2001

Distributed Maintenance Error Information, Investigation and Intervention

Zolla, George; Boex, Tony; Flanders, Pat; Nelson, Doug; Tufts, Scott; Schmidt, John K.

Society of Automotive Engineers, Inc.
http://hdl.handle.net/10945/40103

2001-01-2950

Distributed Maintenance Error Information, Investigation and Intervention

George Zolla, Tony Boex, Pat Flanders, Doug Nelson, Scott Tufts
Naval Postgraduate School

John K. Schmidt
Naval Safety Center

ABSTRACT

This paper describes a safety information management system designed to capture maintenance factors that contribute to aircraft mishaps. The Human Factors Analysis and Classification System-Maintenance Extension taxonomy (HFACS-ME), an effective framework for classifying and analyzing the presence of maintenance errors that lead to mishaps, incidents, and personal injuries, is the theoretical foundation. An existing desktop mishap application is updated, a prototype web-based model is developed and an Asynchronous Distributed Learning (ADL) module is conceptualized. These tools facilitate data collection, organization, query, analysis, and the reporting of maintenance errors that contribute to aviation mishaps. Together they represent a complete, robust system for analyzing aircraft maintenance mishap related factors anywhere at anytime.

INTRODUCTION

From 1950 to 2000 Naval Aviation had great success in substantially reducing its Flight Mishap (FM) rate (Figure 1). However, the proportion of mishaps attributed to human error remained at a relative constant rate of about 80 percent (Nutwell & Sherman, 1997). In 1996, a Navy F-14 Tomcat crashed shortly after taking off from Nashville, Tennessee killing both aircrew and three civilians on the ground. Since the cause of this mishap was exclusively human error, Department of the Navy (DON) leaders established a Human Factors Quality Management Board (HFQMB) to significantly reduce mishaps caused by human error by identifying systemic improvements in the processes and systems that guard against error and enhance human performance. The HFQMB's goals were to reduce the Naval Aviation Class A (most serious) Flight Mishap (FM) rate by 50 percent by FY 2000 and 75 percent by FY 2006 (HFQMB, 1997).

The HFQMB's initial thrust was to conduct an extensive mishap data analyses focused on human factors. The Naval Safety Center (NSC) developed the Human Factors Analysis and Classification System (HFACS) taxonomy to capture aircrew errors in Naval Aviation mishaps. This taxonomy identifies areas for potential intervention by describing factors that may be precursors to accidents. The resulting HFACS taxonomy focused solely on aircrew errors and identified both active failures and latent conditions within four categories: 1) unsafe acts, 2) pre-conditions for unsafe acts, 3) unsafe supervision, and 4) organizational influences (DON, 2001). NSC has adopted HFACS for analyzing human error in Naval Aviation mishaps and targeting appropriate prevention (DON, 2001).

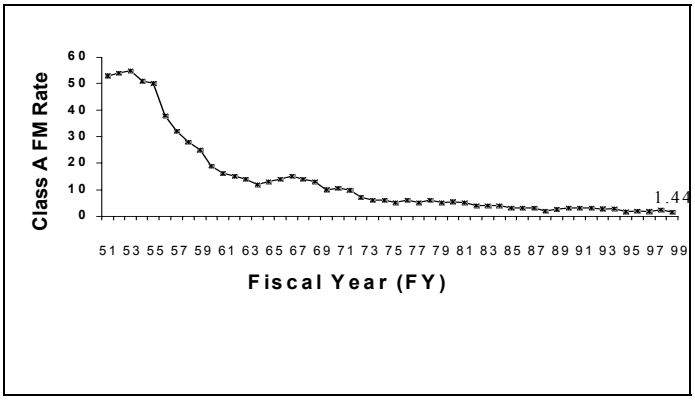

Figure 1: Naval Aviation Class A Flight Mishap Rates for FY 1950-1999 (NCS, 2001)

After the establishment of HFACS in FY 1999, Naval Aviation enjoyed its lowest mishap rate ever. Even with the reduction in the mishap rate, the HFQMB's goal to reduce human error related mishaps by 50 percent by FY 2000 was not achieved (NSC, 2001). During this period, a study discovered that HFACS could be extended to cover maintenance errors (Schmidt, Schmorrow, & Hardee, 1997). Hence, a Maintenance Extension (ME) for the HFACS taxonomy was adapted to classify causal factors that contributed to maintenance mishaps (Schmidt, 1996).

The HFACS-ME classification system (Appendix A) contains four first order, ten second order and thirty-four third order human error categories. The first order categories consist of: 1) Management Conditions, 2) Working Conditions, 3) Maintainer Conditions, and 4) Maintainer Acts. A mathematical analysis of 470 Naval Aviation Mishaps by Schmorrow (1998) determined the HFACS-ME taxonomy was an effective classification system for determining trends in aviation mishaps. Building on Schmorrow's research, Fry (2000) developed the Maintenance Error Information Management System (MEIMS), a desktop database application, for the analysis of maintenance related mishaps. MEIMS effectively used more mishap data leading to a refinement of HFACS-ME, making it more comprehensive and accessible.

Fry's rudimentary MIEMS tool was further refined by Wood (2000) and developed into a working prototype for U.S. Navy Fleet testing and evaluation. A usability study of the prototype MEIMS tool determined that it could be an effective system, not only in determining trends but providing information for mishap prevention efforts. Wood's study identified the need for MEIMS to incorporate improved HFACS-ME definitions, improve the user interface, simplify data entry procedures, and include a narrative of common mishap scenarios. This paper discusses how these improvements are designed into the new system as well as how a web-based version for intranet and Internet use was developed. Future enhancements include an Asynchronous Distance Learning (ADL) module to teach users to effectively manipulate the web-based version for data-mining and trend analysis.

A grant from NASA and support from civilian air carriers expanded the original scope of the project to include the design and implementation of a civilian variant of MEIMS. NASA directed that an upgraded desktop database model of MEIMS be created that would be compatible with Access 2000.

The next section of this paper will discuss the enhanced database model. Then the user interface will be described followed by the web-based version implementation. Next the ADL module is discussed, followed by the conclusions.

DATABASE MODEL

REQUIRMENTS - Ideally, the updated database model should:

1) Capitalize on object-oriented technology to provide better performance, greatly increased scalability, and broader opportunity for code re-use.

2) Provide multi-user access to the database in a true client-server environment while maintaining the ability to function as a small footprint, easy to use, standalone desktop application.

3) Provide a full-featured interface for web-based access.

4) Isolate itself, as best as possible, from compatibility changes in new versions of software. This includes development language, database engine, and operating system.

DISCUSSION - The current version of MEIMS was originally implemented using *Microsoft Access 97* and then migrated to *Access 2000*. *Microsoft Access* has built-in functionality to create desktop applications with forms, reports, and embedded support for *Visual Basic for Applications* (VBA). The data in an *Access* database can be manipulated using several different programming languages, Active Server Pages (ASPs) via the web, and via third party add-in tools. A key feature of *Access* over other databases and development tools is its ease of use - it is a very effective rapid application development (RAD) platform. *Access* can be (depending upon the implementation options) much simpler to use for building standalone, small-scale database solutions than many of its competitors. Our current development effort focuses on meticulous software engineering to adapt the current *Access* "structured" systems to *Microsoft*-based object oriented architectures ensuring future scalability and increased potential for code-reuse.

A new feature of *Access 2000* that made it appealing for the HFACS-ME project is the ability to use more than one type of database engine. A database engine is the part of a database management system (DBMS) that actually stores and retrieves data. *Access 2000* provides support for both the *Microsoft JET* database engine and *the Microsoft SQL Server* engine. This is a key distinction. *Access* originally had only one choice as a database engine: *JET*. The main problem with *JET* is that it is not a client/server capable engine. It really performs as a file server. This means that anytime a client wants to request something from a *JET* database, a lot more has to be done on the client-side. The result is a lot of network traffic and unacceptable response times for more than only a handful of simultaneous users. With the release of *Office 2000*, however, *Microsoft* provided a royalty free version of the *SQL Server* engine for use with *Access*. Most significant, (other than it being free) is that it is capable of running on a desktop computer. This change allowed an *Access* solution that had the ability to operate as a stand-alone application using the same engine as the full version of *SQL Server.* Upgrading from a desktop application to a server-based application was no longer an issue because the engine is the same.

One confusing aspect of the standalone engine is the difference in naming conventions between various versions of *SQL Server*. The *SQL Server 6.5 and 7.0* compliant version is called the *Microsoft Data Engine* (MSDE), while the *SQL Server 2000* compatible variant is called *Microsoft SQL Server Desktop Edition.* Both versions provide *SQL Server* database functionality, but there are significant differences. For the remainder of this paper, however, in order to provide greater emphasis on the distinction between full *SQL Server* and the two Desktop editions, we will refer to both the *SQL 6.5/7.0* and *2000* versions of the desktop engine as *MSDE* unless otherwise stated.

In our research we found that it is a very common requirement to migrate a *JET* based database to a more robust database engine - namely *Microsoft SQL Server 7.0* or *SQL Server*

2000. We also found that automated migration tools designed to port *JET* databases over to *SQL server* are useful only for very simple databases. We experimented with using the *Microsoft Access* "Upsizing wizard" on MEIMS with very poor results. Structured Query Language written in *Access* using *JET* did not transfer correctly. Functions written in VBA did not transfer correctly. In addition, the data types used by *JET* are different from those in *SQL Server* and they did not transfer properly. Finally, *Access* uses "queries" in place of stored procedures and queries did not transfer at all. To put it simply, the *JET* database engine is not scaleable and was ruled out as a viable option for the new version of MEIMS very early in the requirements analysis process.

Following the decision to use *Microsoft SQL Server* as the database engine for MEIMS, we realized that the majority of our personal experience with database design dealt with *Microsoft JET*. Our review of *MSDE* indicated that it had a lot to offer in terms of use with *Access* and *Visual Basic*. For example, the desktop engine supports record-level locking, transaction logs, operating-system integrated security under Windows 2000, and many other advanced features of full *SQL Server* (like replication) -- all from *Visual Basic* and VBA. In fact, we found that the *SQL Server* engine actually had a dizzying array of options, most formidable of which was the selection of programming interface to access the data in it.

Programming Microsoft Access and SQL Server - Microsoft *Access* has a history of notorious incompatibilities between versions. Since 1993, *Access* has undergone fundamental changes with each new release. *Access 2.0* applications used *Access Basic* rather than VBA and did not convert to *Access 95* format. *Access 95* implemented many new technologies and did not always convert to Access 97 format. Our personal attempt to upsize an old version of MEIMS from Access 97 to Access 2000 clearly demonstrated that there were problems with it as well. Based on this history alone we concluded that the next version of *Access* would no doubt have similar problems. The search for a method to use the RAD environment of *Access* but lessen the impact of version changes became paramount.

As mentioned earlier, *Access* has embedded support for *Visual Basic for Applications*. The SQL engine, however, is accessible via any language capable of creating COM objects. This realization presented a unique option for mitigating the effects of future *Access* version incompatibilities. Using *Visual Basic* or C++, we could design *ActiveX* object-oriented components that encapsulated much of the code that would normally be written within *Access*. These compiled components would reside outside of *Access* theoretically making them less susceptible to version changes and maximizing potential for code reuse. *Access* would just be a client shell and all business logic would be placed in these external components. The beauty of this approach is that the RAD methods of *Access* used to create forms, reports, and controls were still available. In addition, this approach is in keeping with the migration path of a small-scale application to a larger enterprise level one using OLE DB and DCOM. The location of the external components (either client-side or server-side) would define the architecture of the system (3-tier or n-tier).

Removing the business logic from *Access* allows MEIMS to grow by enabling modification of component code without making changes (or many changes) to the client code in *Access* or the database elements in *SQL Server*. Since code in the components is compiled, changes in versions of the programming language used to create them are much less significant over the lifespan of the program. We knew this would be especially significant for MEIMS because of *Microsoft's* upcoming release of new technologies like *C#* and *Visual Basic .NET*.

Regardless of the technology changes associated with these upcoming releases, current versions of C++ and *Visual Basic* should still be able to create compiled components compatible with new versions of *Access* and *SQL Server*. Since code is removed from the front and back-end *Microsoft* products, we believe that components are much less likely to suffer from versioning issues. The disadvantage of all this of course, is the inherent complexity in creating these components. As eluded to in the previous sections, the vast array of features in *Access* and *SQL Server* make creating components to take advantage of these products a very ambitious goal.

Based on our decision to implement components, our next decision involved selection of a COM compatible programming language. In keeping with the requirement for a Microsoft based solution our choices were either *Visual Basic 6.0* (VB) or *Visual C++*. Both C++ and VB are capable of implementing the four data access technologies that we knew we would need. Since *Access* provides inherent support for VBA and *Visual Basic 6.0* is a superset of this technology, VB could provide a single language for use in both *Access* and the components. C++, on the other hand, offered greater support in terms of threading (which is a serious deficiency in VB). A major disadvantage of C++, however, was its added complexity in a program designed for RAD. In the end, the idea of a using VB in all coding for HFACS was truly the key factor in weighing advantages and disadvantages. Our final choice for programming language was *Microsoft Visual Basic 6.0* using Service Pack 5.

Results - Our requirements analysis effort coupled with our research of data access technologies, programming languages, and trends in *Microsoft* products enabled us to develop a vision of the new MEIMS system. Armed with this information we set about creating a conceptual framework for the design of the system. As part of this process we inferred the following:

1) MEIMS should consist of a *Microsoft Access* client application using external compiled components to encapsulate business processes wherever possible. This would provide greater opportunity for code reuse and mitigate the effects of version changes in *Access*.

2) MEIMS would implement the *SQL Server* database engine and therefore should be developed to connect to an instance of MSDE as well as true SQL server. In order to facilitate differences in these connections, a component would be needed to perform management functions such as installation of the programs, installation of the database, logon

options, and starting and stopping the server. Management functions of this depth should be performed using SQLDMO and are specific to each client, therefore, this component must also reside on the client.

To facilitate these findings, our MEIMS development effort was divided into two phases. Phase I focused on development and implementation of the MEIMS Connection component. Phase II did the same for the MEIMS business logic component. The development of the connection component was executed first because it involved creating the foundation and environment for the business logic component to operate in. In addition to creation of the connection component and the inherent connection functions, Phase I involved creating the installation programs needed to deploy and configure all the pieces of this operational environment on a wide array of platforms supporting various editions of *SQL Server* and *Windows* operating system. Upon completion of Phase I, we possessed a much broader understanding of the SQL engine, which helped us immensely in developing database schema, classes, and interfaces for the business logic component in phase II. The high-level conceptual architecture is illustrated below.

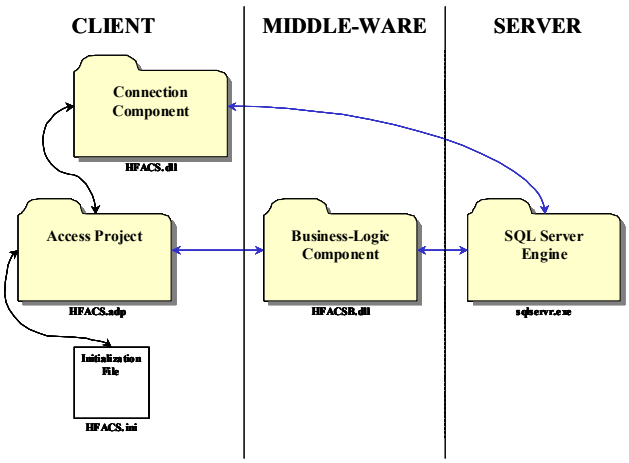

Figure 2: Conceptual Architecture

Within phases, we used Spiral Development Model (SDM) techniques. The SDM made the most sense to us because although requirements had been fairly well defined for MEIMS, there was still a substantial amount of risk associated with our lack of experience with SQL Server, object oriented programming with Visual Basic, and the Component Object Model. In addition, we knew that in the course of our development process, requirements might change. For instance, new requirements for the commercial aircraft version of MEIMS might arise. The SDM provides built-in methods for mitigating these risks through its use of development stages. Each stage was a normal development project producing a superset of the prior stage and yet a subset of the final system. Planning for each successive stage was structured to exploit the experiences of the former stages and to reduce perceived risk factors in the current and future iterations.

In the end, we hypothesized that the products produced using the methods that we have described will meet all our system requirements, greatly enhancing current desktop MEIMS capabilities and providing the means to weather further changes in requirements and application platforms. A further benefit of this environment is that it enabled all desktop MEIMS applications to connect with a true SQL Server DBMS on an intranet for updates. Using the Navy and Marine Corps Intranet or a civilian air carrier intranet allows all instances of the desktop version to be linked to a database that is behind a firewall. This keeps the mishap data current and confidential.

Finally, an additional benefit of using the SQL Server database engine for the desktop version is that the enhancements we made using stored procedures could be reused in an enterprise-level SQL Server DBMS linked to the web-based version of the system.

USER INTERFACE

MEIMS is designed to allow the user to access the database via three functional tools: (a) extracting data using user-created and predefined *queries*, (b) obtaining output from the database via *written reports* and *graphical displays*, and (c) inputting new data via forms. Each function is displayed on separate pages with controls providing user interaction. The system facilitates simple searches for novice uses and complex data-mining operations for more sophisticated users. The following paragraphs provide a description of the user interface.

The Main Menu allows the user to select one of five different options: (a) Query Menu, (b) Graph Menu, (c) Report Menu, (d) Add New Data, (e) Initiate Investigation and (e) Exit.

The Query Menu (Figure 3) provides the user with the ability to search by single field queries or multiple field queries. The single queries offer one of eight command buttons to extract data by one or more of its fields: aircraft model (F-14, H-46, etc.), aircraft type (tactical aircraft (TACAIR), helicopters, heavy aircraft, trainers, and others), branch of service of the aircraft (USN, USMC) or carrier name for the civilian variant, location of the mishap (ashore, embarked, and detached), mishap classification (A, B, or C), mishap type (Flight Mishap (FM), Flight-Related Mishap (FRM), or Aircraft-Ground Mishap (AGM)), and calendar year of the mishap (1989-1999).

When a single category control button is selected, a sub-menu appears allowing the user to define the exact description of the category via a combo box. Upon selecting the “View” control button, a Maintenance Mishap Query window is displayed revealing each instance (mishap) of the selected description. In addition, the user may page through all mishaps of the selection by selecting the right arrow on the bottom of the window. The data for each mishap is displayed in text boxes, with the selected category denoted with blue background. Additionally, maintenance related contributing factors to the mishap with their HFACS-ME codes are displayed at the bottom of the window. At any time the user may close a window and return to the previous sub-menu by selecting the

"Close Form" or "< Back" control. Each of the primary menus has a "Return to Main Menu" control which returns the user to the Main Menu when selected.

The query menu also contains control buttons that allow queries based on "Multiple Criteria" and multiple "HFACS-ME Elements". Selecting the "Multiple Criteria" control on the Query Menu allows the user to select any or all of the seven solo categories. A Multiple Criteria sub-menu appears and allows the user to "check" the desired categories and further define them by selecting criteria provided in combo boxes on the sub-menu. A summary of all mishaps sorted by HFACS-ME factors is also available.

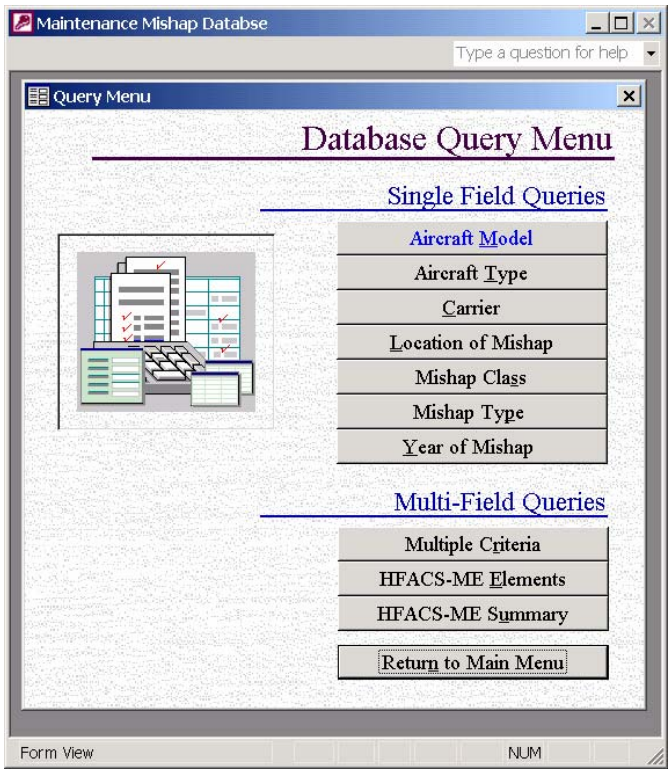

Figure 3: MEIMS Query Menu

The second option from the Main Menu is the Graph Menu. The user can create a two-axis, three-dimensional graph presentation. The x- and y-axes are populated with one of the following categories: aircraft model, aircraft type, mishap location, branch of service, mishap class, mishap type, HFACS-ME level one, HFACS-ME level two, and HFACS-ME level three. The user may then select one or more of the fields from each axis category sub-menu via a combo box. The resultant graph is presented in a three-dimensional, multi-colored view.

The Reports Menu is the third option a user may select from the Main Menu. The Report Menu offers eight reports which provide data listing the total number of mishaps and the number and percentages of mishaps by HFACS-ME levels one, two, and three. The user may select from the following distribution presentations: all mishaps, aircraft model, mishap class, mishap type, and mishap class by mishap type, branch of service, mishap location, and chronological listing by aircraft model. All reports are closed by selecting the "Close" control at the top of the window.

Add New Mishap is the fourth option in the main menu. This option allows the user to add data directly to the database tables.

The Initiate Investigation option takes the user to a Decision Support System that is designed to aide the investigator in determining the mishap cause factors as well as entering the required mishap data.

WEB-BASED DESIGN

A significant drawback to the original MEIMS tool was that it ran as a stand-alone application with the underlying database stored locally on the host computer. While this model provides for responsive data queries, analysis, report generation and local data entry, it did not support data import/export to a centralized database. Consequently, the data contained in the original MEIMS tool could be out of date before it was received by the end user. Newly entered or modified data remained unavailable to end users until the MEIMS application was updated.

To obtain the latest mishap information, users must contact the Naval Safety Center. In its current implementation, authorized personnel wishing to query the NSC Aviation Safety database must submit query requests via email or by phone. Technicians manually key in the search criteria. After the query is executed the results are forwarded to the requestor via email or hardcopy. This process may take several hours to weeks, depending on the query backlog and the requestor's location. To update MEIMS, the user must then manually enter this data.

This problem of updating MEIMS can be avoided by using an enterprise-level database management system (DBMS), updated with the latest information, tied to a web server. In this client-server environment, any user with a web browser can query the enterprise-level DBMS by navigating to the Uniform Resource Locator (URL) or "address" of its web server. A Hypertext Markup Language (HTML) page is returned that gives the user the ability to select options from drop-down lists, text input boxes and menu selections. The query criteria are then transmitted to the web server which contains the business logic and database connection information. The web server processes the appropriate HTML page or Active Server Page (ASP) and forwards the request for data to the database server. Once the database server returns the corresponding raw data, the web server formats the data according to the specification of the called page and returns the results to the browser. The total time to process the custom query and display the results to the user is measured in seconds rather than hours. The end user is able to access the HFACS-ME database in a fraction of the time and obtain up-to-date information that can then be used in training, hazard identification, and trend analysis to prevent possible future incidents. The resulting application takes advantage of a centrally managed and secured database while providing the ability to make the information available to the greatest number of authorized users.

However, the web based benefits come at a cost. There are numerous challenges that must be overcome to ensure that the data upon which the HFACS-ME Web Site is built remains secure. Mechanisms must be in place to authenticate users before allowing access to the data. Once authenticated, a user should not have to re-authenticate during a single session. This challenge stems from the "stateless" nature of the Hypertext Transport Protocol (HTTP) server, which view all page requests as independent and therefore does not maintain access credentials as the user moves from page to page by default. Data security/integrity as it traverses the Internet is also of concern given the open nature of the Internet. To ensure data is not viewed by unauthorized personnel or tampered with in route, security measures such as secure socket layer (SSL) may be in order. Other issues unique to the web implementation deal with concurrency and scalability. Concurrency deals with the problems resulting from two or more end users modifying the same data at the same time. In the current prototype there is no data input, deletion or modification implemented, so this issue does not pose a problem – yet. Scalability, on the other hand, has to do with the Web server and database server's ability to process hundreds, thousands, possibly millions of requests for data in a given period of time. We are currently using Microsoft Internet Information Server (IIS) and SQL Server 2000 for our prototype development effort and we expect both will remain viable options for the foreseeable future. A demo of our home page is located at: http://unixpreview.universal-net.com/hfacs_me/pages/home/default.php

The web-based version of MEIMS fits in well with the global mission of the Navy. And the future growth of Web-based capabilities provided by the Navy Marine Corps Intranet (NMCI) and IT-21 will provide a logical path for the Naval Safety Center to provide access to the Aviation Safety database. This vision is echoed in the Web Enabled Navy Architecture, dated 31 January 2000, where Admiral W. J. Fallon sees "A Navy in which operational and business processes are conducted worldwide via interconnected and interoperable Web-based IT systems."

DISTRIBUTED LEARNING MODULE

The real test for the system depends on whether the system adds value to the underlying mishap data, and ultimately, whether the end user gains knowledge that leads to effective intervention and mishap prevention. This hinges on the ability of the user to conduct iterative searches on the mishap data, analyze past maintenance mishap factors, extract trends and construct predictors of future problems. A pilot study found that many potential users were unfamiliar with query techniques or trend analysis and could not use the system to its full potential. Therefore, training in these areas is considered critical.

Since many of the potential users of organizational knowledge bases will be organizationally and geographically distributed, conventional classroom methods for instruction will not be feasible and tacit knowledge development by organizational socialization will be difficult. In addition to an on-line tutorial, web-based Asynchronous Distributed Learning (ADL) may be useful because it can give users the ability to learn at their own pace in an interactive environment (Dijkstra, Seel, Schott, & Tennyson, 1997).

An interactive ADL module is currently being designed to assist geographically dispersed users develop the ability to use the web-based version of MEIMS for data-mining maintenance mishap factors. It is anticipated that this will encourage the use of MEIMS by global users and ultimately lead to fewer maintenance mishaps.

CONCLUSIONS

Mathematical analysis of the HFACS-ME taxonomy has shown it to be viable model for discovering trends in aviation mishaps. Initial pilot testing of earlier models of MEIMS has shown it to have great promise. Our task was to reengineer an earlier version of MEIMS, redesign the database model and create a web-based version to enable distributed database-mining of current aircraft mishap data. We believe the enhanced desktop version and the new web-based version meet these requirements. When they are used with the upcoming ADL module, MEIMS will be a complete, robust system for analyzing aircraft maintenance mishap related factors anywhere at anytime.

REFERENCES

1. Department of the Navy (DON) (2001). Draft copy of *Naval Aviation Safety Program, OPNAVINST 3750.6R, Appendix O*. Washington, DC: Office of the Chief of Naval Operations
2. Dijkstra, S., N.M. Seel, F. Schott, and R.D. Tennyson. *Instructional Design*, Lawrence Erlbaum Associates: Mahwah, N.J. (1997).
3. Fry, A.D. (2000). *Modeling and Analysis of Human Error in Naval Aviation Maintenance Mishap's,* Master's Thesis. Operations Research Department, Naval Postgraduate School, Monterey, CA.
4. Nutwell, R., & Sherman, K. (1997). *Changing the Way We Operate*. Naval Aviation News, March-April, 79(3), Washington, DC.
5. Schmidt, J., Schmorrow, D., & Hardee, M. (1997). *A Preliminary Human Factors Analysis of Naval Aviation Maintenance Related Mishaps*. (983111), Society of Automotive Engineers, Inc.
6. Schmorrow, D. (1998). *A Human Error Analysis and Model of Naval Aviation Maintenance Related Mishaps*. Master's Thesis, Operations Research Department, Naval Postgraduate School, Monterey, CA.
7. Wood, B.P. (2000). *Information Management System development for the Characterization and Analysis of Human Error in Naval Aviation Maintenance Related Mishaps*. Master's Thesis, Computer and Information Systems Department, Naval Postgraduate School, Monterey, CA.

APPENDIX A - HFACS-ME TAXONOMY FACTORS

First Order	Second Order	Third Order
Management Conditions	Organizational	Inadequate Processes
		Inadequate Documentation
		Inadequate Design
		Inadequate Resources
	Supervisory	Inadequate Supervision
		Inappropriate Operations
		Uncorrected Problem
		Supervisory Misconduct
Maintainer Conditions	Medical	Adverse Mental State
		Adverse Physical State
		Unsafe Limitation
	Crew Coordination	Inadequate Communication
		Inadequate Assertiveness
		Inadequate Adaptability/Flexibility
	Readiness	Inadequate Training/Preparation
		Inadequate Certification/Qualification
		Personnel Readiness Infringement
Working Conditions	Environment	Inadequate Lighting/Light
		UnsafeWeather/Exposure
		Unsafe Environmental Hazards
	Equipment	Damaged/Unserviced
		Unavailable/Inappropriate
		Dated/Uncertified
	Workspace	Confining
		Obstructed
		Inaccessible
Maintainer Acts	Error	Attention/Memory
		Knowledge/Rule
		Skill/Technique
		Judgment/Decision
	Violation	Routine
		Infraction
		Exceptional
		Flagrant