look_For_Contact_AfterUpdate()
    ' Disable control in detail section after changing search criterion.
    DisableControl
End Sub

Private Sub Look_For_Country_AfterUpdate()
    ' Disable control in detail section after changing search criterion.
    DisableControl
End Sub

Private Sub Show_Customers_Click()
    ' Create a WHERE clause using search criteria entered by user and
    ' set RecordSource property of Find Customers Subform.
    Dim MySQL As String, MyCriteria As String, MyRecordSource As String
    Dim ArgCount As Integer
    Dim Tmp As Variant
    ' Initialize argument count.
    ArgCount = 0
    ' Initialize SELECT statement.
    MySQL = "SELECT * FROM Customers WHERE "
    MyCriteria = ""
    ' Use values entered in text boxes in form header to create criteria for WHERE clause.
    AddToWhere [Look For Company], "[Company Name]", MyCriteria, ArgCount
    AddToWhere [Look For Contact], "[Contact Name]", MyCriteria, ArgCount
    AddToWhere [Look For City], "[City]", MyCriteria, ArgCount
    AddToWhere [Look For Country], "[Country]", MyCriteria, ArgCount
    ' If no criterion specifed, return all records.
    If MyCriteria = "" Then
        MyCriteria = "True"
    End If
    ' Create SELECT statement.
    MyRecordSource = MySQL & MyCriteria
    ' Set RecordSource property of Find Customers Subform.
    Me![Find Customers Subform].Form.RecordSource = MyRecordSource
    ' If no records match criteria, display message.
    ' Move focus to Clear button.
    If Me![Find Customers Subform].Form.RecordsetClone.RecordCount = 0 Then
        MsgBox "No records match the criteria you entered.", 48, "No Records Found"
```

```
    Me!Clear.SetFocus
  Else
    ' Enable control in detail section.
    Tmp = EnableControls("Detail", True)
    ' Move insertion point to Find Customers Subform.
    Me![Find Customers Subform].SetFocus
  End If
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button65_Click()
On Error GoTo Err_Button65_Click
  DoCmd.Quit
Exit_Button65_Click:
  Exit Sub
Err_Button65_Click:
  MsgBox Error$
  Resume Exit_Button65_Click
End Sub

Private Sub Button66_Click()
On Error GoTo Err_Button66_Click
  DoCmd.DoMenuItem    A_FORMBAR,    A_FILE,    A_SAVERECORD,    ,
A_MENU_VER20
Exit_Button66_Click:
  Exit Sub
Err_Button66_Click:
  MsgBox Error$
  Resume Exit_Button66_Click
End Sub

Private Sub Close_Click()
On Error GoTo Err_Close_Click
  DoCmd.Close
Exit_Close_Click:
  Exit Sub
Err_Close_Click:
  MsgBox Error$
  Resume Exit_Close_Click
End Sub

Private Sub Comment_NotInList(NewData As String, Response As Integer)
    DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
    DoCmd.OpenForm "NewComment", A_NORMAL, , , A_ADD
```

```
    Forms![NewComment]![Comment] = NewData
    Response = DATA_ERRCONTINUE
End Sub

Private Sub Comment_Type_AfterUpdate()
  Me![SubComment] = ""
  Me![SubComment].Enabled = True
  Me![SubComment].Requery
End Sub

Private Sub Comment_Type_Enter()
  Me![SubComment] = ""
  Me![SubComment].Enabled = True
  Me![SubComment].Requery
End Sub

Private Sub DG_Num_AfterUpdate()
  Me![Location].Requery
  Me![Drill Anomalies].Form![Anomaly].Requery
End Sub

Private Sub EnterComments_Click()
On Error GoTo Err_EnterComments_Click
  Dim DocName As String
  Dim LinkCriteria As String
  DocName = "SelectComments"
  LinkCriteria = "[Drill ID] = Forms![Input Comments]![Drill ID]"
  DoCmd.OpenForm DocName, , , LinkCriteria
Exit_EnterComments_Click:
  Exit Sub

Err_EnterComments_Click:
  MsgBox Error$
  Resume Exit_EnterComments_Click

End Sub

Private Sub Form_Open(Cancel As Integer)
  Dim M_PtlNum, M_PtlStatus As Integer
  Dim M_Date As Variant
  Dim M_LastWS, M_DGNum As String
End Sub

Private Sub Next_Click()
On Error GoTo Err_Next_Click
  DoCmd.GoToRecord , , A_NEXT
```

```
    Me![Drill Anomalies].Form![Selected Anomalies].Requery
    [Forms]![Input Comments]![DG Num].SetFocus
Exit_Next_Click:
    Exit Sub
Err_Next_Click:
    MsgBox Error$
    Resume Exit_Next_Click
End Sub

Private Sub Patrol_Number_AfterUpdate()
    Me![OOD].Requery
    Me![EOOW].Requery
    Me![COW].Requery
    Me![EWS].Requery
End Sub

Private Sub Patrol_Number_NotInList(NewData As String, Response As Integer)
        DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
        DoCmd.OpenForm "NewPatrol", A_NORMAL, , , A_ADD
        Forms![NewPatrol]![Patrol Number] = NewData
        Response = DATA_ERRCONTINUE

End Sub

Private Sub PatrolNumber_NotInList(NewData As String, Response As Integer)
        DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
        DoCmd.OpenForm "NewPatrol", A_NORMAL, , , A_ADD
        Forms![NewPatrol]![Patrol Number] = NewData
        Response = DATA_ERRCONTINUE
End Sub

Private Sub Save_Exit(Cancel As Integer)
    Me![SubComment] = ""
End Sub

Private Sub SubCategory_Enter()
    Me![SubCategory] = ""
    Me![SubCategory].Enabled = True
    Me![SubCategory].Requery

End Sub

Private Sub SubComment_AfterUpdate()
   Me![Comment].Requery
End Sub
```

```
Private Sub SubComment_NotInList(NewData As String, Response As Integer)
    M_COMMENT = Me![Comment Type]
    DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
    DoCmd.OpenForm "NewSubComment", A_NORMAL, , , A_ADD
    Forms![NewSubComment]![Comment SubType] = NewData
    Response = DATA_ERRCONTINUE
End Sub

Private Sub temp_Enter()
  Me![Comment].SetFocus
  Me![Comment].Requery
End Sub

Private Sub test_Enter()
  Me![Comment].SetFocus
  Me![Comment].Requery
End Sub

Private Sub Test1_Enter()
  Me![Comment].SetFocus
  Me![Comment].Requery
End Sub

Private Sub Watch_Sta__NotInList(NewData As String, Response As Integer)
  Dim NewWS As Integer, Title As String
  DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD, ,
A_MENU_VER20
  DoCmd.OpenForm "NewWatchStation", A_NORMAL, , , A_ADD
  Forms![NewWatchStaion]![NewWS] = NewData
  Response = DATA_ERRCONTINUE
End Sub

Private Sub Watch_Station_Change()
    DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
    DoCmd.OpenForm "NewWatchStation", A_NORMAL, , , A_ADD
    Forms![NewWatchStation]![Watch Station] = NewData
End Sub

Private Sub Watch_Station_NotInList(NewData As String, Response As Integer)
    DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
    DoCmd.OpenForm "NewWatchStation", A_NORMAL, , , A_ADD
    Forms![NewWatchStation]![Watch Station] = NewData
    Response = DATA_ERRCONTINUE
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button104_Click()
On Error GoTo Err_Button104_Click
   Dim DocName As String
   Dim LinkCriteria As String
   LinkCriteria = "[Drill ID] = Forms![Input Comments2]![Drill ID]"
   DocName = "View Comments"
   DoCmd.OpenForm DocName, A_NORMAL, , LinkCriteria, A_READONLY
Exit_Button104_Click:
   Exit Sub
Err_Button104_Click:
   MsgBox Error$
   Resume Exit_Button104_Click
End Sub

Private Sub Button65_Click()
On Error GoTo Err_Button65_Click
   DoCmd.Quit
Exit_Button65_Click:
   Exit Sub
Err_Button65_Click:
   MsgBox Error$
   Resume Exit_Button65_Click
End Sub

Private Sub Button66_Click()
On Error GoTo Err_Button66_Click
   DoCmd.DoMenuItem     A_FORMBAR,     A_FILE,     A_SAVERECORD,     ,
A_MENU_VER20
Exit_Button66_Click:
   Exit Sub
Err_Button66_Click:
   MsgBox Error$
   Resume Exit_Button66_Click
End Sub

Private Sub Close_Click()
On Error GoTo Err_Close_Click
   DoCmd.Close
Exit_Close_Click:
   Exit Sub
Err_Close_Click:
   MsgBox Error$
   Resume Exit_Close_Click
End Sub
```

```
Private Sub Comment_NotInList(NewData As String, Response As Integer)
    DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
    DoCmd.OpenForm "NewComment", A_NORMAL, , , A_ADD
    Forms![NewComment]![Comment] = NewData
    Response = DATA_ERRCONTINUE
End Sub

Private Sub Comment_Type_AfterUpdate()
  Me![SubComment] = ""
  Me![SubComment].Enabled = True
  Me![SubComment].Requery
End Sub

Private Sub Comment_Type_Enter()
  Me![SubComment] = ""
  Me![SubComment].Enabled = True
  Me![SubComment].Requery
End Sub

Private Sub EnterComments_Click()
On Error GoTo Err_EnterComments_Click
  Dim DocName As String
  Dim LinkCriteria As String
  DocName = "SelectComments"
  LinkCriteria = "[Drill ID] = Forms![Input Comments]![Drill ID]"
  DoCmd.OpenForm DocName, , , LinkCriteria
Exit_EnterComments_Click:
  Exit Sub
Err_EnterComments_Click:
  MsgBox Error$
  Resume Exit_EnterComments_Click
End Sub

Private Sub Field61_NotInList(NewData As String, Response As Integer)
End Sub
Private Sub Next_Click()
On Error GoTo Err_Next_Click
  DoCmd.GoToRecord , , A_NEXT
Exit_Next_Click:
  Exit Sub
Err_Next_Click:
  MsgBox Error$
  Resume Exit_Next_Click
End Sub
```

```
Private Sub Patrol_Number_NotInList(NewData As String, Response As Integer)
    DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
    DoCmd.OpenForm "NewPatrol", A_NORMAL, , , A_ADD
    Forms![NewPatrol]![Patrol Number] = NewData
    Response = DATA_ERRCONTINUE
End Sub

Private Sub PatrolNumber_NotInList(NewData As String, Response As Integer)
    DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
    DoCmd.OpenForm "NewPatrol", A_NORMAL, , , A_ADD
    Forms![NewPatrol]![Patrol Number] = NewData
    Response = DATA_ERRCONTINUE
End Sub

Private Sub Save_Exit(Cancel As Integer)
   Me![SubComment] = ""
End Sub

Private Sub SubCategory_Enter()
  Me![SubCategory] = ""
  Me![SubCategory].Enabled = True
  Me![SubCategory].Requery
End Sub

Private Sub SubComment_AfterUpdate()
 Me![Comment].Requery
End Sub

Private Sub SubComment_NotInList(NewData As String, Response As Integer)
    M_COMMENT = Me![Comment Type]
    DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
    DoCmd.OpenForm "NewSubComment", A_NORMAL, , , A_ADD
    Forms![NewSubComment]![Comment SubType] = NewData
    Response = DATA_ERRCONTINUE
End Sub

Private Sub temp_Enter()
  Me![Comment].SetFocus
  Me![Comment].Requery
End Sub

Private Sub test_Enter()
  Me![Comment].SetFocus
  Me![Comment].Requery
End Sub
```

```
Private Sub Test1_Enter()
   Me![Comment].SetFocus
   Me![Comment].Requery
End Sub

Private Sub Watch_Sta__NotInList(NewData As String, Response As Integer)
   Dim NewWS As Integer, Title As String
   DoCmd.DoMenuItem      A_FORMBAR,      A_EDIT,      A_UNDOFIELD,      ,
A_MENU_VER20
   DoCmd.OpenForm "NewWatchStation", A_NORMAL, , , A_ADD
   Forms![NewWatchStaion]![NewWS] = NewData
   Response = DATA_ERRCONTINUE
End Sub

Private Sub Watch_Station_Change()
      DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
      DoCmd.OpenForm "NewWatchStation", A_NORMAL, , , A_ADD
      Forms![NewWatchStation]![Watch Station] = NewData
End Sub

Private Sub Watch_Station_NotInList(NewData As String, Response As Integer)
      DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
      DoCmd.OpenForm "NewWatchStation", A_NORMAL, , , A_ADD
      Forms![NewWatchStation]![Watch Station] = NewData
      Response = DATA_ERRCONTINUE
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button31_Click()
On Error GoTo Err_Button31_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "Evolutions Switchboard"
   DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button31_Click:
   Exit Sub
Err_Button31_Click:
   MsgBox Error$
   Resume Exit_Button31_Click
End Sub

Private Sub Button32_Click()
On Error GoTo Err_Button32_Click
   Dim DocName As String
```

```
    Dim LinkCriteria As String
    DocName = "DrillComments Switchboard"
    DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button32_Click:
    Exit Sub
Err_Button32_Click:
    MsgBox Error$
    Resume Exit_Button32_Click
End Sub

Private Sub Button33_Click()
On Error GoTo Err_Button33_Click
    Dim DocName As String
    Dim LinkCriteria As String
    DocName = "Reports Switchboard"
    DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button33_Click:
    Exit Sub
Err_Button33_Click:
    MsgBox Error$
    Resume Exit_Button33_Click
End Sub

Private Sub Button34_Click()
On Error GoTo Err_Button34_Click
    Dim DocName As String
    Dim LinkCriteria As String
    DocName = "Maintenance Switchboard"
    DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button34_Click:
    Exit Sub
Err_Button34_Click:
    MsgBox Error$
    Resume Exit_Button34_Click
End Sub

Private Sub Button35_Click()
On Error GoTo Err_Button35_Click
    DoCmd.Quit
Exit_Button35_Click:
    Exit Sub
Err_Button35_Click:
    MsgBox Error$
    Resume Exit_Button35_Click
End Sub
```

```
Private Sub Button36_Click()
On Error GoTo Err_Button36_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "Query Switchboard"
   DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button36_Click:
   Exit Sub
Err_Button36_Click:
   MsgBox Error$
   Resume Exit_Button36_Click
End Sub

Private Sub Button37_Click()
On Error GoTo Err_Button37_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "Drill/Evolutions Switchboard"
   DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button37_Click:
   Exit Sub
Err_Button37_Click:
   MsgBox Error$
   Resume Exit_Button37_Click
End Sub

Private Sub CommentList_Click()
On Error GoTo Err_CommentList_Click
   Dim DocName As String
   DocName = "CommentInput"
   DoCmd.OpenForm DocName, A_NORMAL, , , A_ADD
Exit_CommentList_Click:
   Exit Sub
Err_CommentList_Click:
   MsgBox Error$
   Resume Exit_CommentList_Click
End Sub

Private Sub DrillComments_Click()
On Error GoTo Err_DrillComments_Click
   Dim DocName As String
   DocName = "Input Comments"
   DoCmd.OpenForm DocName, A_NORMAL, , , A_ADD
Exit_DrillComments_Click:
   Exit Sub
Err_DrillComments_Click:
```

```
    MsgBox Error$
    Resume Exit_DrillComments_Click
End Sub

Private Sub DrillGuides_Click()
On Error GoTo Err_DrillGuides_Click
    Dim DocName As String
    DocName = "New DrillGuide"
    DoCmd.OpenForm DocName, A_NORMAL, , , A_ADD
Exit_DrillGuides_Click:
    Exit Sub
Err_DrillGuides_Click:
    MsgBox Error$
    Resume Exit_DrillGuides_Click
End Sub

Private Sub View_Drills_Click()
On Error GoTo Err_View_Drills_Click
    Dim DocName As String
    Dim LinkCriteria As String
    DocName = "View Comments"
    DoCmd.OpenForm DocName, , , A_READONLY
Exit_View_Drills_Click:
    Exit Sub
Err_View_Drills_Click:
    MsgBox Error$
    Resume Exit_View_Drills_Click
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button32_Click()
On Error GoTo Err_Button32_Click
    Dim DocName As String
    Dim LinkCriteria As String
    DocName = "EvolutionsPerformed"
    DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button32_Click:
    Exit Sub
Err_Button32_Click:
    MsgBox Error$
    Resume Exit_Button32_Click
End Sub

Private Sub Button33_Click()
```

```
On Error GoTo Err_Button33_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "UpdateEvolsList"
   DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button33_Click:
   Exit Sub
Err_Button33_Click:
   MsgBox Error$
   Resume Exit_Button33_Click
End Sub

Private Sub Button34_Click()
On Error GoTo Err_Button34_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "UpdateCrossRefTable"
   DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button34_Click:
   Exit Sub
Err_Button34_Click:
   MsgBox Error$
   Resume Exit_Button34_Click
End Sub

Private Sub Button35_Click()
On Error GoTo Err_Button35_Click
   DoCmd.Close
Exit_Button35_Click:
   Exit Sub
Err_Button35_Click:
   MsgBox Error$
   Resume Exit_Button35_Click
End Sub

Private Sub Button36_Click()
On Error GoTo Err_Button36_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "UpdateWatchStanderList"
   DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button36_Click:
   Exit Sub
Err_Button36_Click:
   MsgBox Error$
   Resume Exit_Button36_Click
```

```
End Sub

Private Sub Button37_Click()
On Error GoTo Err_Button37_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "UpdateCommentSubCategory"
   DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button37_Click:
   Exit Sub
Err_Button37_Click:
   MsgBox Error$
   Resume Exit_Button37_Click
End Sub

Private Sub Button38_Click()
On Error GoTo Err_Button38_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "UpdateCommentsList"
   DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button38_Click:
   Exit Sub
Err_Button38_Click:
   MsgBox Error$
   Resume Exit_Button38_Click
End Sub

Private Sub Button39_Click()
On Error GoTo Err_Button39_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "UpdatePatrolInfo"
   DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button39_Click:
   Exit Sub
Err_Button39_Click:
   MsgBox Error$
   Resume Exit_Button39_Click
End Sub

Private Sub CommentList_Click()
On Error GoTo Err_CommentList_Click
   Dim DocName As String
   DocName = "CommentInput"
   DoCmd.OpenForm DocName, A_NORMAL, , , A_ADD
```

```
Exit_CommentList_Click:
   Exit Sub
Err_CommentList_Click:
   MsgBox Error$
   Resume Exit_CommentList_Click
End Sub

Private Sub DrillComments_Click()
On Error GoTo Err_DrillComments_Click
   Dim DocName As String
   DocName = "Input Comments"
   DoCmd.OpenForm DocName, A_NORMAL, , , A_ADD
Exit_DrillComments_Click:
   Exit Sub
Err_DrillComments_Click:
   MsgBox Error$
   Resume Exit_DrillComments_Click
End Sub

Private Sub DrillGuides_Click()
On Error GoTo Err_DrillGuides_Click
   Dim DocName As String
   DocName = "New DrillGuide"
   DoCmd.OpenForm DocName, A_NORMAL, , , A_ADD
Exit_DrillGuides_Click:
   Exit Sub
Err_DrillGuides_Click:
   MsgBox Error$
   Resume Exit_DrillGuides_Click
End Sub

Private Sub View_Drills_Click()
On Error GoTo Err_View_Drills_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "View Comments"
   DoCmd.OpenForm DocName, , , A_READONLY
Exit_View_Drills_Click:
   Exit Sub
Err_View_Drills_Click:
   MsgBox Error$
   Resume Exit_View_Drills_Click
End Sub
```

```
Option Compare Database   'Use database order for string comparisons
```

```
Private Sub Close_Click()
On Error GoTo Err_Close_Click
   DoCmd.Close
Exit_Close_Click:
   Exit Sub
Err_Close_Click:
   MsgBox Error$
   Resume Exit_Close_Click
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Category_AfterUpdate()
    [New DG Subform].Requery
    [New DG Subform].Enabled = True
End Sub

Private Sub Category_NotInList(NewData As String, Response As Integer)
      DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
      DoCmd.OpenForm "New Category", A_NORMAL, , , A_ADD
      Forms![New Category]![Category] = NewData
      Response = DATA_ERRCONTINUE
End Sub

Private Sub Close_Click()
On Error GoTo Err_Close_Click
   DoCmd.Close
Exit_Close_Click:
   Exit Sub
Err_Close_Click:
   MsgBox Error$
   Resume Exit_Close_Click
End Sub

Private Sub FwdAft_Enter()
   Me![FwdAft] = True
   Me![SelectCategory] = Null
   [New DG Subform].Requery
End Sub

Private Sub FwdAft_Exit(Cancel As Integer)
      Me![SelectCategory].Requery
End Sub
Private Sub SelectCategory_AfterUpdate()
```

```
    [New DG Subform].Enabled = True
    [New DG Subform].Requery
    [New DG Subform].SetFocus
End Sub

Private Sub SelectCategory_GotFocus()
    Me![SelectCategory] = Null
    [New DG Subform].Requery
End Sub

Private Sub SelectCategory_NotInList(NewData As String, Response As Integer)
        DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
        DoCmd.OpenForm "New Category", A_NORMAL, , , A_ADD
        Forms![New Category]![Category] = NewData
        Response = DATA_ERRCONTINUE
End Sub

Private Sub Word_Click()
On Error GoTo Err_Word_Click

    Dim x As Integer
    Dim WordObj As Object
'   Set wordobj = CreateObject("word.document")
    x = Shell("c:\msoffice\winword\WINWORD.EXE", 1)
    WordObj.Fileopen "c:\drilbase\4000.doc"
Exit_Word_Click:
    Exit Sub
Err_Word_Click:
    MsgBox Error$
    Resume Exit_Word_Click
End Sub

Private Sub WORD1_Click()
On Error GoTo Err_WORD1_Click
    Dim x As Integer
    Dim FileName As String
    Dim WordObj As Object
    Set WordObj = CreateObject("Word.Basic")
    FileName = "C:\Drilbase\" + Me![New DG Subform].Form![DrillGuideNo] + ".DOC"
    WordObj.Fileopen FileName
    WordObj.EditSelect
    Response = DATA_ERRCONTINUE
Exit_WORD1_Click:
   Exit Sub
Err_WORD1_Click:
    MsgBox Error$
```

```
    Resume Exit_WORD1_Click
End Sub

Private Sub Word2_Click()
On Error GoTo Err_WORD2_Click
    Dim x As Integer
    Dim FileName As String
    Dim WordObj As Object
    Set WordObj = CreateObject("Word.Basic")
    FileName = "C:\Drilbase\" + Me![New DG Subform].Form![DrillGuideNo] + ".DOC"
    WordObj.Fileopen FileName
    WordObj.EditSelect
    Response = DATA_ERRCONTINUE
Exit_WORD2_Click:
   Exit Sub
Err_WORD2_Click:
    MsgBox Error$
    Resume Exit_WORD1_Click
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Close_Click()
On Error GoTo Err_Close_Click
    DoCmd.Close
Exit_Close_Click:
    Exit Sub
Err_Close_Click:
    MsgBox Error$
    Resume Exit_Close_Click
End Sub

Private Sub Form_Unload(Cancel As Integer)
    Dim MyControl As Control
    Set MyControl = Forms![Input Comments]![Drill Anomalies].Form![Anomaly]
    DoCmd.SelectObject A_FORM, "Input Comments"
    MyControl.Requery
    MyControl = Forms![Input Comments]![Drill Anomalies].Form![Anomaly]
    Forms![Input Comments]![Drill Anomalies].Form![Anomaly] = Me![Anomaly ID]
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Close_Click()
```

```
On Error GoTo Err_Close_Click
   DoCmd.Close
Exit_Close_Click:
   Exit Sub
Err_Close_Click:
   MsgBox Error$
   Resume Exit_Close_Click
End Sub

Private Sub Form_Unload(Cancel As Integer)
   ' If Input Comments form is loaded,
   ' select it, requery Watch Station combo box,
   ' and set value of Watch Station combo box.
   Dim MyControl As Control
      Set MyControl = Forms![Input Comments]![SelectComments].Form![Comment]
      DoCmd.SelectObject A_FORM, "Input Comments"
      MyControl.Requery
      MyControl = Forms![Input Comments]![SelectComments].Form![Comment]
      Forms![Input  Comments]![SelectComments].Form![Comment]  =  Me![Comment
ID]
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Close_Click()
On Error GoTo Err_Close_Click
   DoCmd.Close
Exit_Close_Click:
   Exit Sub
Err_Close_Click:
   MsgBox Error$
   Resume Exit_Close_Click
End Sub

Private Sub Form_Unload(Cancel As Integer)
   Dim MyControl As Control
      Set MyControl = Forms![Input Comments2]![SelectComments2].Form![Comment]
      DoCmd.SelectObject A_FORM, "Input Comments2"
      MyControl.Requery
      MyControl = Forms![Input Comments2]![SelectComments2].Form![Comment]
      Forms![Input  Comments2]![SelectComments2].Form![Comment]  =  Me![Comment
ID]
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button20_Click()
On Error GoTo Err_Button20_Click
   DoCmd.Close
Exit_Button20_Click:
   Exit Sub
Err_Button20_Click:
   MsgBox Error$
   Resume Exit_Button20_Click
End Sub

Private Sub Form_Unload(Cancel As Integer)
   Dim MyControl As Control
   If IsLoaded("Input Comments") Then
      Set MyControl = Forms![Input Comments]![Patrol Number]
      DoCmd.SelectObject A_FORM, "Input Comments"
      MyControl.Requery
      MyControl = Forms![Input Comments]![Patrol Number]
      Forms![Input Comments]![Patrol Number] = Me![Patrol Number]
  End If
End Sub

Private Sub Patrol_Number_AfterUpdate()
   If IsLoaded("Input Comments") Then
      Set MyControl = Forms![Input Comments]![Patrol Number]
      DoCmd.SelectObject A_FORM, "Input Comments"
      MyControl.Requery
      MyControl = Forms![Input Comments]![Patrol Number]
      Forms![Input Comments]![Patrol Number] = Me![Patrol Number]
   End If
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Close_Click()
On Error GoTo Err_Close_Click
   DoCmd.Close
Exit_Close_Click:
   Exit Sub
Err_Close_Click:
   MsgBox Error$
   Resume Exit_Close_Click
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
   ' If Input Comments form is loaded,
   ' select it, requery Watch Station combo box,
   ' and set value of Watch Station combo box.
   Dim MyControl As Control
   Set MyControl = Forms![Input Comments]![SelectComments].Form![SubComment]
   DoCmd.SelectObject A_FORM, "Input Comments"
   MyControl.Requery
   MyControl = Forms![Input Comments]![SelectComments].Form![SubComment]
   Forms![Input  Comments]![SelectComments].Form![SubComment]  =  Me![Category
ID]
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Close_Click()
On Error GoTo Err_Close_Click
   DoCmd.Close
Exit_Close_Click:
   Exit Sub
Err_Close_Click:
   MsgBox Error$
   Resume Exit_Close_Click
End Sub

Private Sub Form_Unload(Cancel As Integer)
   ' If Input Comments form is loaded,
   ' select it, requery Watch Station combo box,
   ' and set value of Watch Station combo box.
   Dim MyControl As Control
   Set MyControl = Forms![UpdateCommentsList]![SubComment]
   DoCmd.SelectObject A_FORM, "UpdateCommentsList"
   MyControl.Requery
   MyControl = Forms![UpdateCommentsList]![SubComment]
   Forms![UpdateCommentsList]![SubComment] = Me![Category ID]
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Close_Click()
On Error GoTo Err_Close_Click
   DoCmd.Close
Exit_Close_Click:
   Exit Sub
```

```
Err_Close_Click:
   MsgBox Error$
   Resume Exit_Close_Click
End Sub

Private Sub Form_Unload(Cancel As Integer)
   ' If Input Comments2 form is loaded,
   ' select it, requery Watch Station combo box,
   ' and set value of Watch Station combo box.
   Dim MyControl As Control
   Set MyControl = Forms![Input Comments2]![SelectComments2].Form![SubComment]
   DoCmd.SelectObject A_FORM, "Input Comments2"
   MyControl.Requery
   MyControl = Forms![Input Comments2]![SelectComments2].Form![SubComment]
   Forms![Input Comments2]![SelectComments2].Form![SubComment] = Me![Category
ID]
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Close_Click()
On Error GoTo Err_Close_Click
   DoCmd.Close
Exit_Close_Click:
   Exit Sub
Err_Close_Click:
   MsgBox Error$
   Resume Exit_Close_Click
End Sub

Private Sub Form_Unload(Cancel As Integer)
' If Input Comments form is loaded,
' select it, requery Watch Station combo box,
' and set value of Watch Station combo box.
   Dim MyControl As Control
   Set MyControl = Forms![Input Comments]![SelectComments].Form![Watch Station]
   DoCmd.SelectObject A_FORM, "Input Comments"
   MyControl.Requery
   MyControl = Forms![Input Comments]![SelectComments].Form![Watch Station]
   Forms![Input Comments]![SelectComments].Form![Watch Station] = Me![Watch
Station ID]
End Sub
```

```
Option Compare Database   'Use database order for string comparisons
```

```
Private Sub Close_Click()
On Error GoTo Err_Close_Click
   DoCmd.Close
Exit_Close_Click:
   Exit Sub
Err_Close_Click:
   MsgBox Error$
   Resume Exit_Close_Click
End Sub

Private Sub Form_Unload(Cancel As Integer)
   Dim MyControl As Control
   Set MyControl = Forms![Input Comments2]![SelectComments2].Form![Watch
Station]
   DoCmd.SelectObject A_FORM, "Input Comments2"
   MyControl.Requery
   MyControl = Forms![Input Comments2]![SelectComments2].Form![Watch Station]
   Forms![Input Comments2]![SelectComments2].Form![Watch Station] = Me![Watch
Station ID]
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button35_Click()
On Error GoTo Err_Button35_Click
   DoCmd.Close
Exit_Button35_Click:
   Exit Sub
Err_Button35_Click:
   MsgBox Error$
   Resume Exit_Button35_Click
End Sub

Private Sub Button38_Click()
On Error GoTo Err_Button38_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "Find Comments"
   DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button38_Click:
   Exit Sub
Err_Button38_Click:
   MsgBox Error$
   Resume Exit_Button38_Click
```

```
End Sub
```

```
Option Compare Database   'Use database order for string comparisons
Option Explicit

Const ERR_RPT_CANCELED = 2501
Private Sub Cancel_Click()
' Close form.  (This code created by Command Button Wizard.)
On Error GoTo Err_Cancel_Click
   DoCmd.Close
Exit_Cancel_Click:
   Exit Sub
Err_Cancel_Click:
   MsgBox Error$
   Resume Exit_Cancel_Click
End Sub

Private Sub Form_Load()
   ' Used by Solutions to return records for example.
   '
   ' Enter parameters in Beginning Date and Ending Date text boxes when
   ' form is loaded.
   Me![StartDate] = #1/2/1993#
   Me![StopDate] = #1/16/1993#
End Sub

Private Sub Patrol_AfterUpdate()
   Me![StartDate] = Me![Patrol].Column(1)
   Me![StopDate] = Me![Patrol].Column(2)
End Sub

Private Sub Preview_Click()
' Preview report.
On Error GoTo Err_Preview_Click

   Dim DocName As String
   DocName = "EngDrillLogReport"
   DoCmd.OpenReport DocName$, 2
Exit_Preview_Click:
   Exit Sub
Err_Preview_Click:
   If Err = ERR_RPT_CANCELED Then
      Resume Exit_Preview_Click
   Else
      MsgBox Error$
```

```
      Resume Exit_Preview_Click
    End If
End Sub

Private Sub Print_Click()
' Print report.
On Error GoTo Err_Print_Click
    Dim DocName As String
    DocName = "EngDrillLogReport"
    DoCmd.OpenReport DocName$, 0
Exit_Print_Click:
    Exit Sub
Err_Print_Click:
    If Err = ERR_RPT_CANCELED Then
      Resume Exit_Print_Click
    Else
      MsgBox Error$
      Resume Exit_Print_Click
    End If
End Sub

Private Sub RangeStart_AfterUpdate()
    Me![StartDate] = Me![RangeStart]
    Me![StopDate] = Me![RangeStart]
    Me![RangeStop] = Me![RangeStart]
End Sub

Private Sub RangeStop_AfterUpdate()
    Me![StopDate] = Me![RangeStop]
End Sub

Private Sub Report_Drills_AfterUpdate()
Select Case [Report Drills]
    Case 1
      Me![Fwd/Aft] = True
    Case 2
      Me![Fwd/Aft] = False
End Select
End Sub

Private Sub RptDates_AfterUpdate()
If RptDates = 1 Then
    Me![Patrol].Enabled = True
    Me![RangeStart].Enabled = False
    Me![RangeStop].Enabled = False
End If
```

```
If RptDates = 2 Then
   Me![RangeStart].Enabled = True
   Me![RangeStop].Enabled = True
   Me![Patrol].Enabled = False
End If
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button35_Click()
On Error GoTo Err_Button35_Click
   DoCmd.Close
Exit_Button35_Click:
   Exit Sub
Err_Button35_Click:
   MsgBox Error$
   Resume Exit_Button35_Click
End Sub

Private Sub Button36_Click()
On Error GoTo Err_Button36_Click
   Dim DocName As String
   DocName = "ORSE Drill List"
   DoCmd.OpenReport DocName, A_PREVIEW
Exit_Button36_Click:
   Exit Sub
Err_Button36_Click:
   MsgBox Error$
   Resume Exit_Button36_Click
End Sub

Private Sub Button37_Click()
On Error GoTo Err_Button37_Click
   Dim DocName As String
   DocName = "Drill/EvolCrossRef"
   DoCmd.OpenReport DocName, A_PREVIEW
Exit_Button37_Click:
   Exit Sub
Err_Button37_Click:
   MsgBox Error$
   Resume Exit_Button37_Click
End Sub

Private Sub Button38_Click()
On Error GoTo Err_Button38_Click
```

```
    Dim DocName As String
    Dim LinkCriteria As String
    DocName = "ReportParameters Dialog Box"
    DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button38_Click:
    Exit Sub
Err_Button38_Click:
    MsgBox Error$
    Resume Exit_Button38_Click
End Sub

Private Sub Button42_Click()
On Error GoTo Err_Button42_Click
    Dim DocName As String
    DocName = "Drills/Evol Cross Ref Report"
    DoCmd.OpenReport DocName, A_PREVIEW
Exit_Button42_Click:
    Exit Sub
Err_Button42_Click:
    MsgBox Error$
    Resume Exit_Button42_Click
End Sub

Private Sub Button43_Click()
On Error GoTo Err_Button43_Click
    Dim DocName As String
    Dim LinkCriteria As String
    DocName = "Drill/Evolutions Dialog Box"
    DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button43_Click:
    Exit Sub
Err_Button43_Click:
    MsgBox Error$
    Resume Exit_Button43_Click
End Sub

Private Sub Button44_Click()
On Error GoTo Err_Button44_Click
    Dim DocName As String
    Dim LinkCriteria As String
    DocName = "Select Patrol2"
    DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button44_Click:
    Exit Sub
Err_Button44_Click:
    MsgBox Error$
```

```
    Resume Exit_Button44_Click
End Sub

Private Sub Drill_Critique_Click()
On Error GoTo Err_Drill_Critique_Click
    Dim DocName As String
    Dim LinkCriteria As String
    DocName = "Drill Critique Dialog Box"
    DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Drill_Critique_Click:
    Exit Sub
Err_Drill_Critique_Click:
    MsgBox Error$
    Resume Exit_Drill_Critique_Click
End Sub

Private Sub EvolutionsPerformed_Click()
On Error GoTo Err_EvolutionsPerformed_Click
    Dim DocName As String
    Dim LinkCriteria As String
    DocName = "Evolutions Performed Dialog Box"
    DoCmd.OpenForm DocName, , , LinkCriteria
Exit_EvolutionsPerformed_Click:
    Exit Sub
Err_EvolutionsPerformed_Click:
    MsgBox Error$
    Resume Exit_EvolutionsPerformed_Click
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button40_Click()
On Error GoTo Err_Button40_Click
    Dim DocName As String
    Dim LinkCriteria As String
    DocName = "Input Comments2"
    LinkCriteria = "[Drill ID] = Forms![Select Drill]![SelectedDrill]"
    DoCmd.OpenForm DocName, , , LinkCriteria
Exit_Button40_Click:
    Exit Sub
Err_Button40_Click:
    MsgBox Error$
    Resume Exit_Button40_Click
End Sub
```

```
Private Sub Button41_Click()
On Error GoTo Err_Button41_Click
   DoCmd.Close
Exit_Button41_Click:
   Exit Sub
Err_Button41_Click:
   MsgBox Error$
   Resume Exit_Button41_Click
End Sub

Private Sub Patrol_Number_AfterUpdate()
   Me![SelectedDrill].Requery
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button40_Click()
On Error GoTo Err_Button40_Click
   Dim DocName As String
   Dim LinkCriteria As String
   DocName = "View Comments"
   LinkCriteria = "[Patrol Number] = Forms![Select Patrol]![Patrol Number]"
   DoCmd.OpenForm DocName, A_NORMAL, , LinkCriteria, A_READONLY
Exit_Button40_Click:
   Exit Sub
Err_Button40_Click:
   MsgBox Error$
   Resume Exit_Button40_Click
End Sub

Private Sub Button41_Click()
On Error GoTo Err_Button41_Click
   DoCmd.Close
Exit_Button41_Click:
   Exit Sub
Err_Button41_Click:
   MsgBox Error$
   Resume Exit_Button41_Click
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Option Explicit
Const ERR_RPT_CANCELED = 2501
```

```
Private Sub Cancel_Click()
' Close form. (This code created by Command Button Wizard.)
On Error GoTo Err_Cancel_Click
   DoCmd.Close
Exit_Cancel_Click:
   Exit Sub
Err_Cancel_Click:
   MsgBox Error$
   Resume Exit_Cancel_Click
End Sub

Private Sub Preview_Click()
' Preview report.
On Error GoTo Err_Preview_Click
   Dim DocName As String
   DocName = "Recurring Deficiencies"
   DoCmd.OpenReport DocName$, 2
Exit_Preview_Click:
   Exit Sub
Err_Preview_Click:
   If Err = ERR_RPT_CANCELED Then
      Resume Exit_Preview_Click
   Else
      MsgBox Error$
      Resume Exit_Preview_Click
   End If
End Sub

Private Sub Print_Click()
' Print report.
On Error GoTo Err_Print_Click
   Dim DocName As String
   DocName = "Recurring Deficiencies"
   DoCmd.OpenReport DocName$, 0
Exit_Print_Click:
   Exit Sub
Err_Print_Click:
   If Err = ERR_RPT_CANCELED Then
      Resume Exit_Print_Click
   Else
      MsgBox Error$
      Resume Exit_Print_Click
   End If
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button65_Click()
On Error GoTo Err_Button65_Click
   DoCmd.Quit
Exit_Button65_Click:
   Exit Sub
Err_Button65_Click:
   MsgBox Error$
   Resume Exit_Button65_Click
End Sub

Private Sub Button66_Click()
On Error GoTo Err_Button66_Click
   DoCmd.DoMenuItem      A_FORMBAR,      A_FILE,      A_SAVERECORD,      ,
A_MENU_VER20
Exit_Button66_Click:
   Exit Sub
Err_Button66_Click:
   MsgBox Error$
   Resume Exit_Button66_Click
End Sub

Private Sub Close_Click()
On Error GoTo Err_Close_Click
   DoCmd.Close
Exit_Close_Click:
   Exit Sub
Err_Close_Click:
   MsgBox Error$
   Resume Exit_Close_Click
End Sub

Private Sub Comment_NotInList(NewData As String, Response As Integer)
      DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
      DoCmd.OpenForm "NewComment", A_NORMAL, , , A_ADD
      Forms![NewComment]![Comment] = NewData
      Response = DATA_ERRCONTINUE
End Sub

Private Sub Comment_Type_AfterUpdate()
   Me![SubComment] = ""
   Me![SubComment].Enabled = True
   Me![SubComment].Requery
End Sub
```

```
Private Sub Next_Click()
On Error GoTo Err_Next_Click
   DoCmd.GoToRecord , , A_NEXT
   If IsLoaded("Input Comments") Then
      Forms![Input Comments]![SelectComments].Form![Comment Type] = ""
      Forms![Input Comments]![SelectComments].Form![SubComment] = ""
      Forms![Input Comments]![SelectComments].Form![Comment].Requery
   End If
   If IsLoaded("Input Comments2") Then
      Forms![Input Comments2]![SelectComments].Form![Comment Type] = ""
      Forms![Input Comments2]![SelectComments].Form![SubComment] = ""
      Forms![Input Comments2]![SelectComments].Form![Comment].Requery
   End If
Exit_Next_Click:
   Exit Sub
Err_Next_Click:
   MsgBox Error$
   Resume Exit_Next_Click
End Sub

Private Sub Save_Exit(Cancel As Integer)
    Me![SubComment] = ""
End Sub

Private Sub SubCategory_Enter()
   Me![SubCategory] = ""
   Me![SubCategory].Enabled = True
   Me![SubCategory].Requery
End Sub

Private Sub SubComment_AfterUpdate()
   If IsLoaded("Input Comments") Then
      Me![Comment].Requery
   End If
   If IsLoaded("Input Comments2") Then
      Me![Comment].Requery
   End If

End Sub

Private Sub SubComment_NotInList(NewData As String, Response As Integer)
      DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
      DoCmd.OpenForm "NewSubComment", A_NORMAL, , , A_ADD
      Forms![NewSubComment]![Comment SubType] = NewData
      Response = DATA_ERRCONTINUE
End Sub
```

```
Private Sub temp_Enter()
   Me![Comment].SetFocus
   Me![Comment].Requery
End Sub

Private Sub test_Enter()
   Me![Comment].SetFocus
   Me![Comment].Requery
End Sub

Private Sub Test1_Enter()
   Me![Comment].SetFocus
   Me![Comment].Requery
End Sub

Private Sub Watch_Sta__NotInList(NewData As String, Response As Integer)
   Dim NewWS As Integer, Title As String
   DoCmd.DoMenuItem      A_FORMBAR,      A_EDIT,      A_UNDOFIELD,      ,
A_MENU_VER20
   DoCmd.OpenForm "NewWatchStation", A_NORMAL, , , A_ADD
   Forms![NewWatchStaion]![NewWS] = NewData
   Response = DATA_ERRCONTINUE
End Sub

Private Sub Watch_Station_Change()
      DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
      DoCmd.OpenForm "NewWatchStation", A_NORMAL, , , A_ADD
      Forms![NewWatchStation]![Watch Station] = NewData
End Sub

Private Sub Watch_Station_NotInList(NewData As String, Response As Integer)
      DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
      DoCmd.OpenForm "NewWatchStation", A_NORMAL, , , A_ADD
      Forms![NewWatchStation]![Watch Station] = NewData
      Response = DATA_ERRCONTINUE
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button65_Click()
On Error GoTo Err_Button65_Click
   DoCmd.Quit
Exit_Button65_Click:
   Exit Sub
```

```
Err_Button65_Click:
   MsgBox Error$
   Resume Exit_Button65_Click
End Sub

Private Sub Button66_Click()
On Error GoTo Err_Button66_Click
   DoCmd.DoMenuItem      A_FORMBAR,      A_FILE,      A_SAVERECORD,      ,
A_MENU_VER20
Exit_Button66_Click:
   Exit Sub
Err_Button66_Click:
   MsgBox Error$
   Resume Exit_Button66_Click
End Sub

Private Sub Close_Click()
On Error GoTo Err_Close_Click
   DoCmd.Close
Exit_Close_Click:
   Exit Sub
Err_Close_Click:
   MsgBox Error$
   Resume Exit_Close_Click
End Sub

Private Sub Comment_NotInList(NewData As String, Response As Integer)
      DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
      DoCmd.OpenForm "NewComment2", A_NORMAL, , , A_ADD
      Forms![NewComment2]![Comment] = NewData
      Response = DATA_ERRCONTINUE
End Sub

Private Sub Comment_Type_AfterUpdate()
   Me![SubComment] = ""
   Me![SubComment].Enabled = True
   Me![SubComment].Requery
End Sub

Private Sub Next_Click()
On Error GoTo Err_Next_Click

   DoCmd.GoToRecord , , A_NEXT
      Forms![Input Comments2]![SelectComments2].Form![Comment Type] = ""
      Forms![Input Comments2]![SelectComments2].Form![SubComment] = ""
      Forms![Input Comments2]![SelectComments2].Form![Comment].Requery
```

```
Exit_Next_Click:
   Exit Sub
Err_Next_Click:
   MsgBox Error$
   Resume Exit_Next_Click
End Sub

Private Sub Save_Exit(Cancel As Integer)
   Me![SubComment] = ""
End Sub

Private Sub SubCategory_Enter()
   Me![SubCategory] = ""
   Me![SubCategory].Enabled = True
   Me![SubCategory].Requery
End Sub

Private Sub SubComment_NotInList(NewData As String, Response As Integer)
      DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
      DoCmd.OpenForm "NewSubComment3", A_NORMAL, , , A_ADD
      Forms![NewSubComment3]![Comment SubType] = NewData
      Response = DATA_ERRCONTINUE
End Sub

Private Sub temp_Enter()
   Me![Comment].SetFocus
   Me![Comment].Requery
End Sub

Private Sub test_Enter()
   Me![Comment].SetFocus
   Me![Comment].Requery
End Sub

Private Sub Test1_Enter()
   Me![Comment].SetFocus
   Me![Comment].Requery
End Sub

Private Sub Watch_Sta__NotInList(NewData As String, Response As Integer)
   Dim NewWS As Integer, Title As String
   DoCmd.DoMenuItem      A_FORMBAR,      A_EDIT,      A_UNDOFIELD,      ,
A_MENU_VER20
   DoCmd.OpenForm "NewWatchStation", A_NORMAL, , , A_ADD
   Forms![NewWatchStaion]![NewWS] = NewData
   Response = DATA_ERRCONTINUE
```

```
End Sub

Private Sub Watch_Station_Change()
    DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
    DoCmd.OpenForm "NewWatchStation", A_NORMAL, , , A_ADD
    Forms![NewWatchStation]![Watch Station] = NewData
End Sub

Private Sub Watch_Station_NotInList(NewData As String, Response As Integer)
    DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
    DoCmd.OpenForm "NewWatchStation2", A_NORMAL, , , A_ADD
    Forms![NewWatchStation2]![Watch Station] = NewData
    Response = DATA_ERRCONTINUE
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub DelRec_Click()
On Error GoTo Err_DelRec_Click
  DoCmd.DoMenuItem A_FORMBAR, A_EDITMENU, A_SELECTRECORD_V2, ,
A_MENU_VER20
  DoCmd.DoMenuItem A_FORMBAR, A_EDITMENU, A_DELETE_V2, ,
A_MENU_VER20
Exit_DelRec_Click:
  Exit Sub
Err_DelRec_Click:
  MsgBox Error$
  Resume Exit_DelRec_Click
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button13_Click()
On Error GoTo Err_Button13_Click
  DoCmd.DoMenuItem A_FORMBAR, A_EDITMENU, A_SELECTRECORD_V2, ,
A_MENU_VER20
  DoCmd.DoMenuItem A_FORMBAR, A_EDITMENU, A_DELETE_V2, ,
A_MENU_VER20
Exit_Button13_Click:
  Exit Sub
Err_Button13_Click:
  MsgBox Error$
  Resume Exit_Button13_Click
End Sub
```

```
Private Sub Button14_Click()
On Error GoTo Err_Button14_Click
  DoCmd.Close
Exit_Button14_Click:
  Exit Sub
Err_Button14_Click:
  MsgBox Error$
  Resume Exit_Button14_Click
End Sub

Private Sub Comment_AfterUpdate()
  Response = MsgBox("Changes to the Drill Comment will affect all Drills it is
associated with.", 48, "WARNING")
End Sub
Private Sub Comment_Type_AfterUpdate()
    Me![SubComment] = " "
    DoCmd.Requery
    Me![SubComment].Requery
End Sub

Private Sub ORSE_Commen_Deficenc_AfterUpdate()
  Response = MsgBox("Changes to the Drill Comment will affect all Drills it is
associated with.", 48, "WARNING")
End Sub

Private Sub SubComment_AfterUpdate()
  DoCmd.Requery
End Sub

Private Sub SubComment_NotInList(NewData As String, Response As Integer)
    DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
    DoCmd.OpenForm "NewSubComment2", A_NORMAL, , , A_ADD
    Forms![NewSubComment2]![Comment SubType] = NewData
    Response = DATA_ERRCONTINUE
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button13_Click()
On Error GoTo Err_Button13_Click
  DoCmd.DoMenuItem A_FORMBAR, A_EDITMENU, A_SELECTRECORD_V2, ,
A_MENU_VER20
  DoCmd.DoMenuItem    A_FORMBAR,    A_EDITMENU,    A_DELETE_V2,    ,
A_MENU_VER20
```

```
Exit_Button13_Click:
   Exit Sub
Err_Button13_Click:
   MsgBox Error$
   Resume Exit_Button13_Click
End Sub

Private Sub Button14_Click()
On Error GoTo Err_Button14_Click
   DoCmd.Close
Exit_Button14_Click:
   Exit Sub
Err_Button14_Click:
   MsgBox Error$
   Resume Exit_Button14_Click
End Sub

Private Sub Comment_SubType_AfterUpdate()
   Response = MsgBox("Changes to the Comment Sub-Category will affect all Drill
Comments it is associated with.", 48, "WARNING")
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button27_Click()
On Error GoTo Err_Button27_Click
   DoCmd.GoToRecord , , A_NEXT
Exit_Button27_Click:
   Exit Sub
Err_Button27_Click:
   MsgBox Error$
   Resume Exit_Button27_Click
End Sub

Private Sub Button28_Click()
On Error GoTo Err_Button28_Click
   DoCmd.GoToRecord , , A_LAST
Exit_Button28_Click:
   Exit Sub
Err_Button28_Click:
   MsgBox Error$
   Resume Exit_Button28_Click
End Sub

Private Sub Button29_Click()
```

```
On Error GoTo Err_Button29_Click
   DoCmd.GoToRecord , , A_FIRST
Exit_Button29_Click:
   Exit Sub
Err_Button29_Click:
   MsgBox Error$
   Resume Exit_Button29_Click
End Sub

Private Sub Button30_Click()
On Error GoTo Err_Button30_Click
   DoCmd.DoMenuItem A_FORMBAR, A_EDITMENU, A_SELECTRECORD_V2, ,
A_MENU_VER20
   DoCmd.DoMenuItem A_FORMBAR, A_EDITMENU, A_DELETE_V2, ,
A_MENU_VER20
   DoCmd.Requery "CrossTabSubForm"
Exit_Button30_Click:
   Exit Sub
Err_Button30_Click:
   MsgBox Error$
   Resume Exit_Button30_Click
End Sub

Private Sub Button31_Click()
On Error GoTo Err_Button31_Click
   DoCmd.GoToRecord , , A_NEWREC
Exit_Button31_Click:
   Exit Sub
Err_Button31_Click:
   MsgBox Error$
   Resume Exit_Button31_Click
End Sub

Private Sub Close_Click()
On Error GoTo Err_Close_Click
   DoCmd.Close
Exit_Close_Click:
   Exit Sub
Err_Close_Click:
   MsgBox Error$
   Resume Exit_Close_Click
End Sub

Private Sub DelRecord_Click()
On Error GoTo Err_DelRecord_Click
```

```
    DoCmd.DoMenuItem A_FORMBAR, A_EDITMENU, A_SELECTRECORD_V2, ,
A_MENU_VER20
    DoCmd.DoMenuItem A_FORMBAR, A_EDITMENU, A_DELETE_V2, ,
A_MENU_VER20
Exit_DelRecord_Click:
    Exit Sub
Err_DelRecord_Click:
    MsgBox Error$
    Resume Exit_DelRecord_Click
End Sub

Private Sub Evolution_NotInList(NewData As String, Response As Integer)
        DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
        DoCmd.OpenForm "UpdateEvolsList", A_NORMAL, , , A_ADD
        Forms![UpdateEvolsList]![Evolution] = NewData
        Response = DATA_ERRCONTINUE
End Sub

Private Sub Field20_NotInList(NewData As String, Response As Integer)
        DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
        DoCmd.OpenForm "UpdateEvolsList", A_NORMAL, , , A_ADD
        Forms![UpdateEvolsList]![Evolution] = NewData
        Response = DATA_ERRCONTINUE
End Sub

Private Sub Form_AfterUpdate()
    DoCmd.Requery "CrossTabSubForm"
End Sub

Private Sub SaveRecord_Click()
On Error GoTo Err_SaveRecord_Click
    DoCmd.DoMenuItem A_FORMBAR, A_FILE, A_SAVERECORD, ,
A_MENU_VER20
Exit_SaveRecord_Click:
    Exit Sub
Err_SaveRecord_Click:
    MsgBox Error$
    Resume Exit_SaveRecord_Click
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button13_Click()
On Error GoTo Err_Button13_Click
```

```
    DoCmd.DoMenuItem A_FORMBAR, A_EDITMENU, A_SELECTRECORD_V2, ,
A_MENU_VER20
    DoCmd.DoMenuItem    A_FORMBAR,    A_EDITMENU,    A_DELETE_V2,    ,
A_MENU_VER20

Exit_Button13_Click:
    Exit Sub
Err_Button13_Click:
    MsgBox Error$
    Resume Exit_Button13_Click
End Sub

Private Sub Button14_Click()
On Error GoTo Err_Button14_Click
    DoCmd.Close
Exit_Button14_Click
    Exit Sub
Err_Button14_Click:
    MsgBox Error$
    Resume Exit_Button14_Click
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button13_Click()
On Error GoTo Err_Button13_Click
    DoCmd.DoMenuItem A_FORMBAR, A_EDITMENU, A_SELECTRECORD_V2, ,
A_MENU_VER20
    DoCmd.DoMenuItem    A_FORMBAR,    A_EDITMENU,    A_DELETE_V2,    ,
A_MENU_VER20
Exit_Button13_Click:
    Exit Sub
Err_Button13_Click:
    MsgBox Error$
    Resume Exit_Button13_Click
End Sub

Private Sub Button14_Click()
On Error GoTo Err_Button14_Click
    DoCmd.Close
Exit_Button14_Click:
    Exit Sub
Err_Button14_Click:
    MsgBox Error$
    Resume Exit_Button14_Click
```

```
End Sub

Private Sub Form_Unload(Cancel As Integer)
   Dim MyControl As Control
   If IsLoaded("Drills/Evol Cross Ref") Then
      Set MyControl = Forms![Drills/Evol Cross Ref]![Drill/Evol ID]
      DoCmd.SelectObject A_FORM, "Drills/Evol Cross Ref"
      MyControl.Requery
      MyControl = Forms![Drills/Evol Cross Ref]![Drill/Evol ID]
      Forms![Drills/Evol Cross Ref]![Drill/Evol ID] = Me![Drill/Evol ID]
  End If
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button13_Click()
On Error GoTo Err_Button13_Click
   DoCmd.DoMenuItem A_FORMBAR, A_EDITMENU, A_SELECTRECORD_V2, ,
A_MENU_VER20
   DoCmd.DoMenuItem A_FORMBAR, A_EDITMENU, A_DELETE_V2, ,
A_MENU_VER20
Exit_Button13_Click:
   Exit Sub
Err_Button13_Click:
   MsgBox Error$
   Resume Exit_Button13_Click
End Sub

Private Sub Button14_Click()
On Error GoTo Err_Button14_Click
   DoCmd.Close
Exit_Button14_Click:
   Exit Sub
Err_Button14_Click:
   MsgBox Error$
   Resume Exit_Button14_Click
End Sub

Private Sub Form_Unload(Cancel As Integer)
   Dim MyControl As Control
   If IsLoaded("EvolutionsPerformed") Then
      Set MyControl = Forms![EvolutionsPerformed]![Evolution]
      DoCmd.SelectObject A_FORM, "EvolutionsPerformed"
      MyControl.Requery
      MyControl = Forms![EvolutionsPerformed]![Evolution]
```

```
    Forms![EvolutionsPerformed]![Evolution] = Me![Evolution ID]
  End If
  If IsLoaded("UpdateCrossRefTable") Then
    Set MyControl = Forms![UpdateCrossRefTable]![Evolution]
    DoCmd SelectObject A_FORM, "UpdateCrossRefTable"
    MyControl.Requery
    MyControl = Forms![UpdateCrossRefTable]![Evolution]
    Forms![UpdateCrossRefTable]![Evolution] = Me![Evolution ID]
  End If
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button15_Click()
On Error GoTo Err_Button15_Click
  DoCmd.Close
Exit_Button15_Click:
  Exit Sub
Err_Button15_Click:
  MsgBox Error$
  Resume Exit_Button15_Click
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button13_Click()
On Error GoTo Err_Button13_Click
  DoCmd.Close
Exit_Button13_Click:
  Exit Sub
Err_Button13_Click:
  MsgBox Error$
  Resume Exit_Button13_Click
End Sub

Private Sub Button14_Click()
On Error GoTo Err_Button14_Click
  DoCmd.DoMenuItem A_FORMBAR, A_EDITMENU, A_SELECTRECORD_V2, ,
A_MENU_VER20
  DoCmd.DoMenuItem A_FORMBAR, A_EDITMENU, A_DELETE_V2, ,
A_MENU_VER20
Exit_Button14_Click:
  Exit Sub
Err_Button14_Click:
```

```
    MsgBox Error$
    Resume Exit_Button14_Click
End Sub

Private Sub Watch_Station_AfterUpdate()
    Response = MsgBox("Changes to the Watch Station will affect all Drill Comments it is
associated with.", 48, "WARNING")
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Private Sub Button65_Click()
On Error GoTo Err_Button65_Click
    DoCmd.Quit
Exit_Button65_Click:
    Exit Sub
Err_Button65_Click:
    MsgBox Error$
    Resume Exit_Button65_Click
End Sub

Private Sub Button66_Click()
On Error GoTo Err_Button66_Click
    DoCmd.DoMenuItem    A_FORMBAR,    A_FILE,    A_SAVERECORD,    ,
A_MENU_VER20
Exit_Button66_Click:
    Exit Sub
Err_Button66_Click:
    MsgBox Error$
    Resume Exit_Button66_Click
End Sub

Private Sub Close_Click()
On Error GoTo Err_Close_Click
    DoCmd.Close
Exit_Close_Click:
    Exit Sub
Err_Close_Click:
    MsgBox Error$
    Resume Exit_Close_Click
End Sub

Private Sub Comment_NotInList(NewData As String, Response As Integer)
    DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
    DoCmd.OpenForm "NewComment", A_NORMAL, , , A_ADD
```

```
    Forms![NewComment]![Comment] = NewData
    Response = DATA_ERRCONTINUE
End Sub

Private Sub Comment_Type_AfterUpdate()
  Me![SubComment] = ""
  Me![SubComment].Enabled = True
  Me![SubComment].Requery
End Sub

Private Sub Comment_Type_Enter()
  Me![SubComment] = ""
  Me![SubComment].Enabled = True
  Me![SubComment].Requery
End Sub

Private Sub DelRec_Click()
On Error GoTo Err_DelRec_Click
  Dim MyControl As Control
  Set MyControl = Me![SelectCommentsView]
  MyControl.SetFocus
  DoCmd.DoMenuItem A_FORMBAR, A_EDITMENU, A_SELECTRECORD_V2, ,
A_MENU_VER20
  DoCmd.DoMenuItem A_FORMBAR, A_EDITMENU, A_DELETE_V2, ,
A_MENU_VER20
Exit_DelRec_Click:
  Exit Sub

Err_DelRec_Click:
  MsgBox Error$
  Resume Exit_DelRec_Click
End Sub

Private Sub EnterComments_Click()
On Error GoTo Err_EnterComments_Click
  Dim DocName As String
  Dim LinkCriteria As String
  DocName = "SelectComments"
  LinkCriteria = "[Drill ID] = Forms![Input Comments]![Drill ID]"
  DoCmd.OpenForm DocName, , , LinkCriteria
Exit_EnterComments_Click:
  Exit Sub
Err_EnterComments_Click:
  MsgBox Error$
  Resume Exit_EnterComments_Click
End Sub
```

```
Private Sub Form_Current()
   Me![Anomalies].Requery
End Sub

Private Sub Form_Open(Cancel As Integer)
   If IsLoaded("Select Drill") Then
      Me![Next].Enabled = False
      Me![Next].Visible = False
      Me![DelRec].Enabled = True
      Me![DelRec].Visible = True
    End If
End Sub

Private Sub Next_Click()
On Error GoTo Err_Next_Click
   DoCmd.GoToRecord , , A_NEXT
Exit_Next_Click:
   Exit Sub
Err_Next_Click:
   MsgBox Error$
   Resume Exit_Next_Click
End Sub

Private Sub Save_Exit(Cancel As Integer)
   Me![SubComment] = ""
End Sub

Private Sub SubCategory_Enter()
   Me![SubCategory] = ""
   Me![SubCategory].Enabled = True
   Me![SubCategory].Requery
End Sub

Private Sub SubComment_AfterUpdate()
  Me![Comment].Requery
End Sub

Private Sub SubComment_NotInList(NewData As String, Response As Integer)
      M_COMMENT = Me![Comment Type]
      DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
      DoCmd.OpenForm "NewSubComment", A_NORMAL, , , A_ADD
      Forms![NewSubComment]![Comment SubType] = NewData
      Response = DATA_ERRCONTINUE
End Sub
```

```
Private Sub temp_Enter()
   Me![Comment].SetFocus
   Me![Comment].Requery
End Sub

Private Sub test_Enter()
   Me![Comment].SetFocus
   Me![Comment].Requery
End Sub

Private Sub Test1_Enter()
   Me![Comment].SetFocus
   Me![Comment].Requery
End Sub

Private Sub Watch_Sta__NotInList(NewData As String, Response As Integer)
   Dim NewWS As Integer, Title As String
   DoCmd.DoMenuItem      A_FORMBAR,      A_EDIT,      A_UNDOFIELD,      ,
A_MENU_VER20
   DoCmd.OpenForm "NewWatchStation", A_NORMAL, , , A_ADD
   Forms![NewWatchStaion]![NewWS] = NewData
   Response = DATA_ERRCONTINUE
End Sub

Private Sub Watch_Station_Change()
      DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
      DoCmd.OpenForm "NewWatchStation", A_NORMAL, , , A_ADD
      Forms![NewWatchStation]![Watch Station] = NewData
End Sub

Private Sub Watch_Station_NotInList(NewData As String, Response As Integer)
      DoCmd.DoMenuItem A_FORMBAR, A_EDIT, A_UNDOFIELD
      DoCmd.OpenForm "NewWatchStation", A_NORMAL, , , A_ADD
      Forms![NewWatchStation]![Watch Station] = NewData
      Response = DATA_ERRCONTINUE
End Sub
```

```
Option Compare Database   'Use database order for string comparisons

Option Explicit
' Constant for maximum number of columns Employee Sales query would
' create plus 1 for a Totals column.
Const TOTCOLS = 11
' Variables for database object and recordset.
Dim RptDB As Database
```

```
Dim RptRS As Recordset
' Variables for number of columns and row and report totals.
Dim IColCnt As Integer
Dim RgColTot(1 To TOTCOLS) As Long
Dim RptTotal As Long

Private Sub Report_Close()
   'Close recordset.
   RptRS.Close
End Sub

Private Sub Report_Open(Cancel As Integer)
   ' Create underlying recordset for report using criteria entered in
   ' ReportParameters Form.
   Dim i As Integer
   Dim MyQuery As QueryDef
   ' Don't open report if ReportParameters form isn't loaded.
   If Not (IsLoaded("ReportParameters Dialog Box")) Then
      Cancel = True
      MsgBox "To preview or print this report, you must open the ReportParameters in
Form view.", 48, "Must Open Dialog Box"
      Exit Sub
   End If
   ' Set database variable to current database.
   Set RptDB = DBEngine.Workspaces(0).Databases(0)
   ' Open QueryDef.
   Set MyQuery = RptDB.QueryDefs("Q_EngDrillLog_R")
   ' Set parameters for query based on values entered in ReportParameters form.
   MyQuery.Parameters("Forms![ReportParameters Dialog Box]![StartDate]") =
Forms![ReportParameters Dialog Box]![StartDate]
   MyQuery.Parameters("Forms![ReportParameters Dialog Box]![StopDate]") =
Forms![ReportParameters Dialog Box]![StopDate]
   MyQuery.Parameters("Forms![ReportParameters Dialog Box]![Fwd/Aft]") =
Forms![ReportParameters Dialog Box]![Fwd/Aft]
   ' Open Recordset.
   Set RptRS = MyQuery.OpenRecordset()
   ' If no records match criteria, display message,
   ' close recordset, and cancel Open event.
   If RptRS.RecordCount = 0 Then
      MsgBox "No records match the criteria you entered.", 48, "No Records Found"
      RptRS.Close
      Cancel = True
      Exit Sub
   End If
   ' Set a variable to hold number of columns in crosstab query.
   IColCnt = RptRS.Fields.Count
```

```
End Sub
```

```
Option Compare Database   'Use database order for string comparisons
Option Explicit

' Constant for maximum number of columns Evolutions Performed query would
' create plus 1 for a Totals column. *** Totals Not Needed
Const TOTCOLS = 4
' Variables for database object and recordset.
Dim RptDB As Database
Dim RptRS As Recordset
' Variables for number of columns and row and report totals.
Dim IColCnt As Integer
Dim RgColTot(1 To TOTCOLS) As Long
Dim RptTotal As Long

Private Sub Detail1_Format(Cancel As Integer, FormatCount As Integer)
   ' Place values in text boxes and hide unused text boxes.
   Dim i As Integer
   ' Verify that not at end of recordset.
   If Not RptRS.EOF Then
      ' If FormatCount is 1, place values from recordsSet into text boxes
      ' in detail section.
      If Me.FormatCount = 1 Then
         For i = 1 To IColCnt
            ' Convert null values to 0.
            Me("Col" + Format$(i)) = "  " & xtabCnulls(RptRS(i - 1))  '*
         Next i
         ' Hide unused text boxes in detail section.
         For i = IColCnt + 1 To TOTCOLS
            Me("Col" + Format$(i)).Visible = False
         Next i
         ' Move to next record in recordset.
         RptRS.MoveNext
      End If
   End If
End Sub

Private Sub Detail1_Retreat()
   ' Always back up to previous record when detail section retreats.
   RptRS.MovePrevious
End Sub
Private Sub InitVars()
   Dim i As Integer
   ' Initialize RptTotal variable.
```

```
    RptTotal = 0
    ' Initialize array that stores column totals.
    For i = 1 To TOTCOLS
        RgColTot(i) = 0
    Next i
End Sub

Private Sub PageHeader0_Format(Cancel As Integer, FormatCount As Integer)
    Dim i As Integer
    ' Put column headings into text boxes in page header.
    For i = 1 To IColCnt
        If i = 1 Then
            Me("Head" + Format$(i)) = RptRS(i - 1).Name
        Else
            Me("Head" + Format$(i)) = "SECTION " & RptRS(i - 1).Name
        End If
    Next i
    For i = (IColCnt + 1) To TOTCOLS
        Me("Head" + Format$(i)).Visible = False
    Next i
End Sub

Private Sub Report_Close()
    ' Close recordset.
    RptRS.Close
End Sub

Private Sub Report_Open(Cancel As Integer)
    ' Create underlying recordset for report using criteria entered in
    ' Evolutions Performed Dialog Box form.
    '
    Dim i As Integer
    Dim MyQuery As QueryDef
    ' Don't open report if Evolutions Performed Dialog Box form isn't loaded.
    If Not (IsLoaded("Evolutions Performed Dialog Box")) Then
        Cancel = True
        MsgBox "To preview or print this report, you must open the Evolutions Performed
Dialog Box in Form view.", 48, "Must Open Dialog Box"
        Exit Sub
    End If
    ' Set database variable to current database.
    Set RptDB = DBEngine.Workspaces(0).Databases(0)
    ' Open QueryDef.
    Set MyQuery = RptDB.QueryDefs("Q_Evolutions_CrossTab1")
    ' Set parameters for query based on values entered in Evolutions Performed Dialog
Box form.
```

```
    MyQuery.Parameters("Forms![Evolutions Performed Dialog Box]![StartDate]") =
Forms![Evolutions Performed Dialog Box]![StartDate]
    MyQuery.Parameters("Forms![Evolutions Performed Dialog Box]![StopDate]") =
Forms![Evolutions Performed Dialog Box]![StopDate]
    ' Open Recordset.
    Set RptRS = MyQuery.OpenRecordset()
    ' If no records match criteria, display message,
    ' close recordset, and cancel Open event.
    If RptRS.RecordCount = 0 Then
        MsgBox "No records match the criteria you entered.", 48, "No Records Found"
        RptRS.Close
        Cancel = True
        Exit Sub
    End If
    ' Set a variable to hold number of columns in crosstab query.
    IColCnt = RptRS.Fields.Count
End Sub

Private Sub ReportHeader3_Format(Cancel As Integer, FormatCount As Integer)
    ' Move to first record in recordset at beginning of report
    ' or when report is restarted. (A report is restarted when
    ' you print a report from Print Preview window, or when you return
    ' to a previous page while previewing.)
    RptRS.MoveFirst
    'Initialize variables.
    InitVars
End Sub

Private Function xtabCnulls(MyVal As Variant)
    ' Test if a value is null.
    If IsNull(MyVal) Then
        ' If MyVal is null, set MyVal to 0.
        xtabCnulls = 0
    Else
        ' Otherwise, return MyVal.
        xtabCnulls = MyVal
    End If
End Function
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Balanced Scorecard (BSC) Home Page.
[www.pr.doe.gov/bsc001.htm]. August 2000.

Baltzis, S., *Web Server Configuration for an Academic Intranet*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 2000.

Cebrowski, A.K., Vice Admiral, USN, Before the House Armed Services Committee: Hearing on Network-Centric Warfare and Information Superiority, February 23, 1999.

Clemins, Archie, Admiral, USN, Before the House Armed Services Committee: Hearing on Information Superiority, February 23, 1999.

CNET Pensacola, Fl Naval Message, Subject: Information Technology (IT) Short and Long Range Architectures, 300944Z Mar 97.

Coleman-Martin, "State of Delaware: N-tier Computing Architecture", 1998.

Compaq, "Services/Systems Integration/Web-enabled Applications."
[www.compaq.com/services/integration/si_webapps.html]. April 4, 2001.

Data Constructs, "Using Stored Documents, Technical Note – GenTN002, Rev A."
[www.databaseconstructs.com/GenTN002.html]. Jan 2001.

Director for Strategic Plans and Policy (J5), Joint Chiefs of Staff, *Joint Vision 2020*, US Government Printing Office, Washington D.C., June 2000.

Edwards, Jeri, *3-Tier Client/Server At Work*, Revised Edition, John Wiley & Sons, Inc., 1999.

Edwards, John, "Business Tools Get "Webified"," *CIO Magazine*, 1 August, 1999.

Fabris, Peter, "You Think Tomaytoes, I Think Tomahtoes," *CIO Magazine*, 1 April, 1999.

Fichera, R., "Firm Foundations", *CIO Magazine*, 15 November 1999.

Finegan, J., "Joining Forces", *CIO Magazine*, 1 December 1997.

Fleet Training Toolbox, *Training Links*.
[http://www.fleettraining.navy.mil/]. 2001.

Friedrichsen, L., "Data-Driven Web Sites with Microsoft Access 2000: Tools for E-Commerce," Thompson Learning, 2001.

Kalin, Sari, "Blue on Blue," *CIO Magazine*, 1 July, 1998.

Gould, Steven, "Develop n-tier applications using J2EE", *Java World*, December 2000.

Hapgood, F., "Tools for Teamwork", *CIO Magazine*, 1 November 1999.

Kalin, S., "Overdrive", *CIO Magazine*, 1 July 1999.

Ladd, E. and J. O'Donnell, "Using XHTML, XML, and Java 2," Platinum ed., Que Corporation, 2001.

Matheson, P.G., *An Operational Intranet for Fighter Composite Squadron Thirteen (VFC-13)*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1999.

McCreary, L. and T. Horgan, "On the Inside Looking Out", *CIO Magazine,* 1 July 1999.

Mercer, D., "ASP 3.0 A Beginner's Guide," The McGraw-Hill Companies, 2001.

Paige, Emmett Jr., Assistant Secretary of Defense (Command, Control, Communications and Intelligence), *Achieving the Integrated Systems Concept*, Defense Issues, Vol 11, Number 51. 4 June 1996.

Patrou, B.M., *A Naval Reserve Database Application and Future Network Solutions,* Master's Thesis, Naval Postgraduate School, Monterey, California, June 1998.

Pratt, P.J. and J.J. Adamski, "Database Systems Management and Design," 2d ed., Boyd & Fraser Publishing Company, 1991.

Procurement Executives' Association, "Guide to a Balance Scorecard Performance Management Methodology."
[www.statebuy.gov/bsc.htm]. October 1998.

Reinholt, Kurt B., *Automating Aviation Training Records*, Master's Thesis, Naval Post Graduate School, Monterey, California, Sept 2000.

Santosus, M., "Wire Education", *CIO Magazine*, 1 October 1999.

SAS Institute Inc., "Turn Measures into Strategies Through Focus and Alignment". [www.sas.com/solutions/bsc/]. 2001.

Shand, D., "Learning Unbound", *CIO Magazine*, 15 March 1999.

Sizemore, S., *Requirements Analysis and Infrastructure Assessment Methodologies for Intranet Development,* Master's Thesis, Naval Postgraduate School, Monterey, California, September 2000.

SOBT Learning Management System, Submarine On Board Training. [sobtweb.csg2.navy.mil/]. May 2001.

Strebe, M. and C. Perkins, "MCSE: Internet Information Server 4 Study Guide," 3d ed., Sybex Inc., 2000.

Stuart, A., "Continuing Education", *CIO Magazine*, 1 September 1999.

Sun Microsystems, Inc., Software White Papers, "Scaling the N-Tier Architecture: Solaris[tm] Infrastructure Products and Architecture." [www.sun.com/software/whitepapers/wp-ntier/]. 2001.

Terronez, T., *Implementation of a Submarine Ship-Wide, Common Network Architecture,* Master's Thesis, Naval Postgraduate School, Monterey, California, September 2000.

The Balance Scorecard Institute. [www.balancedscorecard.org/]. April 2001.

Tomsen, Mai-lan, "Build Reliable and Scalable N-tier Applications that Run on Both Windows NT and Unix," *Microsoft Systems Journal*, December 1998.

Young, Debby, "An Audit Tail," *CIO Magazine*, May 01, 2000.

U.S. Marine Corps Training and Education Division, Standards Branch, MCCDC. [http://www.tecom.usmc.mil/stds/Default.htm]. 2001

Chief of Naval Operations (CNO) Office of Training Technology (N75) [http://www.ott.navy.mil]. 2001.

Web-Based Training Information Center. [www.filename.com/wbt/index.html]. 2001.

Whitecar, M., *The Use of Real Time Intelligent Technologies to Collect, Report and Forcast Medical Readiness,* Master's Thesis, Naval Postgraduate School, Monterey, California, June 2001.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

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

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Road
   Monterey, CA 93943-5101

3. Professor William J. Haga 1
   Naval Post Graduate School
   Monterey, CA 93943

4. Professor Douglas E. Brinkley 1
   Naval Post Graduate School
   Monterey, CA 93943

5. CDR Joe Watkins 1
   COMSUBPAC N76, Knowledge Management Officer
   1430 Morton Street
   Pearl Harbor, HI 96860

6. Maj. David J. Murphy 1
   4441 Mayfair Ct.
   Carlsbad, CA 92008

7. LT. Kevin S. Anderson 1
   31 Talbot Ct.
   Saint Marys, GA 31558

8. Director, Training and Education 1
   MCCDC, Code C46
   1019 Elliot Road
   Quantico, VA 22134-5027

9. Director, Marine Corps Research Center 2
MCCDC, Code C40RC
2040 Broadway Street
Quantico, VA 22134-5107

10. Marine Corps Representative 1
Naval Post Graduate School
Code 037, Bldg. 330, Ingersoll Hall, Room 116
555 Dyer Road
Monterey, CA 93943

11. Marine Corps Tactical Systems Support Activity 1
Technical Advisory Branch
Box 555171
Camp Pendleton, CA 92055-5080