2001-01

XML Technology Assessment - Progress Report (7/18/2000 - 12/31/2000)

Berzins, Valdis

Monterey, California. Naval Postgraduate School

NPS-SW-01-002

NAVAL POSTGRADUATE SCHOOL
Monterey, California

XML Technology Assessment

Progress Report (07/18/2000 – 12/31/2000)

by

Valdis Berzins

January 2001

Approved for public release; distribution is unlimited.

Prepared for: Joint C4ISR Battle Center
116 Lake View Parkway, Suite 150
Suffolk, VA 23435-2697

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

RADM David R. Ellison
Superintendent

Richard S. Elster
Provost

This report was prepared for Joint C4ISR Battle Center (JBC)
and funded in part by the JBC.

Prepared by:

[redacted]

Valdis Berzins
Professor, Computer Science

Reviewed by:

[redacted]

Luqi
Director, Software Engineering
Automation Center

Reviewed by:

[redacted]

Dan C. Boger
Dean of Computer and Information Sciences
and Operations

Released by:

[redacted]

David W. Netzer
Associate Provost and
Dean of Research

REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	01/25/2001	Technical Report 07/18/2000 - 12/31/2000

4. TITLE AND SUBTITLE
XML Technology Assessment

5. FUNDING NUMBERS
N4778300WRRD460

6. AUTHOR(S)
Professor Valdis Berzins

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Software Engineering Group,
Department of Computer Science,
Naval Postgraduate School, Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER
NPS-SW-01-002

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Joint C4ISR Battle Center
116 Lake View Parkway, Suite 150
Suffolk, VA 23435-2697

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Navy position, policy or decision, unless so designated by other documentation.

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

12 b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Objectives of this project are to assess the use of XML in reconciling the different formats of the same object in DoD database systems. We leveraged ongoing NPS faculty and graduate students' efforts on providing conceptual models and model-based experiments to address this objective.

14. SUBJECT TERMS
data interoperability, XML, database, software integration, computer software

15. NUMBER OF PAGES
471

16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT	18. SECURITY CLASSIFICATION ON THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

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

TABLE OF CONTENTS

CONTRIBUTORS

Faculty:

Berzins, Valdis
Luqi

Graduate Students:

Allen, James
Ehrhardt, Todd
Halle, Robert
Hina, Dave
Lyttle, Brian
Pradeep, Kris
Tran, Tam
Young, Paul
Zobair, Hamza

ABSTRACT

XML (eXtensible Markup Language) is a useful tool for accurate data interchange involving the same real-world object shared by different legacy systems. Reconciling different views of the same data is a major concern for data interoperability. The NPS Software Engineering Group proposed to evaluate and assess the use of XML in various DoD applications and standards. The project included the following sub-tasks:

(1) The methods of assuring the upward compatibility of evolving XML standards for supporting data interchange among legacy DoD systems were to be assessed in specific databases of JBMI (Joint Battle Management Integration) projects;
(2) The feasibility of using XML within existing database schemas for data interchange was to be assessed. The currently available XML-based COTS tools were to be evaluated to support the database queries; and
(3) The real-time overhead due to XML messaging was to be evaluated.

During the past year the faculty and graduate students in the NPS Software Engineering Group conducted research in these tasks and made substantial progress in these areas. This technical report provides the research results on three different levels.

(1) Section II gives a brief summary of results that address each of the research topics correspondingly;
(2) Section III contains a more detailed summary of each individual effort (both completed and those in process) that addresses the topics of interest; and
(3) Completed theses and thesis proposals are attached as appendices in Section V.

Several relevant directions for extending this research are listed in Section IV.

I. Review of Tasks for FY 2000

The main objective of the NPS project on XML technology assessment for JBC was assessing XML for DoD data interchange. There were three major tasks in FY00.

Task 1: JBMI XML Schema Integration

NPS was tasked to examine XML-based integration of four principal database schemas including

(1) Army Advanced Field Artillery Tactical Data System (AFATDS 98 – Release U.15)
(2) Global Command and Control System (GCCS) Track Database Management System (TDBM)
(3) GCCS Integrated Imagery and Intelligence (I3) Intelligence Shared Data Server (ISDS) General Military Intelligence (GMI) Database
(4) Army Joint Common Database (JCDB) as implemented in GCCS-Army (GCCS-A) version 3.2, Army Battle Command System (ABCS) version 6.1

DII/COE Integration & Run Time Specification (I&RTS), version 4, includes some specific XML guidance with respect to DII COE XML compliance. NPS was to provide recommendations for improvement of the process and summarize the existing duplication at the level of data elements and/or associated XML tags.

Task 2: JBMI COTS Database Management System (DBMS) Crosswalk with regard to Improved Interoperability Options

NPS was tasked to evaluate the hypothesis that XML can provide an effective approach to data exchange between heterogeneous databases that may have different schemas or data models. Specifically, several possible points were to be addressed:

(1) Feasibility of publishing selected parts of views of a database by using XML, and importing information from an XML page to a different database. To realize such connection, the control policies and software architectures were to be defined.
(2) Assessment of the cost and effort of creating XML structure according to a given database schema.
(3) Survey of currently available COTS tools for supporting database queries via XML, particularly for COTS DBMS systems supporting AFATDS (InterBase), JCDB (Informix), GCCS I3 (Sybase) and GCCS (Oracle).

Task 3. XML as a Data Exchange Medium in Real-Time Systems

Due to the real-time constraints in most DoD systems, the penalty of generating XML format as a data modeling and interchange standard from native data, and later parsing XML document into native data format, was to be assessed. Major points of study include:

(1) Feasibility of automatically generating a highly efficient parser/compiler for XML data
(2) Assessment of bandwidth penalties associated with the use of XML as a data interchange tool, and ways to minimize it.

Research on these feasibility assessments was to be applied in four DoD real-time or near real-time systems:

(1) Joint Data Network (JDN)
(2) Joint Composite Tracking Network (JCTN)
(3) Near Real Time Dissemination (NRTD)
(4) RADAR

II. Summary of NPS Solutions

JBMI XML Schema Integration

Four principal database schemas including AFATDS 98 (InterBase), GCCS-TDBM (flat file), GCCS-I3 ISDS (Sybase), and JCDB in GCCS-Army (Informix) were to be examined. Existing duplication at the data element level and/or associated XML tags with recommendations were to be summarized in the research.

The physical schema of these databases is very complicated. There are no existing XML-based analysis technique that could analyze the given databases. The AFATDS database schema has not been available so far.

An analysis process for determining commonality between database elements is proposed in the study (Robert Halle and Hamza Zobair). This process defines a step-by-step approach that can be used to seek out commonality between similar databases and support the growth of common XML standards as the legacy databases evolve. The approach can be applied to virtually any type of database including identified the legacy databases and future databases. The process certainly provides the capability to develop common elements, along with the database hierarchy/schemas.

Some of the XML Recommendations have not been finalized. Although most of them are nearing completion, they will take time to incorporate into software and database programs. To execute the proposed process, not only the databases, but also the supporting database software are required. However, some databases are not available. In addition, computing resources and available support software have been very limited, further hampering progress in this effort.

JBMI COTS Database Management System (DBMS) Crosswalk with regard to Improved Interoperability Options

This task includes: (1) the feasibility of publishing selected parts or views of a database by using XML, and importing information from an XML page to a different database. To realize such connection, the control policies and software architectures are to be defined; (2) assessment of the cost and effort of creating XML structure according to a given database schema; and (3) survey of currently available COTS tools for supporting database queries via XML.

NPS has been focusing on solving this task by assessing the heterogeneous DoD databases. NPS has assessed and compared the relevant technologies and COTS tools (Dave Hina). As the result, a description of XML and its advantages and disadvantages as a tool for interoperability is a necessary preface to any analysis of how and where XML can be applied to solving data interoperability issues.

The study covers the how and where by looking at relevant technologies and tools that currently exist and that can be applied toward resolving the issues in the task.

Another effort for the correspondent task is to evaluate the use of XML for integration of heterogeneous databases (Paul Young). Specifically, the study seeks to address the problem that differences in data representation pose to the integration of heterogeneous systems. A conceptual model, the Consolidated Type Hierarchy (CTH), is proposed to give a standard methodology in publishing and/or importing native database schemas via XML. For a group of heterogeneous systems being integrated, a specific CTH instance is developed for the federation. A system designer will create a CTH instance for the federation that captures the relationships between different system representations of a data object as well as the translations required to resolve such differences. The CTH instance will then provide the basis for adding computer aid to the process of reconciling such representational differences.

NPS also gives the first example of applying the Consolidated Type Hierarchy (CTH) in constructing a conceptual model of data interchange between two message formats in different XML schemas (Ehrhardt and Lyttle). It provides not only the data fields mapping from one schema to the other, but also the basic functional conversion (computations) on different representations of the same real-world object. The example demonstrates the conversion based on the execution of JavaScript functionality contained in XSLT stylesheets. Although a functional calling of external/legacy transformation components has not been completed because of limited time, the study shows the feasibility of applying any functional data type conversion by using existing COTS tools.

XML as a Data Exchange Medium in Real-Time Systems

The task includes the study on the feasibility of automatically generating a highly efficient parser/compiler for XML data, and the assessment of bandwidth penalties associated with the use of XML as a data interchange tool, and ways to minimize it.

NPS has prepared benchmark data samples to be used for bandwidth assessment (Pradeep). The main challenge was to understand the four databases of interest to JBC, and to decide what would constitute representative benchmark data. Due to the difficulties encountered in obtaining real data from the four databases, (for reasons ranging from Security issues to unavailability of data), the GCCS TDBM was chosen as a representative database. Using this database, programs were created that generated messages in both the USMTF and its parsed XML versions. These generated messages have random values but are within the legal limits specified by its unique message formats. The selected subsets of the database schemas are representative of the typical interactions that might be encountered in system operation.

Further assessment based on the XML-message and original formats is to be addressed in the next phase, which will perform timing assessments by measurement using the benchmarks and the translation approach enabled by (Ehrhardt and Lyttle).

Conclusion

The detailed assessments suggested that a major obstacle to data interchange among legacy systems is lack of agreement on data representations and conceptual data models. Especially when the scope of the required agreement is many large organizations, as is common in joint scenarios, agreement on the relevant standards may take many years, well beyond the time horizon of the JBC charter. The main conclusion of our study is that data interoperability is feasible without requiring a comprehensive data standard, and that methods for incrementally growing localized standards and bridging the gaps among them without requiring global agreement appear to be possible. Further assessments are proposed to determine the risks, costs, and detailed issues influencing the practical feasibility of applying this approach in DoD operations.

III. Results of NPS Research

The NPS Software Engineering Group has conducted correspondent research with respect to the three listed tasks. The results include three completed MS theses, four on-going MS theses, and one Ph.D. dissertation.

A. Evaluation of the Extensible Markup Language (XML) as a Means for Establishing Interoperability Between Heterogeneous Department of Defense (DoD) Databases, MS Thesis by Dave Hina

Hina's thesis focuses on solving Task #2 by assessing the heterogeneous DoD databases. A description of XML and its advantages and disadvantages as a tool for interoperability is a necessary preface to any analysis of how and where XML can be applied to solving data interoperability issues. This thesis covers the how and where by looking at relevant technologies and tools that currently exist and that can be applied toward resolving the issues expressed early in this document.

A.1. Survey of the technologies:

The thesis investigated the status of many commercial available tools and technologies, since many of the XML-related specifications and standards are still in their early stages. The technologies that are being widely researched include Extensible Stylesheet Language Transformations (XSLT), Extensible Query Language (XQL), Simple Object Access Protocol (SOAP), and XML Schemas.

A.2. Survey of the tools:

The COTS tools that are of most interest for this evaluation fall into one of two categories: middleware and parts of specific Database Management Systems.

The first category is XML tools that lie between the source and destination databases, and provide data translation, manipulation, and mapping services. They are relevant to utilizing a standard method for interfacing with a database such as Open Database Connectivity (ODBC) or Java Database Connectivity (JDBC), and then performing the translation from data elements to XML using a middleware solution that handles the mapping. In certain cases, when the data can not be exported directly via ODBC/JDBC due to legacy or security issues, some middleware solutions can still be used with success by using an existing API or some different method for extracting and updating the data.

Data Joiner of IBM provides integration capabilities on a large scale, such as transparent SQL access and relational joins across multiple different DBMSs. It also provides comprehensive APIs for working with data that cannot be exported via standard access, such as ODBC and JDBC.

Data Transformation Services of Microsoft operate independently of the DBMS to perform translations between a large numbers of formats. It has both offline and online processing modes, and while it does not yet have XML capability as part of the translation services, it can be used in conjunction with *Active Data Object* (ADO) technology to provide this capability.

One type of schema mapping and data translation tool is Microsoft's *BizTalk* products and specifications. *BizTalk* is a comprehensive set of products that provides a number of services at runtime, including document validation against a set of business rules, translation of data formats, schema transformation, document transportation, and tracking and logging capabilities. Its transformation and mapping capabilities are built on top of an XSLT engine, which can perform mappings between numbers of different formats. The *BizTalk* server provides the processing functionality that maps data to an XML stream based on a mapping provided by each organization. The schema used by the *BizTalk* framework is currently implemented in XML Data Reduced (XDR) and Microsoft has promised that the XML Schema standard will be used when it is finalized and released by the W3C. This can provide a significant advantage over the use of a DTD for specifying the schema and for document validation. One difference between *BizTalk* and some of the other middleware frameworks is that *BizTalk* is designed as an end-to-end product, and handles the transmission of XML streams from source to destination over a number of different possible transport protocols.

XML-DBMS is a model-driven, open source middleware solution for moving data between relational databases and XML, a typical open source tool. XML-DBMS has been implemented in both Java and Perl, and consists of an API to a set of packages that provide services for extracting data from a relational database into an XML format, and for taking data already in XML and inserting it into a relational database. The product uses ODBC and JDBC interface standards for accessing the data, so it can be used with Sybase, Oracle, SQL Server, or any other database server that has JDBC or ODBC drivers. In order to perform the mapping, an object view of the targeted XML format is developed. This object view is then mapped to the relational schema by taking the object properties, represented by XML elements and attributes, and linking them to specific columns in the database. This mapping is then performed at runtime whenever data is moved to or from the database. This method has the advantage of requiring no modification to the database.

The second category of tools involves utilizing the features of the different database management systems for accessing and storing data as XML. Although there are several types of databases in this category, including relational, object, and true XML databases, only relational databases are considered in the thesis since they are used by the legacy systems that are of interest. One of the tools which is discussed in greater depth, Sybase

Application Server Enterprise (ASE) version 12, represents the category that is specific to a single Database Management solution. This DBMS was chosen since it can provide a solution for data exchange from GCCS-I3.

There are many similarities in the functionality provided. The XML capabilities being integrated into databases fall into two broad categories: document-centric and data-centric. Document-centric capabilities focus on the storage and access of documents, which can be characterized by an irregular structure and larger grained database representation. In contrast, data-centric refers to a more regular structure where each XML element has a corresponding data element within the relational database. There are a number of advantages to using XML extensions to existing databases rather than third party middleware. In many cases, performance enhancements can be expected since better optimization can often be performed by the database vendors because they have access to the underlying structure of the database. Additionally, since data can often be stored or maintained in memory in XML format, instead of the tags and hierarchical relationships being established when the data is requested, gains in performance and architectural simplicity can result. Disadvantages to the use of database XML extensions include the fact that not all database systems currently have the same level of capability, and there are a number of differences in the ways that the capabilities are being implemented. Most major database systems now have some method available for exporting data as XML. Relational database systems that have integrated XML capability typically provide this capability in three different forms. One form allows XML-formatted documents to be generated from the individual data elements stored in the database. Another form involves extracting the data and structure from an XML document for insertion into a database. The third form of functionality allows entire XML documents to be stored as a single entity within the database. This discussion deals only with the first two forms and their specific application in the Sybase Application Server Enterprise (ASE) database. Sybase ASE currently lacks some of the XML capability found in other large DBMSs, such as Oracle 8i, DB2, and SQL Server. Specifically, these other DBMSs provide additional transformations on query results through the use of XSLT. Additional capabilities, such a publishing views as XML and facilities for conducting XML queries are not currently available within Sybase ASE.

Additional tools, including XML parsers, XML editor, and compression technology, are discussed as well.

A.3. Application

A process for applying them to one specific database architecture, GCCS-I3 is discussed in the thesis. It includes a section that lays out the design goals for the process, a description of the steps involved in the process, and a description of the GCCS-I3 MIDB schema, which are used for illustration. Following this is a description of a data model and examples of an XML-to-

relational-database mapping that could be used. The second part of the chapter applies some of the XML COTS tools and technologies to assist with the process. Finally, an assessment is made of the advantages and disadvantages to the data exchange process.

OMF was developed to provide annotations that would extend the existing weather reporting formats. In a similar manner, the process described and assessed below is designed to create a data-sharing environment that is usable by existing military systems, yet extensible enough to accomplish interoperability objectives. A basic set of design goals for our process follow:

- The data exchange process must handle multiple databases, each with a different schema, running under different DBMSs, and containing data, some of which has the same or similar semantics.
- The schemas for the existing databases cannot be modified.
- Where appropriate, data must be easily transformed between different message formats, including, but not limited to, USMTF, CIX, NATO Allied Data Publication Number 3 (AdatP-3), and Theater Ballistic Missile (TBM) track format.
- Operations that act on the XML schema must allow the schema to change over time.
- The XML schemas must be simple to construct using data element definitions derived from a central repository.
- Use of standardized technologies and COTS tools must be used wherever possible.

The primary step of interest, since it will have the most effect on the XML toolset to be used, is to develop the data model and method for performing relational to XML mappings. This will be discussed in some level of detail in relation to the Modernized Integrated Database (MIDB). Other steps, some of which will be covered in lesser detail include the following:

- Establish a common vocabulary.
- Analyze legacy systems to determine the subset of data to be shared.
- Analyze legacy systems to obtain data structures, semantics, etc.
- Determine mechanism for accessing the data (e.g. stored procedures/ triggers, special APIs, ad hoc queries, etc.)
- Determine an XML Data transformation engine
- Determine mechanism and protocol for data transport.
- Determine tools that can be used to assist with or provide the capabilities needed for each of the items above. This includes determining the amount of custom code that will be required and the amount of risk involved with specific tools and technologies.

A.4. Conclusions:

It is possible to utilize XML and its associated technologies to greatly reduce the barriers to data interoperability. This issue is very important in the face of

past failures in this area. It is clear from the previous assessment, however, that this is still not an easy task, there are many choices to be made, and the process is not risk free.

One conclusion that can be drawn about each of the products and technologies that have been reviewed, is that they are relatively new to the market and they each represent the first generation in their respective categories. The result of this has been a lack of satisfactory case studies from which to draw conclusions. This fact, although it will certainly change over time, will increase risk for investments in products and technologies.

It is also evident from the study that there is no single product or technology that can be expected to accomplish each of the goals. Further, it appears that no single vendor seems to dominant in this area, and that each of the individual products works well with others. The overall affect from this is an increased modularity within the architecture and decreased risk from not having a reliance on any single product or technology.

The most established set of products and technologies appear to be the middleware tools for performing data mapping and translations. This is true in terms of the number of products available and in their functionality. The XML-enabled DBMSs are mostly still in the early stages of adopting XML and, in the case of Sybase, appear to be lacking in functionality.

It is clear that some of the steps listed in Section A.3 will not be easily accomplished. Analyzing the existing legacy structures is never an easy task, even when using COTS products to assist with evaluations. Much of the necessary information is buried in application code that can be very time consuming to analyze.

One of the challenges that must be overcome in translating between the extended messaging discussed here and standard message formats is handling the set of restrictions that standard messaging formats place on the structure of the messages. One example is the allowable line length and message length for the OTH-G format. OTH-G message lengths are limited to 100 lines, and line lengths are restricted to 69 characters.

In surveying the COTS tools available on the open marketplace, it is clear that there are important distinctions that need to be made between the diverse set of requirements that exist within the DOD when it comes to data interoperability, and the more narrowly focused requirements presented by standard business applications. While many tools are being developed specifically to address data interchange between heterogeneous database systems, tools that are designed for use in the business to business arena are not always suited to handle some of the requirements that arise due to the size and diversity of data within the DOD.

B. Integration of Heterogeneous Software Systems Through Computer-Aided Resolution of Data Representation Differences, Dissertation Proposal by CAPT Paul Young, USN.

CAPT Young's dissertation proposal is directed at extending the Task 2 effort to evaluate the use of XML for integration of heterogeneous databases. Specifically, he seeks to address the problem that differences in data representation pose to the integration of heterogeneous systems.

The goal of enabling data interchange between systems is the attainment of a federation of cooperating, autonomous, heterogeneous software systems. Achievement of such a federated system is accomplished using a two-phased process. In the first phase, accomplished prior to runtime, a formal model for capturing the relationships between producer and consumer data elements is defined. In the second phase, the model developed prior to runtime is used to automatically translate between heterogeneous data representations.

Pre-runtime phase

In the pre-runtime phase, a model for establishing relationships between producer and consumer types is defined. This model, termed a Consolidated Type Hierarchy (CTH), utilizes an object-based model to capture relationships between data elements in producer and consumer systems, translations required to convert between different representations used by a producer and consumer system, and information used to establish relationships between producer and consumer data elements.

The CTH model consists of three basic object types, as depicted in Figure 1. The first object type, the ProducerType, contains the producer's view of an exported data element in the form of a ProducerTypeSchema. Similarly, a ConsumerType contains the consumer's view of a data element being imported in the form of a ConsumerTypeSchema. In order for the interchange of data between a producer and a consumer to have meaning, the exported data element and the import data element must both be representations of the same real-world object. Any differences in the representation of the real-world object are resolved by introduction of a third object type in the CTH, termed a ConsolidatedType, which provides a "standard" or normal representation for the real-world object, as captured in the ConsolidatedTypeSchema. In a sense, two or more different representations of the same real-world object are consolidated into a standard ConsolidatedType that is added to the CTH.

The ConsolidatedType maintains a representative of relationship with the producer and consumer types that depict the same real-world object. The representative of relationship is defined to mean that the ConsolidatedTypeSchema contains a representative of each of the elements contained in every related ProducerTypeSchema and ConsumerTypeSchema. The ConsolidatedType representation can be a mirror of either the ProducerType

or the ConsumerType representation or it can be different from either. The key consideration is that the ConsolidatedType representation be of sufficient fidelity that the meaning and precision of the data is preserved when converting between different representations.

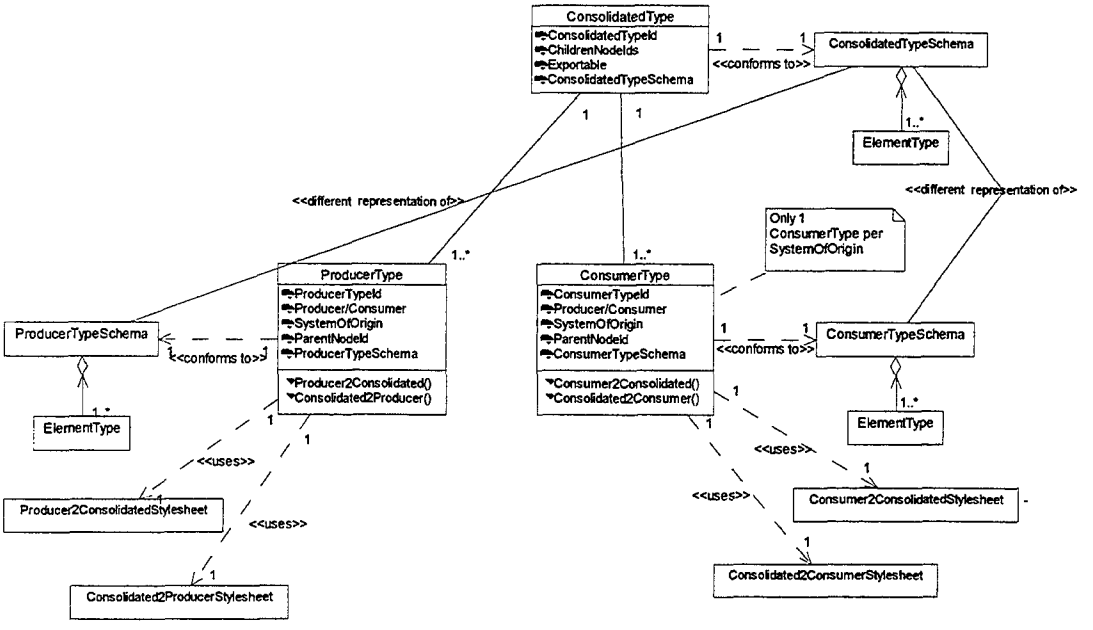

Figure 1

In addition to capturing the relationships between producer and consumer system data elements, the CTH contains translations required to convert a producer's representation of an object to the consumer's representation. Finally, the CTH contains attributes used to determine the data type relationships between consumer and producer systems. These attributes include both structural and semantic information about a data element. This information is used to bring computer aid to solving the problem of establishing whether two data elements are representations of the same real-world object.

Use of the eXtensible Markup Language (XML) for implementing the CTH

In order to assist the data transfer and conversion process, use of a standard method for representation of the abstract CTH model is proposed. One possible representation, the eXtensible Markup Language (XML) offers a mechanism to separately identify the elements in the abstract data model along with methods required for implementing the conversion process.

The first function of the CTH model, capturing the relationships between producer, consumer, and consolidated types, is accomplished through the use of XML's capability to define a hierarchical structure for the CTH document or by features of the XML schema that allow links between document elements.

The second aspect of the CTH model function, capturing the translations required to transform a data element from a producer to a consumer representation, is accomplished through the use of eXtensible Stylesheet Language Transformation (XSLT) stylesheets. In the CTH application, an XSLT stylesheet defines the translations required to convert from a producer to a consumer representation as a sequence of data element attribute mappings and functional transformations. Given a producer message and the appropriate XSLT stylesheets, the proposed translator will invoke an XSLT engine to transform the data element into the proper consumer representation.

Finally, structural and semantic information about a data element that can be used to determine the data type relationships between consumer and producer systems is also stored in the CTH.

Consolidated Type Hierarchy development

The Consolidated Type Hierarchy is constructed for a federation of heterogeneous systems from the data elements defined in each system's external interface. Construction of the CTH is an incremental process involving the following computer-aided human activities: 1) *Registration*, 2) *Discovery*, 3) *Consolidation*, and 4) *Reconciliation*. *Registration* provides the means for adding producer and consumer data elements to the Consolidated Type Hierarchy. Registration utilizes the *Discovery* process to assist the system designer in determining whether there are any producer data elements relevant to a consumer element being registered. Once determined by the system designer that a producer and consumer data element are both representations of the same real-world object, the *Consolidation* process establishes the required relationships between the producer and consumer objects. Finally, in the *Reconciliation* process, the system designer is aided in defining the mapping and translation functions necessary for reconciling representational differences between producer and consumer types.

CTH development environment

One of the benefits of the CTH model is that it readily supports application of computer aid to building a CTH document that defines a specific federation of component systems. It is expected that computer aid can be applied in the following areas:

a) Registration of producer and consumer types,
b) Discovery of produced type(s) satisfying a consumer type request,
c) Creation of consolidated types as the canonical representation of a producer-consumer relationship, capturing the relationship between consolidated, producer, and consumer types, and
d) Development of translations to convert a producer representation of a data element to its appropriate consumer representation.

Runtime phase

The Consolidated Type Hierarchy (CTH) document constructed during the pre-runtime phase for a specified federation of component systems is used to resolve the data representational differences between data elements from the different systems. Reconciling representational differences is accomplished at runtime by a *translator* that serves as an intermediary between component systems.

The translation function is anticipated to be implemented as part of a *software wrapper* enveloping a producer or consumer system (or both) in a message-based architecture, or as part of the data store (actual or virtual) in a publish/subscribe architecture. A software wrapper is a piece of software used to alter the view provided by a component's external interface without modifying the underlying component code.

For either type of architecture, the function of the translator is similar. In both cases the decision of which producer type should be linked to which consumer type and what translations are required to convert an instance of the producer type representation to an instance of the consumer representation, is determined by the CTH.

Conclusion

In summary, the Consolidated Type Hierarchy model serves as the foundation for automating a process for resolving data representation differences between autonomous, heterogeneous software systems. From the CTH model, a federation-specific hierarchy is developed for the included component systems. Computer aid is applied in the development of this hierarchy to assist the system designer in locating relevant data producers and consumers and in defining the translations required for resolving data representation differences between systems. Finally, the resulting producer-consumer relationships and translation definitions are used to automate resolution of data representational differences in the federation.

C. Interconnectivity via a Consolidated Type Hierarchy and XML, MS Thesis by LT Todd Ehrhardt and CAPT Brian Lyttle

Ehrhardt and Lyttle's thesis gives the first example of applying the Consolidated Type Hierarchy (CTH) in constructing a conceptual model of data interchange between two message formats in different XML schemas. It provides not only the data fields mapping from one schema to the other, but also the basic functional conversion (computations) on different representations of the same real-world object. The example demonstrates the conversion based on calling JavaScript functions in XSLT. Limited by time, a functional calling of external/legacy transformation components is not completed in this thesis.

C.1. Assumptions and Preparations:

During Ehrhardt's and Lyttle's research, they observed Phase Two of the Joint Battle Management Initiative (JBMI) experiment. JBMI sought to prove XML as a valid technology for improving inter-operability and inter-connectivity between systems. All four services provided computer systems for the experiment.

Joint Battle Center (JBC) defined two different levels of sharing information between systems compliant with the DII COE. Interoperability at its highest level allows systems to import and export information as if the remote site were actually part of the user's system. Inter-connectivity is several steps lower, and allows systems to pass limited messages between different systems.

During JBMI, JBC demonstrated the benefits of XML as an enabling technology for achieving cross-platform and cross-service inter-connectivity between legacy command and control systems. The primary system utilized for the experiment was the Navy's Global Command and Control System (GCCS) I3. The U.S. Army also provided the Advanced Field Artillery Tactical Data System (AFATDS), a member of the Army Battle Control System set, and the command and control system for all ground fire-support systems in both the Army and Marine Corps. In addition, JBC integrated two devices, a Palm Pilot V (a personal digital assistant) and a cellular telephone, which are currently available on the commercial market, into the JBMI architecture. JBC programmed the simple USMTF Call for Fire and Observation Report messages into the PDA, and the same ability into the cellular telephone. The two devices communicated by using the Wireless Application Protocol to the networked systems.

All the systems connected via a hardwire LAN into a web server. The web server allowed each unique system to subscribe to a message set or an individual message type from the USMTF. As each legacy system produced a message, a software wrapper transformed the message into an XML formatted message. These XML formatted messages were used as a common data representation for communicating between the various component systems and commercial devices.

During the JBMI experiment, XML presented a means to accomplish interoperability between systems. It allowed messages to be transformed from native legacy format into XML and then to be used in a different system. However, the engineers were required to write source line code in Java to accomplish this. It is believed that using XML and other COTS tools along with a different methodology can accomplish interoperability between systems cheaper and faster than writing source code. The focus of Ehrhardt's and Lyttle's research and experimentation was to show how the use of XML and associated commercially available tools could be used in conjunction with CAPT Young's Consolidated Type Hierarchy model for achieving inter-connectivity between heterogeneous software systems.

C.2. Results

Ehrhardt and Lyttle used the Consolidated Type Hierarchy model and XML to demonstrate the capability to translate between different representations of the same real-world object. They started by creating two XML documents, one to represent a fictitious Army message format (Figure 2) and the other, a Navy message (Figure 3). They created individual schemas for each message and then created a global schema that incorporated elements from both message formats (Figure 4). They then created a file called CT.XML to show the relationships between the elements of the global schema and its constituent schemas.

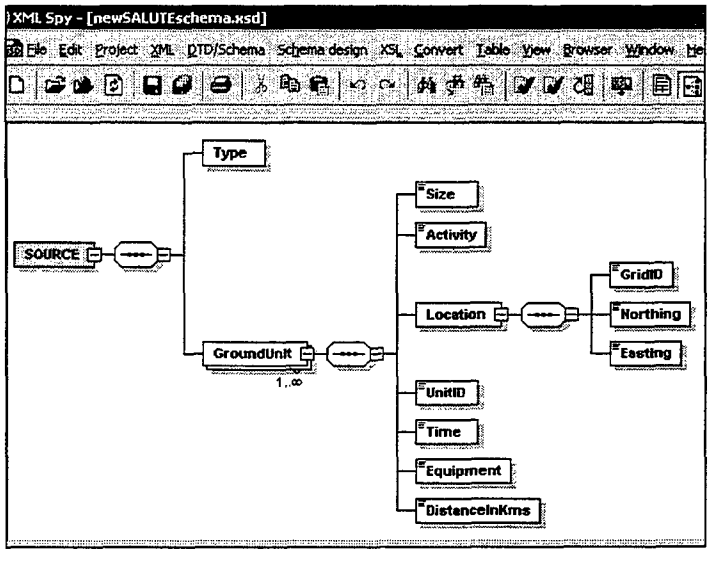

Figure 2. Schema for the Army Salute message format from XML Spy

After entering correspondences between the message formats within CT.XML, they created stylesheets to translate an instance of the Army SALUTE message into an instance of the Navy Track Message. Translation was accomplished through the use of an intermediate representation, which conformed to the global schema. This required the creation of four stylesheets to perform the upward and downward translations for both Army and Navy message formats. In order to test the modularity of the XSL stylesheets they created two additional stylesheets to handle the translation of positions, going from MGRS format to latitude-longitude format. Translating from MGRS to latitude-longitude requires the use of capabilities the W3C implementation does not support. Functional code is required in order to perform calculations on the data contained by an XML document. The Microsoft implementation of XSL supports JavaScript and Visual Basic Script (VBScript) functions that provide this capability. It uses the `xsl:eval` statement to invoke script functions from those two languages. Their implementation demonstrated XSLT's capability to invoke a functional transformation for a user's specific needs, such as converting miles to kilometers.

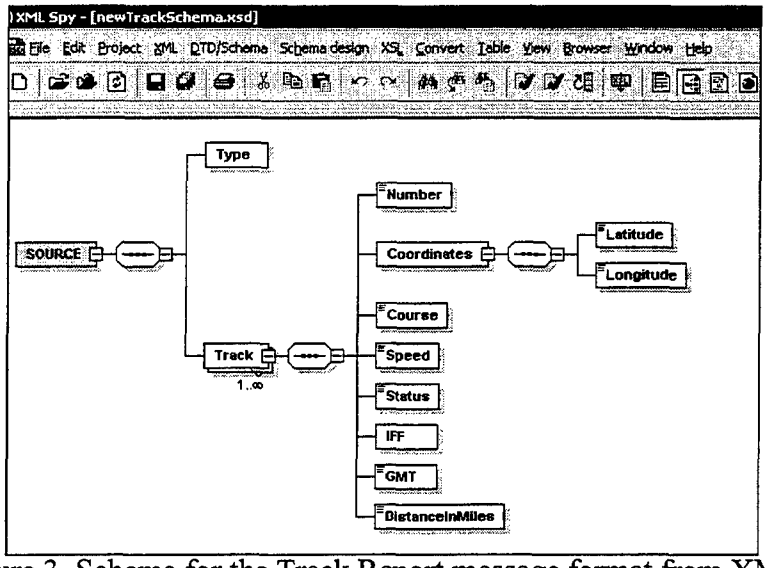

Figure 3. Schema for the Track Report message format from XML Spy

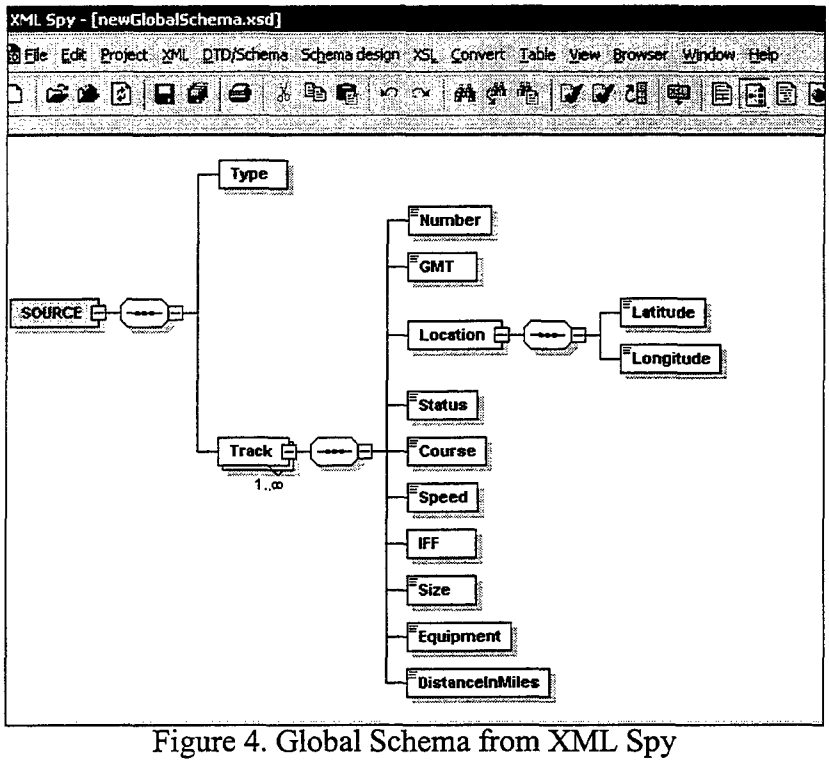

Figure 4. Global Schema from XML Spy

C.3. Recommendations for Prospective Work

Much of the future work remaining on this concept involves constructing the Consolidated Type Hierarchy. To ease the process, some method of automated discovery of types within each message must be found. A computer aided type

discovery process will help reduce the manual correlation of data types by the designer.

Another application that would make the CTH easier to use is the semi-automated generation of the stylesheets. Once a message format has been mapped to the global schema, and the translations for individual elements have been identified in CT.XML, then the program should be able to automatically generate the stylesheets that translate entire messages to and from the global schema.

The best method of implementing the CTH may be in a publish/subscribe architecture. As the different systems log into the networked battlefield, the system would request to receive messages of a certain type. As each individual legacy system sends data over the network, an XML wrapper would intercept the message. The wrapper would mark up the message into an XML representation of the message in the CTH, and then send it to a web server. The web server would check the list of valid subscribers for that message format, and send the message to those destinations. The destination system's XML wrapper would translate from the CTH mark-up form into the correct legacy system format.

The CTH is a powerful model that will allow more than just message systems to exchange information. It could be used for object-oriented databases, as well as source code files and eventually any other kind of data. An application of this nature would allow more reuse of previously developed code and reduce development time and costs. An issue that remains to be investigated is the degree of overhead relative to real-time constraints and optimization methods for mitigating time and space overheads.

D. Interoperability and Security Support for Heterogeneous COTS/GOTS/ Legacy Component-Based Architecture, MS Thesis by Tam Tran and James Allen

The thesis was targeting the previous JBC task with regard to the COTS tools available for interoperability research and was finished within FY2000.

This thesis researches existing open standards solutions to the distributed component integration problem and proposes an application framework that supports application wrappers and a uniform security policy external to the components. This application framework adopts an Object Request Broker (ORB) standard based on Microsoft Distributed Component Object Model (DCOM). Application wrapper architectures are used to make components conform to the ORB standard. The application framework is shown to operate in a common network architecture. A portion of the Naval Integrated Tactical Environmental System I (NITES I) is used as a case study to demonstrate the utility of this distributed component integration methodology (DCIM).

D.1. Surveys

Existing solutions to the distributed component integration problem are studied. The thesis proposes a methodology that can be used to transform desktop legacy applications into distributed web based applications. A design pattern application framework encompassing security and wrappers is presented and applied to the case study.

Existing solutions to the interoperability problem include Generic Security Service Application Program Interface (GSS-API), Kerberos, Secure European System for Applications in a Multi-vendor Environment (SESAME), Distributed Computing Environment (DCE), KryptoKnight, Windows NT Security Model, DCOM, Java, CORBA, Secure Sockets Layer (SSL), Secure Hypertext Transfer Protocol (S-HTTP), IP Security (IPSec).

GSS-API is emerging as an Internet standard for securing applications. GSS-API is embedded in Common Object Request Broker Architecture (CORBA), Kerberos, Distributed Computing Environment/Remote Procedure Call (DCE/RPC), Sequence Packet Exchange (SPX), KryptoKnight, and SOCKS. It is an interface specification that is independent of implementation mechanism, independent of placement, and independent of communication protocol.

The primary goals were to provide single logon to a network of application servers and protect authentication from masquerading attacks. Kerberos is an implementation mechanism for GSS-API. Kerberos assumes the client, network and server cannot be trusted and that a third party key distribution center (KDC) is needed to store secret keys. The KDC is composed of two logical entities, the authentication server (AS) and the ticket-granting server (TGS). Kerberos has several weaknesses. The user's secret key is stored in the host's memory during AS exchange. Kerberos is vulnerable to password guessing attacks. Registering each service with the KDC does not scale. Applications must be modified to take advantage of Kerberos.

Sesame is the European substitute for Kerberos. Sesame implements all the specified security services. There is a project underway to convert Sesame to Java in order to improve portability.

DCE is the Open Systems Foundation (OSF) specification for DCE includes facilities for security, directory services, time services, threads and remote procedure calls. DCE 1.2 is compatible with Kerberos V5 so single logon and mutual authentication services are available. DCE uses Access Control Lists (ACLs) for authorization. Role based authorization is not available. Like Kerberos, DCE/RPC uses a session key to provide secure communication services between the client and server. A rich set of APIs, including GSS-API is available to the programmer. These APIs provide data confidentiality and integrity services.

Kerberos influenced the design of KryptoKnight. The 2-party, 3-party and inter-domain protocols are designed to minimize network usage and computer processing.

The NT security model has three major components: the logon process, the security reference monitor, and other security subsystems.

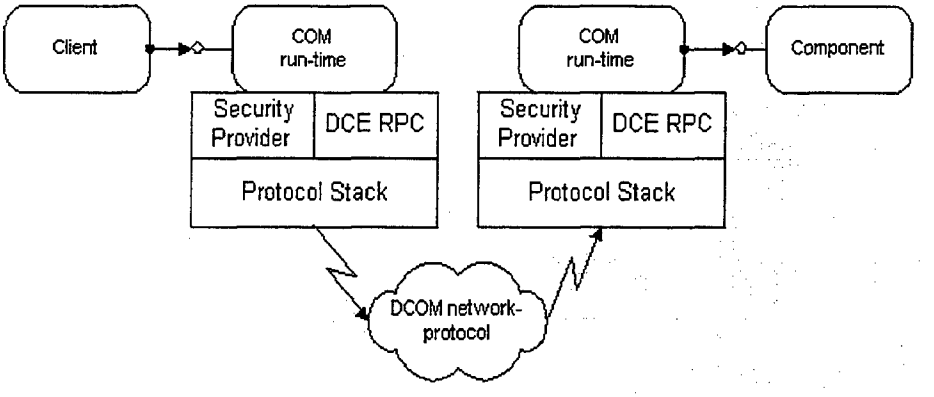

Figure 5. Overall DCOM Architecture

DCOM can provide security services for COTS components externally by using the DCOM configuration tool or by embedding security API calls within components (Figure 5). The primary DCOM security services fall into three categories: access, launch and call. DCOM is layered on Object Remote Procedure Call (ORPC) which is an extension of DCE RPC. These services are accessible through the WIN32 Security Support Provider Interface (SSPI). DCOM can also accommodate multiple third party security providers. DCOM uses the Windows Registry and the ACL facilities of the Windows NT operating system. DCOM is also available on Macintosh and UNIX platforms.

The java.lang.SecurityManager class implements the applet security restrictions. A security policy is created by instantiating and registering a security manager object. A potentially harmful operation causes an exception that is handled by a security manager method.

The Common Object Services specification (CORBASec) describes security related tasks and requirements needed for CORBA. A CORBA ORB, ORBacus, from Object Oriented Concept Inc. has been used to implement some specified security services. Security Level 1 provides security services for applications that are unaware of security including mutual authentication, confidentiality and integrity. The security functionality underneath is that of Kerberos V5 and is accessed through a Java binding of the GSS-API.

SSL is positioned between the TCP/IP application and connections layers enabling multiple services such as Telnet, HTTP and FTP to establish secure connections without modification to the services. SSL utilizes RSA

Public/Private key architecture. The server identity is validated to the client by x.509 digital certificates.

S-HTTP permits parties to negotiate symmetric or asymmetric keys, key management technique, message formats, and cryptographic strength. S-HTTP allows for multiple trust models to be negotiated between client and server. Security features are specific to the HTTP protocol.

IPSec provides for secure transfer of IP packets across an untrusted network. IPSec resides at the network layer of the OSI model. IPSec is transparent to protocols at higher layers in the OSI model. IPSec is an open standard for encryption on an IP network.

A generic wrapper for system components is designed to fit for the conditions as follows:

- Components pass messages synchronously or asynchronously.
- Components may have real-time constraints.
- A hierarchy of interacting COTS, GOTS and custom components may be assembled to form an application.
- Implementation will be dependent on the security services of the host operating systems.
- Security policies need to evolve and policy implementations need to be manageable in a distributed computing environment.
- Some components may be in binary executable form where compile or link is not possible. Other components may be re-linked but not recompiled. Other components may not be re-linked but substitution of dynamic load libraries (DLL) is possible. Other components may be modified at the source code level and recompiled.
- The security services will not be exported outside of the United States.
- Attacks can come from inside or outside an organization.
- This security system must be adaptable to counter new kinds of security attacks.
- The target systems will operate at a single level of security at no higher than the discretionary access control level (C2).

Utilization of XML within wrappers makes data transport mechanism independent of language or operating system.

D.2. Case Study

A subset of the operational NITES system was chosen for the case study. This subset is representative of the issues involved in the integration of COTS software components where only the executables are available. The case study covers the wrapper and security aspects of component integration. The wrapper transforms COTS applications into a COM/DCOM component enabling interfaces with infrastructure components

D.3. Conclusions

Based on application of the distributed component integration methodology (DCIM) to the case study, the thesis has the following conclusions.

DCOM: DCOM is a natural choice for this implementation. The host machine is a PC running Windows NT and DCOM is bundled with the OS. There is familiarity with DCOM from prior projects. Visual Basic development environment hides low-level plumbing from the developer. Security policy can be defined external to the component implementation. The existing design pattern template fit the design of the continuous brief application.
DCOM proved to be a quick and efficient way to implement a robust continuous brief application. Components were tested in the VB debug environment. Then executables were tested on a single machine. Finally, the system was distributed to the Web server machine. No source code changes were made to execute in these three configurations.

ARCHITECTURAL: The architectural design with accompanying VB application framework skeleton code proved to simplify implementation. The details of object creation, push technology, client registration for service, event processing, browser-based components, asynchronous object execution, and polling were provided by the framework.

The framework was extended to poll a directory, make asynchronous database queries, add arguments to events, wrap PowerPoint and add a user interface. The developer is able to focus on the application without being distracted by plumbing details.

WRAPPERS: Three types of wrappers were used in the implementation of the continuous brief: file type in directory, object, and COTS API. The monitor component of the architectural design was extended to periodically check for a new satellite image file in a directory specified by the configuration utility. The object wrapper used the file name structure to extract image time, type and location. The PowerPoint API was used show the continuous brief. Even though the show could have been easily implemented using a Java applet, PowerPoint could simplify future extensions such as image cropping and image titling.

SECURITY: The external security features of DCOM proved to simplify implementation of security policy; however Windows NT Service Pack 5 does not expose DCE encryption to external DCOM security. Single user logon, user privileges based on role and discretionary access control were available.

ImgNT: Administrative problems precluded the use of ImgNT to retrieve selected images from a database and store in a directory. The system had not been installed on an unclassified system, Visual Basic was not available, and ImgNT

patches had not been made. It is assumed that ImgNT had already stored requested images to a directory.

D.4. Future Work

The value of the results of this thesis is time sensitive. Research on this thesis began in April 1999. Since that time Microsoft has released Windows 2000, SPAWAR has unveiled a public key infrastructure for e-mail, SPAWAR has a draft security policy, a network centric architecture has been deployed to the USS Coronado, CORBA has a wider selection of commercial ORBs, new standards for wireless communications have been developed, Linux is gaining support from many communities, security measures are receiving higher priority and many other innovations.

The distributed component integration methodology described in the thesis will remain in the mainstream for the foreseeable future. Independently designed components will need custom integration using some form of wrapper. Network administrators will require implementation of security policy using tools external to the application.

E. TACOM Work

a. XML Schema Integration, ongoing thesis by *Robert F. Halle*

Focus:

This effort will execute an analysis of database components and of XML based database analysis schemas. The results of this analysis will be used to develop a recommendation on how common data elements can be identified using XML based analyses. These identified common elements can then be employed in supporting scalability of the databases to meeting the growing C4I requirements. Originally proposed databases to be analyzed included AFATDS (Interbase) and JCDB (Informix). Due to the unavailability of the AFATDS database, an in-depth examination of a more universal XML analysis approach will be presented in this research effort. This XML database analysis approach could be employed to identify common elements between most types of databases.

This thesis describes an XML based analysis method that could be used to identify equivalent components of similar databases. The Department of Defense currently has multiple databases to support command and control of some portion of the battlefield force. Interoperability between forces will be crucial as the force structure continues to be reduced. This interoperability will be facilitated through the integration of these command and control databases into a singular joint database or by developing inter-communication schemas to support inter-database communications. The first step in either of these alternatives is the identification of equivalent components/elements between the multiple databases.

This thesis will describe how XML can be used to facilitate the process of equivalent database component identification. Each step of the process will be described in detail accompanied by explanations of the XML tools/resources required to execute the step and rationale of why the step is necessary. Detailed graphics and examples will be employed whenever possible to simplify and justify the step-by-step explanations. This thesis will conclude with discussions of the overall value of this XML based analysis process and potential future work that could be pursued to further exploit this XML process.

Progress:

Identifying methods of common data elements and schemas is the primary problem faced in this research task. Both the database element and database schemas are needed to be examined/compared when the databases are being analyzed. Another problem faced was the lack of the AFATDS database. There are no XML-based database analysis approaches. Halle examined XML-based tools, schemas, etc. that could be employed in the development of a database analyses approach that could meet the objectives identified in Task #1.

The thesis report will define a step-by-step XML based approach that can be used to seek out commonality between similar databases and support the growth of common databases as the legacy databases evolve.

To support the analysis of the database schema, the process begins with converting dissimilar databases #1 and #2 to XML documents. The next step is to develop DOM representations of each of the databases. This will allow a person to get a representation of the parent/child relationships of the database that comprise the database hierarchy/schemas. Using XSL, these relationships can be visualized. Another way to represent the database hierarchy is through the use of the XML Information Set (Infoset).

Infoset basically provides a common vocabulary to describe the contents of an XML document. Infoset provides the opportunity to get a different representation (in some cases more detailed) than that of the DOM. It must be pointed out that the Infoset Recommendation is still a relatively new draft that is undergoing significant change.

A parallel effort consists of conducting a data element analyses between database #1 and #2. The objective of this step is to identify any common elements between the two dissimilar databases. This step is not concerned with the database hierarchy. Instead, only the location of the common data elements is concerned. The identified common elements will be used as the basis to execute the analysis steps discussed in the following paragraphs. A number of analysis tools and schemas are available to conduct this type of analysis. Hamza Zobair has identified some of these analysis techniques in his research efforts.

The value of conducting manual analyses of the two databases to locate common elements will be discussed in next step. This step consists of analysis of the previously developed DOM representations of the two databases. It must be noted that the DOM representations of each database (i.e., JCDB) would be extremely large and complex. Basic search techniques of the DOMs would not work well if a person were trying to locate a specific portion of the database. XPath/XPointer and/or XML Search (identified in Dave Hina's thesis) would be used to search each database for the common elements identified in the previous analysis step. Each of these analysis techniques of the DOMs allows one to extract the common data element along with the associated database hierarchy/schemas. This step of the analyses process will examine how XPath/XPointer works along with examining other XML based capabilities. The objective of this step is to describe the means by which the common elements and the hierarchy can be extracted from the extremely large DOMs.

This final step in the process examines how the extracted portions of each database can be examined to determine if they are truly common. This will consist of the reexamination of the data elements and of the individual database hierarchies. The parent-child relationships, attributes, values, etc. will be examined. This step will primarily be a manual examination. It is hoped that the extracted parts of the databases will be at a simple enough level that manual examination techniques will be the simplest and most efficient to execute.

Conclusions:

Task #1 was divided into three sub-tasks: (1) determine methods for assuring the scalability of the solution to legacy systems and migration to satisfy C4I requirements; (2) determine what parts of a legacy system view could be developed from the previous shared schema; and (3) determine how to develop those parts relevant to such an assessment. The research completed shows that both (1) and (3) are satisfied, but not (2). It was determined that there were no existing XML based analysis techniques that could correctly analyze the given databases. Therefore, no common elements could be identified so a new XML based analysis process is proposed. The execution of the new analysis procedure, however, will require increased computing resources and improved database tool support. The AFATDS database has not been available for analysis, thus far.

b. XML as a Data Exchange Medium in Real-time Systems, ongoing thesis by *Kris Pradeep*

Focus:

The primary focus of this effort is to develop programs that generate random data sets according to selected subsets of the database schemas. The subsets chosen will be representative of typical interactions encountered in system interoperations. This data set will form the benchmark data for later analysis.

The thesis effort will evaluate four legacy databases schemas and will determine the subset to be used. In a normal system operation, information from these databases are searched and transmitted to other systems. Hence, the messaging formats employed for message transmission will be analyzed. The subset schemas will be further populated with the message formatting rules and parameter ranges. Finally, programs will be written to create messages from the subset containing random (valid) data for transmission.

Progress:

The main part of the work (sub-portion of Task 3) is "preparing benchmark data samples" that others in the team would use to run their tests. The main challenge was to understand the four databases of interest to JBC, and to decide what would constitute representative benchmark data. Due to the difficulties encountered in obtaining real data from the four databases, (for reasons ranging from Security issues to unavailability of data), the GCCS TDBM was chosen as a representative database. Using this database, programs were created that generated messages in both the USMTF and its parsed XML versions. These select messages have random values but are within the legal limits specified by its unique message formats. These selected subsets of the database schemas are representative of the typical interactions that might be encountered in system operation.

The following message generation in its native and parsed XML versions has been implemented:

In the CIX format:
- Basic Link Track Message Sets: LCTC, XPOS
- Extended Link Track Message Set: LEXT
- Theater Ballistic Missile Track Message Set: BMISL

In the USMTF format:
- WXOBS: chosen to highlight several features of USMTF-XML mapping modes

Timing measurements of overhead due to use of XML, such as translation delays, is proposed to complete the goals of Task 3.

c. Common Data Attributes, ongoing thesis by *Hamza A. Zobair*

Focus:

The primary focus of this effort will be to conduct an analysis of the two database components and their schemas. The results of this analysis will be used to develop a recommendation of how data elements can be employed to support scalability of the databases to support growing C4I requirements. Databases to be analyzed include JCDB and MIDBG.

This thesis effort will evaluate two similar legacy databases and will derive the common data elements that are required to support scalability of these databases

during the migration to more modern C4I systems. Based on our analysis we will recommend common XML based data elements that could support the scalability of the legacy databases. The methodology to be employed in this effort will include analyses of each database along with side-by-side comparison of the databases to identify common elements. The current and future C4I systems database requirements will be acquired from program management offices and analyzed to identify the scalability requirements of the databases. The portions of the legacy database sharing schemas that are required of the future C4I systems will be derived. Currently available XML schemas that support similar data sharing attributes will be examined to determine if there are any reusable components or approaches that could be employed in this research effort. XML schemas will be derived that support scalability of the existing data to meet future C4I requirements.

Progress:

The task undertaken by Zobair is to find common attributes among two different databases. There are three or four major approaches to conducting such an effort. The most obvious is a Boolean logic based text retrieval search engine. Others are Natural Language based search engines, Vector Space and Neural Networks. There are several tools or techniques that can be used to find better matches with the basic search types. They include term weighting, stop-words, stemming, thesauruses, etc.

Boolean logic is a process that requires the user to find a match one by one. It is a process that has not been automated in the sense of being able to input all the dictionaries at once and expect to get matched attributes at the other end. Natural language search engines take into account the frequency of words entered in a query and evaluate them against keywords that are in various data dictionaries. Some of the natural language software packages use stemming, stop-words, and thesaurus tools to assist in finding matches. Full automation of the natural language process has not been achieved either. Vector space and neural network are processes that require manipulation of the data dictionaries into vector or signature files. Once the files are converted into a vector, some linear algebra manipulation is done on the vectors to find close matches. The potential for automation of these processes is high. Leading publications on these processes include the SEMINT paper by Chris Clifton of MITRE Corp and papers on Latent Semantic Indexing or Latent Semantic Analysis. Professor Mike Berry and his colleagues also did some of the work on Latent Semantics at University of Tennessee. These two researchers claim to be very successful and to have automated the process.

Zobair has elected to use a combination of Boolean logic, natural language, and some of the features that were used in SEMINT in his efforts. He is conducting his match finding dynamically, i.e., he is selecting the matching method based on the specific attribute he is evaluating. SEMINT suggests one should not look exclusively at the semantic meaning in finding a match. SEMINT suggests we

evaluate potential correspondence based on the metadata values of the attribute. For example, SEMINT suggests one also look at data types, null values, attribute name lengths, and attribute definition length along with quite a few characteristics that do not focus on the actual meaning of the attribute. Zobair has concentrated mainly on data type, and null value in his correlation efforts. A key feature of SEMINT that made it successful was the fact they were able to automate the process that converts all the data dictionary attributes into vectors. This was not possible in Zobair's efforts in that all he received was the raw data dictionary. The process of converting each attribute into a vector would take longer than would be required to conduct a manual correlation using a natural language or boolean logic search engine.

In his analysis, Zobair divides the data attributes into clusters or basic concept areas. The process he used for accomplishing this was to first scan all the abbreviations and acronyms used within each data dictionary, adding them to a user-defined thesaurus. For example, target, tgt, trgt are all given the same meaning in his dictionary. Then he conducted a simple search on the word Target and obtained all attributes that have the word target or any of its above abbreviations. This became his target cluster. He followed the same procedure for Observation, Track and Equipment.

Zobair then uses the resultant clusters for conducting manual searches. There are many occasions where a specific search type does not yield the desired result so a number of different search types must be used to get a closer match. Search results have averaged about 1 hour to find 16 matches. Once the initial search is completed, the best matches are combined and recommended for standardization. Lesser matches are recommended for future broadening of their definitions during subsequent database upgrades in order for them to be integrated with other databases.

Sample Natural Language and Boolean Logic Query

In the example below he searches for a match to the MIDB attribute "RECUP INTRVL MAX" using a natural language search engine. The natural language query entered is the attribute definition "*When* recuperability interval is *represented* as a range, this *field* indicates the maximum *interval* of time required to repair the damage." The closest match the natural language search engine finds is "FAC_DEPTH_MAX." The definition of this attribute is "*When* depth is *represented* as a *range*, this *field* indicates the *maximum* extent, measurement, or dimension downward, backward, or inward in meters. Unit of Measure = Meters". The reason why the search engines find this match is because of the words ***maximum, range, when, represented, and field*** are common to each attribute definition. Since the other keywords were not in the second data dictionary it skipped those words and found the closest match using the words in common. As a result of the bad match he redid his search using boolean logic query with a thesaurus, and stemming on the terms *damage* and *recup*. The resulting match is

"QTY_RECUP" and the attribute definition is "The quantity of a specific *damaged* MATERIEL-ITEM in a specific MATERIEL-ITEM-FACILITY-HOLDING that is *recuperable* after being *damaged*. >>QTY_*DAMAGED* must be specified<<<. Although this not an exact match it is a lot closer than the results in a natural language query and it leads him closer to a potential match.

Attribute Finding Match for	Natural Language Match	Boolean Logic Match
-HEADER- MIDB	-HEADER- JCDB	-HEADER- JCDB
1. Element Name: RECUP_INTRVL_MAX	ATTRIBUTE NAME: FACILITY maximum depth dimension	ELEMENT NAME: MATERIEL-ITEM-FACILITY-HOLDING recuperable quantity
2. Attribute Name: RECUP INTRVL MAX	PHYSICAL NAME: FAC_DEPTH_MAX	ATTRIBUTE NAME: QTY_RECUP
3. Definition: When recuperability interval is represented as a range, this field indicates the maximum interval of time required to repair the damage.	DEFINITION: When depth is represented as a range, this field indicates the maximum extent, measurement, or dimension downward, backward, or inward in meters. Unit of Measure = Meters	DEFINITION: The quantity of a specific damaged MATERIEL-ITEM in a specific MATERIEL-ITEM-FACILITY-HOLDING that is recuperable after being damaged. >>QTY_DAMAGED must be specified<<<
4. Data Type: int, NULL	DATA TYPE: numeric(5,1)	DATA TYPE: smallint integer NOPTIONS:NULL NULL
5. Permissible Values: RUL_NUM_IN_POS	NULL OPTION	TABLES: MATERIEL-ITEM-FACILITY-HOLDING MATERIEL-ITEM-FACILITY-HOLDING-HISTORY
Positive integer greater than zero	NULL	-END-
Whole positive numbers and zero. Values range between 0 and 2,147,483,647, inclusive. Storage size is four bytes.	ATTRIBUTE ENTITY: FACILITY	
6. Tables: EQP, EQP_ASSESS, FAC, FAC_ASSESS, TGT_DTL, TGT_DTL_ASSESS, TGT_SYS_ASSESS, UNIT, UNIT_ASSESS		
-END-_		

IV. Proposed Continuing Efforts

1. **Schema Integration.**
   This is a continuation and extension of an FY00 task with the objective of assessing methods for identifying corresponding parts of existing XML schemas. In 2000 we have surveyed the literature, identified a promising method and some supporting tools applicable to this process, and have applied that method to approximately 15% of the data models for GCCS GMI and JCDB. In 2001 we propose to complete the application of the method to the case study, use the result to estimate the cost of applying the method, and to discover and assess better methods and tools for supporting this process. The initial results of the 2000 effort in this direction indicate that current approaches to discovery of parts of different schemas that represent the same real-word object are very labor-intensive, and that better methods and tools are possible by applying techniques developed to solve different problems.

2. **Database Crosswalk.**
   This is a continuation and extension of an FY00 task with the objective of evaluating whether XML can provide an effective approach to transferring data between heterogeneous databases that can have different schemas or data models. In 2000 we completed a survey of technologies and COTS tools for supporting database queries via XML, and identified a process that could be applied to the GCCS-I3 MIDB schema. An M.S. thesis addressing these issues was completed in SEP 2000. In 2001 we propose to apply this process to a case study, evaluate its feasibility, and assess associated levels of effort and cost.

3. **Real-Time Data Exchange.**
   This is a continuation and extension of an FY00 task with the objective of assessing time penalties associated with use of XML in real-time systems and methods for overcoming them. In 2000 two students completed a joint M.S. thesis that designed a data translation architecture, identified COTS tools for realizing the architecture, implemented a small application of the architecture, and ran a test case to establish its feasibility. In 2001, we propose to use this architecture and implementation to measure time overhead for data transfers typical of real-time military systems.

4. **Data Compression.**
   Military communications often depend on channels with limited bandwidth, which is subject to duress under conflict – bandwidth narrows or disappears, while volume of traffic jumps. XML has advantages for data interchange, such as lowering the cost of extending data connections between legacy systems, but when used as a transmission format for communications links, the XML tags can greatly increase data size. This task involves identification and assessment of data compression techniques that can counteract this disadvantage. The assessment

will consider general-purpose data compression methods as well as methods that can exploit the structure provided by the XML to improve data compression. For example, XML tags can be used to identify and match fields, so that only modified parts of periodic messages need be transmitted. Previous DoD experience with such methods will be assessed and uniform data compression software architectures will be studied to determine if a systematic low-cost solution is possible.

5. **Translator Maintenance.**

Explicit use of a single form of XML for physical message transfer is attractive from the point of view of software development and maintenance costs, because a relatively small number of different translators is required, but it has disadvantages from the perspective of time delay and bandwidth requirements. Optimizations that overcome these limitations use different kinds of representations for different links, and require a larger number of translators to achieve better real-time and bandwidth performance. We propose to assess the degree to which the software cost disadvantage of these optimizations can be overcome by technologies for automatically generating and maintaining families of translators based on XML descriptions of the data representations at both ends of an optimized link. This includes: (1) assessment of the feasibility of generating a large number of different translators from a relatively small number of XML format definitions; (2) evaluation of the advantages and disadvantages of alternative software architectures for providing such service; and (3) the relative costs of automated assistance for creating and reconfiguring the network of translators versus conventional manual translator implementation approaches.

V. Appendices

a. MS Thesis by Dave Hina

b. PhD Thesis Proposal by Paul Young

c. MS Thesis by Brian Lyttle and Todd Ehrhardt

d. MS Thesis by Tam Tran and James Allen

e. MS Thesis proposal by *Robert F. Halle*

f. MS Thesis Proposal by *Kris Pradeep*

g. MS Thesis by *Hamza A. Zobair*

g.1. Thesis Proposal
g.2. Thesis Draft

ABSTRACT

This thesis evaluates the application of current Extensible Markup Language (XML) tools and technologies toward solving data interoperability issues between legacy data repositories. Past efforts to address these issues have largely failed. XML has the capability to address many of the past problems, but this can only be accomplished when the supporting COTS tools and technologies are available.

The thesis first establishes the underlying issues that need to be addressed. It then evaluates the current state of technologies and COTS products and describes the advantages and disadvantages of each. Finally, it focuses in on the schema for a specific relational database, demonstrates a process by which data exchange can be implemented, and outlines the issues remaining to be solved.

A. SUMMARY

The Joint Battlespace Infosphere (JBI) [JBM00] has been introduced as a concept that will allow the wide sharing of data between heterogeneous systems across multiple domains. The producers of the data, which under this concept remain largely unmodified, are the legacy Department of Defense (DOD) systems upon which battlespace operations depend, currently and for the foreseeable future. The ultimate consumers of the data include legacy applications, web access, commercial-off-the-shelf (COTS), and other mission-specific applications. The middleware layer that makes this level of data sharing possible is based on the eXtensible Markup Language (XML) and its associated specifications and technologies. It is the goal of this thesis to describe the software architectures and available COTS technologies that can be applied to bring this concept closer to reality.

There is currently a large amount of data within the DOD that is restricted to being utilized within either a single system or by a specific group or entity. This restriction is primarily caused by the differences between various software systems and databases. While mechanisms do exist for translating data between the different database systems, these mechanisms are typically restricted to a single application and are easily affected by any changes

that occur. Additionally, due to their custom nature, they are able to make little use of COTS tools for reducing the cost involved in development and upkeep.

XML is a recent technology that is ideally suited for taking data from diverse representations, and reconciling it into a common format that is portable and has a simple interface for data retrieval. It also has the advantage of being an open standard for which COTS products are constantly being developed and improved. A close look needs to be taken at how XML is currently being used to improve data interoperability, and at the future approach that should be taken, given the current state of the technology and existing tools.

B. RESEARCH QUESTIONS

This thesis will answer the following questions:

1. What are the issues that complicate data exchange between the systems of interest?
2. In what ways can XML and its related technologies offer solutions to these issues, and where are the deficiencies?
3. What are the XML specifications, technologies, and products that are applicable for database to database exchange of data?
4. What is the current state of these products and technologies?
5. What process can be followed to successfully apply these products and technologies?

The answers to a number of these questions will be presented within the context of the Global Command and Control System (GCCS) Integrated, Imagery and Intelligence (I^3). This system is representative of the legacy environments where data sharing is becoming a necessity. It will be used to illustrate the concepts laid out in the thesis and to demonstrate some of the issues that need to be addressed.

C. MOTIVATION

Traditionally it has been a difficult, time consuming, and expensive task to share similar data between different database systems. This has been largely due to the fact that there is no single standardized format for data transport between these heterogeneous systems, and no set of COTS tools to provide cost effective support. The primary benefits of this thesis will be to provide an analysis of where and how the use of XML, in its current state, can be applied to improving this level of data interoperability in a cost effective and timely manner.

D. ORGANIZATION

This thesis is organized into the following chapters:

- Chapter II provides background for the thesis and summarizes existing literature that is applicable to both data interoperability within the DOD and to the use of XML for data sharing purposes.

- Chapter III identifies the process by which XML can be used for structured data transport. This includes the challenges and considerations that need to be addressed in the use of XML.

- Chapter IV presents an analysis of relevant technologies and a look at the types of COTS tools that presently exist to support these technologies. Specific examples of COTS tools illustrate each of the tool categories.

- Chapter V presents and evaluates a process for applying XML technologies and tools to the GCCS-I3 database segments. This is done in the context of both Sybase Enterprise Server, the primary Database Management System for GCCS-I3, and middleware data translation and mapping tools.

- Chapter VI provides thesis conclusions and recommendations.

II. BACKGROUND

A. DOD DATA INTEROPERABILITY

1. Understanding the Issues

There are many barriers to data interoperability within the DOD and they have been well documented [NRC99]. When most information systems are first developed, they address a single, very specific set of requirements. The data formats are typically chosen to best suit the mission at hand, with little regard to standardization with other existing systems. When, after a few years, the need arises to communicate with another system, pairwise interfaces are developed between the two systems to support this need. This gives birth to a tangled web of directly interconnected systems that become increasingly difficult to maintain.

a) StovePipe systems

The problems associated with sharing data between systems are inherent in the original design of most legacy systems. When systems are designed and developed to operate in isolation, there is no motivation to consider the need to share data with other systems. Over time, this has changed as users have demanded access to multiple data sources from within a single application. With the advent of the Internet and with the greater emphasis placed on communication, the

advantages to shared data and to the use of open data formats are being realized, and the result is a change in the way systems are designed.

Unfortunately, within the DOD legacy data systems cannot just be redesigned from the ground up since they are critical to daily operations. A phased approach is required to continue the use of these systems while making the move to a shared data environment. Migrating legacy systems, however, is not an easy task. In evaluating the set of problems that must be addressed, Renner [REN96] focuses on the move to a shared schema, the extraction of knowledge encapsulated within the legacy applications, and the risk involved. He states that a shared schema is necessary, but it is difficult to achieve because it requires reverse engineering the schemas of the individual applications. He also points out that the existing applications contain valuable knowledge that cannot be easily discarded. Another area of concern is that a simultaneous cutover of all applications cannot be required, since the risk of failure with the operational systems is too high.

Due to the problems associated with moving directly to a shared schema, many legacy systems have taken the route of creating applications dedicated to providing the interface between each pair of directly connected systems. These applications contain the necessary knowledge to convert between the different schemas. As Renner and

Scarano [REN96] note, this method of communication is expensive because of the development and maintenance involved and because, with the pairwise interfaces involved, the cost increases with the square of the number of systems involved.

Another common approach to getting single use, stovepipe systems to play in this new world of shared data, without the move to a fully shared schema, has been a layered approach. In this approach data from different legacy data stores is maintained separately and with different schemas. Applications are modified to interpret data from each of the systems and provide displays on the same screen, but data from each individual system is layered one on top of the other. From the end user's aspect, these systems can be frustrating and difficult to use because of the lack of synchronization between the layers. They also lack scalability, since there is no true integration at the data level between the different layers.

One other approach that is more appropriate under some circumstances, and the approach discussed in this thesis, is to create a shared data server that accepts appropriate subsets of data in a central schema format from the external legacy systems. [REN96] states that this approach to integration involves three tasks: ``developing a data model and data elements for the shared data, converting the legacy data values to this new

representation, and modifying the application programs to use the shared data server and its schema."

b) Diverse Data Representations

While many of the large data stores within the DOD are handled by very capable DBMSs, such as Oracle, Sybase, Informix and others, there are a number of issues that hinder interoperability between these systems.

One issue is the lack of a common vocabulary for use between the systems and the lack of a framework to support such a concept. While the need for a common vocabulary has been recognized for years, attempts to make it a reality have been met with very limited success. This has been largely due to the fact that there is often little motivation for data providers to spend the time and effort that it takes to integrate their individual data definitions into a central repository. Other tactics, such as forcing data providers into modifying existing systems to use a common set of terminology, also have not produced results.

One issue that hinders the move to a centralized vocabulary, as Hodges and Buck [HOD00] point out, is that the only way to truly understand the semantics of the data is to analyze both the structure of the data and the legacy applications that use it. This can be a costly process and very difficult to perform.

Another problem is the different types of data stores that exist in some of the legacy systems. While the majority of the data within the DOD exists within relational data stores, other formats that must be handled include both flat file and hierarchical structures. These data stores often do not have a standard mechanism for accessing the data, and may require complex, single use application programming interfaces (API) to extract, update, and delete data.

c) Time and Cost of Change

The cost of migrating these systems to exist in a shared data environment, for the reasons listed above, can be very high. History has shown that the timeframe required for the migration can also be unacceptably long.

[ROS00] points out that a large part of the cost historically has been faulty assumptions that are made in the approach taken to solve the integration problems. Some of these assumptions include a focus on the end task instead of an incremental, phased approach, insisting that all participants in the integration process adapt the same standard data models and data definitions, and that mandates are sufficient for getting all participants to contribute meaningful metadata about their systems. They also express concern that top-level infrastructure spending can be wasted, because it provides no motivation for the individual

systems to assist with the interfaces necessary for data integration.

2. Interoperability Strategies

a) Shared Information

Rosenthal et al. [ROS00] state that the goal of data integration is often portrayed as ``all data available to anyone, any way they choose, anywhere, and at any time.'' The proponents of such a broad view typically include in their vision the following:

- Combinations of legacy systems, new systems, and data
- A universal centralized schema
- Metadata describing the individual systems
- Intelligent middleware to connect requesting applications with specific data sources.

This is followed by the observation that broad visions and goals such as these usually lead to failure.

Instead, they propose a more realistic view, with the recognition that constant change is inevitable, sources and consumers of data will be varied and unpredictable, and access to data is not sufficient, but true integration at the data level is required. This can only be achieved by treating data integration as a continuous process, and building in support for continuous improvements.

b) Metadata Tagging

Rosenthal, Sciore, and Renner [ROS97], in their paper ``Toward a Unified Metadata for the Department of Defense,'' state that ``data sharing within and among large organizations is possible only if adequate metadata is captured.'' The paper goes on to discuss the metadata requirements and approaches for metadata collection for a data sharing infrastructure. They state that the most difficult task is the collection of the metadata, and that this should be a collaborative effort between individual system builders, who possess the system knowledge, and the organization responsible for the overall interoperability effort.

c) Common Vocabularies

Another area that is a requirement for true information sharing is that there exist a framework for defining a common language between the different data sources. There are a number of approaches that have been taken to provide standardization, both in and outside of the DOD community. Some of the primary approaches will be described here.

ISO 11179 [ISO97] is a draft standard to establish such standardization in the form of a metadata registry on an international scale. The basis for this standard is to define data element classification schemes, and to use these

schemes to build and populate classification structures. In addition to classification, the standard specifies basic attributes that data elements should possess, rules and guidelines for data definitions, and specific naming and identification conventions.

While each of these are important considerations for a centralized registry, Rosenthal et al. [ROS97] point out some of the shortcomings in this approach, with the primary problem being the lack of specific features, such as actual schema specifications and APIs. Without these, the support of vendors and developers required for widespread adoption is less likely.

A DOD standard for data modeling, DOD 8320.1, was established in 1991 with the goal of collecting data element definitions across the DOD. It outlines a set of procedures for data element standardization, including the Defense Data Repository System (DDRS), which is a central database that includes data standards in terms of standard entities, data elements, and data models.

These past efforts to create centralized catalogs of metadata for the entire DOD or on an international scale have largely failed. This has occurred partly because the individual system developers have not had a real incentive to either pull from or contribute to these central repositories and partly due to maintenance issues with the metadata. More recently, the Shared Data Engineering

(SHADE) Team introduced a smaller catalog effort as part of the Defense Information Infrastructure (DII) Common Operating Environment (COE). The Joint Common Catalog (JCC) is not aimed at becoming a central catalog to handle all DOD metadata, but rather, as described in [HAS00], it will be ``a set of components that can be used to create local catalogs as required that can interoperate because of their common features.'' One important part of the JCC is an XML Namespace Registry, which is an XML representation of the metadata elements maintained by the JCC. As a result of features in the XML specification, the registry will be able to provide runtime access to the metadata repository.

3. Joint Architecture Approaches

a) Joint Common Database

The Joint Common Database (JCDB) [CAR00] brings together in a single data architecture many of the components that this thesis will promote as being central to developing a true shared data environment. It is being developed as a single data repository that integrates data from multiple disparate sources, to generate a Common Tactical Picture.

The JCDB will combine data from multiple external inputs to develop its common data store, and then use a distributed approach to provide a reliable common data source for applications needing it. Elements that are

central to the JCDB's interoperability data model include a Joint Data Dictionary, a set of translators from legacy data stores, and the use of a standardized, semi-structured, data transport format in XML. Hayes et al. [HAS00] expand upon the need for a central data vocabulary as part of this effort and discuss the current state of the JCC. The usefulness of data translation layers and semi-structured data formats is discussed later in this chapter.

b) Garlic Fries

One of the areas where there has traditionally been a lack of interoperability has been between dynamic, near real time (NRT) data sets and less volatile relational data sets. Specifically, within the GCCS-I3 Common Operational Picture (COP) environment, this problem exists between the relational data stores such as the Modernized Integrated Database (MIDB) and the NRT data maintained by the Tactical Management Service (TMS). Track data managed by TMS, which is usually very time sensitive, lacks any consistent method for correlation with similar data within the relational data stores. When new tracks are established within TMS, due to the differences in data representation and the dynamic nature of some of the track parameters, new track IDs are established even when the object is already represented by an existing track ID within the MIDB. One

result of this synchronization problem has been the lack of ability to maintain a longer term history of tracks in TMS.

Garlic Fries is a system currently being developed to address both the synchronization and archiving problems. Synchronization between the data stores is provided by a set of translation filters, correlation logic for associating the NRT tracks and the more static relational data, and a new data store to handle cross-references between the two. Once the data from both data stores is in a common format, the correlation logic will utilize timestamps, positional information, and other attributes to correlate the track representations. Archiving of the TMS tracks will be handled using XML for data transmission from TMS and the relational data stores, and for use by other mission applications. This will allow the client applications to make use of an extensible, standardized interface that is abstracted from the underlying data structures. [FGM98]

This is an example of how XML is finding its way into a DOD data interchange environment. It also points out one of the advantages of using a semi-structured data format such as XML, which is that messages in a single format can function as both a medium for structured data transmission and for formatted data display.

B. USING XML FOR DATA INTERCHANGE

1. Why XML?

In it's most basic sense, XML is a method for transmitting and storing structured information. It allows us to apply meaning to information, in such a way that the information can then be indexed, searched, displayed, and manipulated with greater ease than it might otherwise have been.

The XML Specification [XML98] states that ``XML describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them.'' In the annotated version of the specification, Tim Bray further states that an XML document can be represented in a number of different ways, including as a file, a record in a relational database, an object delivered by an object brokering system, or as a stream of bytes at a network socket [BRA98].

2. Characteristics

XML is a mechanism for describing content, and as such, it has a set of characteristics by which it provides this data description. Some of these characteristics are important to consider when comparing XML to other mechanisms for structured data storage, such as relational and object databases.

XML, as it's name implies, is extensible. One of it's greatest strengths is that it allows a language to be created that is specialized for a particular area of use. Within this area of use, a document type definition (DTD) can be defined that specifies the set of tags to be utilized for documents pertaining to this area. The DTD, in addition to defining a grammar, can also, to a limited extent, function as a schema for the document. Specifically, the XML specification does give the DTD the ability to express which data elements must be present, what attributes they must have, specific ordering of the data elements, null data constraints, and limited functionality for expressing data element uniqueness. The DTD lacks the capability, however, to constrain the data type of elements, to express the allowable size of the data within an element, and to limit the allowable set of values an element can possess.

The structure of an XML document is hierarchical, and can always be described by a tree-like graph. This hierarchical structure is good for providing a clean organization of the data and for easy translation into other environments that require structured data. It also lends itself to other functions that commonly need to be performed on data, such as querying, searching and indexing.

Another characteristic of XML includes a method, through use of the XML *ID/IDREF* attributes, for associating

unique identifiers with each element, the basic unit of an XML document. This becomes important when translating data from an XML document to a structure that might require unique identifiers, such as a relational database.

To illustrate this concept a simple example follows. It consists of a list of two tracks, each of which has a globally unique track ID, a timestamp, and a location identifier. In a relational database, each track would be expressed as a single row in the TRACK table, with track_id representing the unique primary key. The DTD syntax for the TRACK element could be expressed as

```
<!ELEMENT Track (#PCDATA)>
<!ATTLIST Track track_id ID #REQUIRED
timestamp CDATA #REQUIRED
coordinates CDATA #REQUIRED>.
```

A corresponding XML document might be

```
<Track_list>
  <Track track_id=''1001''
         timestamp=''17203055''
         coordinates=''325377680N1171033970E'' />
  <Track track_id=''1002''
         timestamp=''17253000''
         coordinates=''325325440N117102325E'' />
</Track_list>.
```

In this example the use of the *ID* attribute guarantees that if the value for the track_id attribute is not unique, validation of the XML document will fail.

One characteristic of XML that is clearly an advantage over relational systems is the ease with which changes can be made to the data structure. Because structural information is maintained as part of the data itself, these changes can also be easily implemented independently of the source system. Using the previous example to demonstrate this characteristic, imagine that a system that processes the track data adds a requirement that in some cases the *coordinates* field needs to be expressed as two separate fields, *latitude* and *longitude*. This can easily be accomplished by adding a transformation filter at the point at which the data is received, so that when the necessary conditions exist, the resulting XML would become

```
<Track_list>
  <Track track_id=''1001''
         timestamp=''17203055''
         latitude=''325377680N''
         longitude=''1171033970E'' />
  <Track track_id=''1002''
         timestamp=''17253000''
         latitude=''325325440N''
         longitude=''117102325E'' />
</Track_list>.
```

This requires no changes to be made to the source schema, and the structure of the resulting data can easily be understood so that it can be processed in the appropriate manner. An equivalent change to a relational schema might require expensive and risky changes to be made to application code in many different locations, and the change to the structure of the resulting data can not easily be recognized by analyzing the data itself. This is an important difference between XML and relational systems.

3. Design Principles

One of the main reasons that HTML has been so successful as a display technology has been its simplicity. XML has been designed to provide the same level of simplicity, with greatly expanded functionality. The XML syntax is easily read and understood by both humans and machines. The options that the language presents have been kept to a minimum, making it much more unambiguous to work with from a developer's standpoint.

XML was designed to have the same features as HTML, but with added functionality and without some of the problems that HTML has experienced. In the same way that HTML has been successful in its ease of use over the Internet, XML was designed specifically for use in a widely distributed context.

One of the problems that HTML has had, however, has been the error tolerance required for applications that parse HTML documents. One of the most important aspects of the XML specification is the formality and precision with which it requires that conforming DTDs and XML document instances be written. As Bray writes, ``Too many other standards and specifications have relied too heavily on prose and not enough on formalisms.'' [BRA98]

The XML specification introduces the concept of a *well-formed* document. This is an XML document that can always be unambiguously parsed to create a logical tree in memory, meaning that any parser should create the same tree structure. This allows a great deal of reduction in the amount of error handling that must exist in applications that process XML documents - they are either ▒well-formed▒ or they are not, in which case they do not get processed. A large percentage of the code in today's web browsers is there just for error handling. This adds complexity and variability in behavior, thus the formal and concise design requirement for XML.

XML was also designed to be directly compatible with SGML. This meant that XML documents should conform to not only the XML specification, but also to the SGML reference [ISO86]. This goal was added to leverage the existing set of SGML parsers and applications. Therefore, any XML

document should be able to be processed without error by existing SGML applications.

4. Where does XML fall short?

a) XML is not a database management system

XML is a text markup system, and it was not designed specifically for database management. XML does not possess some database-like features in the same way that DBMSs do not possess markup-like ones. A typical database management system possesses not only the ability to store structured data, but also methods for querying, viewing, optimizing, and processing the data in ways that the data can be easily and efficiently utilized in many diverse applications. XML, in and of itself, does not possess these capabilities, although a number of additions and extensions have been made to the original specification that make this capability more of a possibility.

b) Not always the most efficient solution

One of the original design goals from the XML specification stated that ``terseness in XML markup is of minimal importance.'' This underlines the fact that efficiency was not a top priority in the design of XML. The emphasis was instead on clarity, simplicity, and wide area of application. This is one of the reasons that XML, although it is a method for storing structured data, will

not by itself replace all the functionality of a relational database.

5. XML and Structured Data

In addition to its use for describing and storing text data, XML's primary purpose is to transmit structured data [ABI00]. The origin of this purpose is largely a result of the need to provide a mechanism for moving structured data over the Web, which is composed of a wide variety of different types of data sources.

One of the stated design principles behind XML was that it must support a wide variety of applications. More specifically, it was intended to be a vehicle for exchanging data between heterogeneous systems, and as such it can represent data from a wide range of origins in a common format [BOS99]. This is accomplished in XML largely by the way that the structure of the data is described by a formalized, standard mechanism, and this data description is always either maintained as part of the data itself or in a directly referenced description document.

The most important structural characteristic of an XML document is that it is hierarchical - the data is represented in a hierarchy of nested structures. The order of the elements within this hierarchy is important and must match the order outlined in the DTD, if one exists. Additionally, XML does provide for element identifiers that

are unique throughout an entire document and across all element types.

One of the unique features of XML data is its capability to retains its structure and its meaning despite multiple transformations. When information is pulled out of a database for a specific purpose, such as for display or to be stored in an alternate format for later use, it often loses both its structure in relation to other data and its meaning in other contexts. Data from an XML document, since it contains the description of the data as part of the data itself, can maintain much or all of its meaning, despite having undergone one or more transformations or transportation to a different context.

6. XML Messaging Solutions

Standardized messaging has been used for communication within the DOD for many years. The data formats used in this messaging have been implemented in different ways, but their primary purpose remains the same - to provide a mechanism for data interoperability. Examples of these standards include USMTF, TADIL, and CIX.

The success of these messaging standards at providing interoperability is a subject of debate. One of the main reasons cited for their limited success has been the expense involved in maintaining the standards in the face of changing priorities and advances in technology. The

architectures of the systems on which they are based have been described as being inflexible, because they require messaging formats to be known in advance. Any additions and changes that need to be made to the message formats typically involve long, costly trips through standardization committees and development cycles [ROS97].

These messaging standards do, however, have aspects that make them necessary, and even attractive, for continued use, now and in the near-term future. They are widely used for military information exchange both within the DOD and with partner nations around the world, and they are based on years of experience with the collection of information exchange requirements. This infrastructure can, therefore, be immediately utilized as both a vehicle for communication and for information collection, reducing the expense and time required to achieve true data interoperability.

The application of the XML format to these messaging solutions is one approach to making the move from inflexible legacy systems to achieving flexibility via the use of XML. Mapping existing messaging systems directly to XML can be done by developing a set of XML tags that correspond to the data fields within an existing message format, and then using either a DTD or XML-Schema document to describe the constraints implied by the legacy system. Messaging systems usually consist of a fairly simple, hierarchical structure, that maps cleanly into corresponding XML structures. One

advantage of this approach is the continued use of existing extraction and input interfaces to the database, while achieving the desired resulting data format.

This has led to efforts that leverage existing messaging infrastructures within the context of advantages provided by XML. The following section discusses two such efforts.

a) XML-MTF

The US Message Text Format (USMTF), which is widely used by the US and it's allies, has hundreds of classes of strongly typed message formats and thousands of standard data element definitions. The MTF system includes many specialized tools and technologies for the processing of the hierarchical MTF messages, including validating parsers, document creation and editing systems, a query language, and processing and delivery systems.

XML-MTF was developed as an initiative for the continued use of MTF by its large community of users, while reducing the cost and effort required for its maintenance. The expectation is that the use of COTS tools for processing XML and the standardization features of XML will ease the process of extending MTF and make it more interoperable across systems.

Much of the current effort to utilize XML for improving interoperability within the DOD has been focused

upon the use of XML-MTF. Schneider [SCH00] describes ongoing XML-MTF efforts as part of the Joint Battle Management Integration (JBMI) Assessment, to which this thesis is contributing, as having the following attributes:

- Modernizes military information standards through commercial technologies
- Capitalizes on 20+ year investment in military information requirements
- Leverages industry standard XML format
- Defines a standard XML mapping for MTF messages
- Provides simple software tools to support XML-MTF implementation. [SCH00]

It is important to note that while these efforts are critical to establishing the use of XML for interoperability and for leveraging existing channels of communication, this is just the first step in establishing true interoperability. The use of existing message formats brings with it many of the problems that have limited the success of messaging in the past. The systems that utilize XML-MTF will still remain largely inflexible and limited to specific subsets of data. Overcoming these issues will probably require the use of centralized registries and true database-to-database interaction, such as discussed later in this thesis.

Broadcast mode for disseminating track data. Besides limitations common to other messaging systems, such as inflexibility and maintenance issues, the use of GCCS-COP in conjunction with non-GCCS parties has been limited by the message format, Over The Horizon-Gold (OTH-G), which lacks support outside of the GCCS-COP user community. [INR00]

This has led to the update of CIX software to handle messaging in XML format. This is discussed and expanded upon in Chapter V.

C. SEMI-STRUCTURED DATA MODELS

Semistructured data is often explained as ''schemaless'' or ''self-describing,'' terms that indicate that there is no separate description of the type or structure of data. This type of data contains the description of the data as part of the data itself, unlike highly structured data representations, such as most relational databases. Such data is much more portable and free from many of the constraints that are typically associated with database representations, but it can also be more difficult to represent more complex relationships between the data in a semistructured representation.

more difficult to represent more complex relationships between the data in a semistructured representation.

In a more complete description, Florescu and Kossmann [FLO99] describe semi-structured data as having the following characteristics:

- The schema is not given in advance and may be implicit in the data,
- The schema is relatively large with respect to the size of data and may be changing frequently,
- The schema is descriptive rather than prescriptive (i.e. it describes the current state of the data, but violations of the schema are still tolerated)
- The data is not strongly typed (i.e. for different objects, the values of the same attribute may be of differing types). [FLO99]

One common example of semistructured data is a file system hierarchy, which is typically represented in such a way that meta-information in the data itself is used to describe each of the data structures. This is very analogous to the hierarchical structure of XML documents and the self-describing nature of XML elements.

XML is a form of semistructured data. It exists in a hierarchical structure with the markup within a document representing the data description. The XML Document Type defines the class of document, and this document type is defined in a Document Type Definition (DTD), which specifies the structure of the document in terms of the attributes and

elements that it is made up of and the order and relationships between them. One important difference between the DTD and a relational database schema, is that in practice, the DTD is used purely for validation purposes, and the actual structure of the data is maintained as part of the data itself. The XML specification even allows valid XML documents that do not have an associated DTD. This is in contrast to the data in a relational database, which cannot maintain meaningful structure without the externally applied schema.

D. CHAPTER SUMMARY

It has been the purpose of this chapter to outline the issues surrounding the data interoperability problem, and to describe how XML can address these issues. One of the primary problems in the past has been an inability to adapt to change. This is one of XML's greatest strengths and one of the things that make it a good fit. It is important to understand the issues as they currently exist as well as the capabilities and deficiencies of XML, since this is the context on which solutions proposed later in this document are based.

III. ACHIEVING INTEROPERABLE STRUCTURED DATA TRANSPORT VIA XML

A. THE ROLE OF XML IN DATA TRANSPORT

The flexibility and standards based nature of XML make it a good fit for solving many of the problems that currently surround the exchange of data between disparate databases. The original XML specification, however, is deficient in meeting some of the requirements for directly translating the traditional relational data model into XML structures. Many of these deficiencies are being addressed by the application of XML Schemas instead of the DTD mechanisms outlined in the original XML specification. There is, however, a need to take a snapshot of the state of existing technology to determine what is currently possible.

One of the questions that remains to be answered is how many of the current standards and specifications that surround XML can be applied directly to solving DOD data interoperability issues. This question is made more difficult by the fact that many of the technologies are still evolving and many of the tools that support the technologies are chasing a moving target. Therefore, some amount of risk is involved with any large investment of resources in many of these areas.

B. DATA STRUCTURE MAPPINGS

Techniques for mapping between relational database structures and XML have been extensively discussed in the literature [BOU99], [BUC00], [FLO99]. Although the actual methods for implementing the mapping vary, there are a number of similarities in each of the approaches. Most current approaches make a correlation between database table and column structures and XML elements, subelements and attributes. They then make their own extensions to the DTD to handle additional information that is not handled by the XML specification.

There are two primary approaches that are usually taken for creating a relational database to XML mapping. In the first approach, template driven mapping, the structure of the desired XML document is first laid out in a template, which is just a well-formed XML document with the exception that it contains a set of processing instructions that exist within special tags. The processing instructions typically consist of SQL statements which are replaced by query results when the document is processed by the data transfer middleware.

This type of mapping can be very flexible, since the resulting XML document can be formatted as desired prior to any processing. In this approach, the actual mapping between XML elements and database structures does not need

to be predefined. The mapping is done dynamically, based on the processing instructions embedded in the XML tags.

The primary limitation of this type of mapping is the capability of the processing instructions that are included in the template. If the instructions are straight SQL, they inherit the limitations that come with SQL. This limitation can be minimized, however, by providing support for programming constructs such as looping and conditional execution, and by allowing result sets to be input as parameters to follow-on instructions. This can, however, increase the complexity of the template. In some situations, the embedded instructions could also pose a security risk if proper safeguards are not maintained on the templates. Another consideration is that this approach is really only suitable for one way transfer of data from a relational database to XML. Another approach needs to be used for moving data in the other direction.

In the second approach, model-driven mapping, the mapping is clearly defined up front. A data model of some type is utilized to describe the structure for the XML document, and a mapping is then defined between the XML elements and attributes and the database structures. In one possible data model for this type of mapping, a tree is utilized to describe the relationships between the data members, where each inner-node represents an XML element or attribute, and each leaf-node represents the non-element

data values. This tree represents the resulting XML document, and a separate mapping is used to establish the correspondence between the nodes of the tree and individual database structures.

By using a specific uniform procedure for performing the mapping, a DTD can be easily and automatically generated from a relational schema. A typical procedure would be as follows:

1. For each table in the schema, use an XML element.
2. For each column in each table, create an attribute or a child element that is restricted to containing data only (no child elements).
3. Designate a set of fixed attributes that will contain the Meta-Data. These will preserve specific data constraints, including data type, size, precision, and primary key/foreign key relationships.
4. Develop a specific set of appropriate values for data type and size.
5. Utilize some method, such as XML's ID attribute, which can uniquely identify elements, to specify primary and foreign key relationships.

The distinction over whether to use attributes or child elements for displaying data-only entities is not clear and has been a subject of much discussion. In general, entities that might be considered as properties of the parent element are often expressed as attributes, with other entities expressed as elements. As an example of each representation, consider a *Track* element from the *Track_list*

illustrated in Chapter II. Representing the entities as attributes would result in

```
<Track track_id=''1001''
     timestamp=''17203055''
     coordinates=''325377680N1171033970E'' />.
```

Alternatively, if the entities were expressed as individual elements, the result would be

```
<Track>
  <track_id>1001</track_id>
  <timestamp>17203055</ timestamp>
  <coordinates>325377680N1171033970E</coordinates>
</Track>.
```

The primary advantage of the first representation is that the DTD allows greater control over restricting the value of each entity. Extensions to the DTD are required in order to constrain the values contained within data-only elements. In general, the second representation offers greater flexibility in terms of extension and reuse, since elements allow hierarchical structure and repetition that is not possible with attribute representations.

When generating XML from relational data, the question of whether to represent the transition XML data in a deeply nested hierarchical structure, or whether to retain the relational structure within the XML document is important

and there exist supporters on both sides. Liam Quin [QUI00] bluntly states that a hierarchical format in which relational semantics are removed ``is not at all suitable for data archiving or for data transfer.'' The primary reason he cites for this is that once the data is placed in this format, it is no longer in normal form, resulting in data that is duplicated and difficult to maintain. This type of data representation requires the use of references to maintain relationships within the data wherever foreign key relationships exist.

A contrasting view comes from Rosenthal, Sciore, and Renner [ROS97], where they point out that the use of messaging in non-normal form is well-established within the DOD, the Electronic Data Interchange (EDI) community, and others. It is their belief that if this type of transfer structure were eliminated, ``the impact on existing operations and legacy systems would be too traumatic.'' They do however state that current messaging solutions are very expensive to maintain and they inhibit system flexibility.

The ultimate answer to this question is largely dependent in this context on the consumer aspect of the XML. If the data is destined for a central repository with multiple separate sources, such as in a hub-and-spoke architecture, the differences in the source schemas may drive use of the use of a true hierarchical model. The real

advantage here is that the relationship between the data is expressed as part of the data itself instead of being externally expressed within the business rules. If, however, the data is being transitioned directly between two databases possessing similar schemas, maintaining the relational structure may be more appropriate.

C. GENERAL CONSIDERATIONS

1. Data Transformation Model

One of the design decisions that needs to be made up front is the number and type of transformations that will need to be made on the data. This will be dependent upon the number of different schemas involved, the similarities between the schemas, the event model that is used, and a number of other factors. This decision will impact not only the mechanism used to implement the transfer, but also the flexibility and the complexity of the solution.

One model might consist of direct database-to-database mappings. In this case, the number of transformations required is $x(x-1)/2$, where x is the number of databases involved. The data, as it exists in the intermediate XML format, does not need to be transformed multiple times into different XML representations since it is targeted for a single destination. Advantages of this model include simplicity and reduced chances of losing the structural and relational meaning of the data.

A more flexible model involves a single extraction of data, which can then be distributed to multiple different database systems. This implies the use of a single central XML format, which is then transformed one or more times into formats required for any of the target databases. The addition of each database in this scenario adds two transformations that must occur, resulting in a total of 2x transformations, where *x* represents the number of databases involved.

The first model is less flexible and extendable since it relies on a special XML format for each source to target database pair. The second model, however, requires just one database schema to XML mapping per database, meaning that additional databases can be added more easily and changes can be made to existing ones more readily. The ability to utilize the second model, however, is greatly dependent upon the variability in the schemas of the different databases.

The second model might make use of Extensible Style Language Transformations (XSLT), which is a standard that was created for mapping XML document structure into either an HTML document or into another XML document.

2. Business Rules

One of the more difficult problems that must be addressed is handling the differences in business rules across the data repositories. The term *business rules* in

this sense represents the practices and policies of the organization which are embedded in both database schemas and in the applications that manipulate the organizations data. This problem of understanding and dealing with differences in business rules is not specific to the use of XML and is certainly not new for the DOD. The initial, and possibly more difficult issue, is extracting the existing business rules. These can be difficult to define since they are typically buried within application code and are often very poorly documented. As discussed in Chapter II, costly and time expensive reverse engineering is usually the only way to accomplish this task.

3. XML Schema Format

A number of different formats have been proposed for representing schemas in XML format. There are a number of considerations that must go into this decision, especially when considering a schema that will involve sharing across many different domains with diverse needs in terms of data representations. This is discussed in greater detail in later chapters.

4. Message Flow

Following the extraction of data in XML format, the requirements for the type of message flow between the data stores must be addressed. This can be very important when considering the challenges involved in transporting data

between typical DOD data storage facilities which may be in various geographical locations, with restricted bandwidth and unreliable connectivity between them.

5. Event Model

Another important consideration is the type of event model that will effect the data transfers. At one end of the spectrum is a periodic dump of the entire set of data. This could be based on a simple timer and maintained completely independently of changes occurring at the source database. The big advantage here is simplicity; there is no need to tie into the event model of the DBMS itself, and the queries made against the source database will not dynamically change. The potential disadvantage is delay time, and possibly an increase in message traffic, depending on the period of the database dumps and the rate at which data in the database is updated.

At the other end of the spectrum would be an update-driven approach that would trigger a data transfer whenever an update occurs to any field of interest. This approach implies the use of some mechanism within the DBMS, such as a database trigger, that is activated on updates.

6. Loss of Metadata

When moving data either from a database to XML format or from XML to a database, there are a few important aspects that must be taken into account. One of these is the loss

of metadata, or data description, that can take place. When storing data in a database, some of the information pertaining to the physical structure of the data can be lost. This includes the entity definition and usage and encoding information for the data. One example of this loss can occur with the use of identifiers which establish relationships between the tables of a relational database. Since XML can represent relationships hierarchically, these identifiers might be discarded when extracting data as XML. As this example points out, moving data from the database to an XML stream, and then back to a database will often result in a change in the resulting data structure or content - even if the relationship between the data is acceptably maintained.

It is possible to keep all of the metadata intact, with the potential loss of some flexibility and an increase in complexity. For the requirements of this analysis, since the data will be moving in only one direction, from source database to the target database, it will be acceptable to allow some loss in relational and structural integrity of the data. A determination must be made and clearly delineated, however, as to what loss will be acceptable while still maintaining the necessary meaning of the data within the target environment. Abstract data types and object modeling should be able to contribute to a solution of this issue. Through these techniques, information

attributes are relevant if and only if they are observable via a public method.

7. Data Types

The XML specification does not provide direct support for data types. In particular, it does not enforce type constraints automatically, although this can be achieved by following conventions that encode the constraints and by adding external software to check the conventions. Part of the reason for this missing capability is that XML was designed to address a wide variety of applications, with very few constraints and minimal options in the specification. This is in contrast to the standard relational database model, where all data is strongly typed. In XML, with the exception of unparsed entities, all the data in an XML document is considered text.

There have been numerous techniques proposed for typing data within XML and a variety of different implementations exist. This is still an area that is not fully developed and the subject of much research activity [ABI00].

The basic decision that needs to be made is whether to extend the DTD or to use one of the existing XML Schema formats. An example of extensions to the DTD can be seen in Appendix B. An example of the use of an XML schema, XML-Data Reduced (XDR), can be seen in Appendix C. These two

examples are different representations of the same schema, and a sample XML document that could use either type of schema is provided as Appendix A.

A comparison can be made between the two approaches by taking a look at how one of the elements expresses its data constraints. Expressed using the DTD format in Appendix B, the *Target_Name* element is expressed as

```
<!ATTLIST Target_Name
     dtype NMTOKEN #FIXED "string"
     dsize NMTOKEN #FIXED "54"
>
<!ELEMENT Target_Name (#PCDATA)>.
```

Here, both *dtype* and *dsize* are fixed attributes that extend the DTD to provide data type and size constraints for each data element. Since they are extensions to the DTD standard, they require custom code for validation of these parameters. The same element expressed in XDR format is

```
<ElementType name="Target_Name" model="closed"
content="textOnly" dt:type="string"
dt:maxLength="54"/>.
```

In this case, each of the attributes are part of the XML-Data specification [LAY98], so any product that conforms to this specification should be able to properly validate XML documents based on this schema.

The choice of one approach over the other will primarily be based on the availability of COTS products that are able to interpret the respective schemas and the amount of custom code that must be written to perform validation. There are currently more products available for validating against the DTD, but this will change in the near future as the XML schema specifications become more solidified and they become more widely used.

The differences between the approach taken will determine the amount of structure and meaning that is either preserved or lost during the transition from XML to relational database. It will also determine the amount of custom code required to perform the mapping and the complexity and flexibility of the resulting data structure.

In moving data from highly structured relational databases to the semi-structure of XML documents, the concern is primarily with maintaining the meaning of the data through the transition, simplifying the procedure for performing the transition, and validating the data during its transition. Data typing within a database environment, however, serves the purposes of efficient storage, optimizing data queries, and classifying the structure of each element so that a common set of operations can be provided for each element type.

An important aspect of data typing is providing a mechanism for validation. Standard XML parsers provide

validation of the document structure by using the DTD. This does not handle, however, the validation of specific data types. As part of the data transformation model, therefore, there must exist a validation routine for each data type that takes into account the element size, precision, allowable characters, and other properties of the data type. Alternatively, the syntactic structure could be specified in enough detail so that some of these properties will be guaranteed by the parser by making the grammar restrictive enough so only valid data can be represented. This can require putting most of the information in the tags as attributes.

8. Performance

Although there are a number of advantages to using XML as a data transport, assembling and disassembling data as XML documents adds overhead that can affect the overall performance of the data exchange process. This thesis does not address performance issues in detail, but it is an important consideration that can affect the tools, technologies, and methods of implementation.

D. XML QUERIES

One of the requirements for transitioning data between databases and XML documents, is the ability to perform complex queries against the XML structure. The query requirements for processing and retrieving data from a

linked hierarchical structure such as an XML document are very different than those for a relational database structure. While query languages are fairly well established for processing data in relational databases, the same is not true of query languages for XML documents. A number of different approaches have been proposed for creating a query language that will address the requirements inherent in interfacing with hierarchical document structures, but none of these have been adopted as part of the XML specification to date.

There is a Working Group within the W3C that is dedicated to the development of XML query functionality. The purpose of XML queries, as stated by the latest draft document from the Working Group, is ``to produce a data model for XML documents, a set of query operators on that data model, and a query language based on these query operators.'' [XQL00] This will play a very important part in how data gets retrieved from XML documents in the future. It will allow ad hoc queries to be made against XML documents, or repositories of documents, in a wide range of environments and in a similar manner to the queries run against highly structured databases.

Several approaches for translating data between structured databases and XML's hierarchical structure have been discussed in the literature. David [DAV99] concentrated on the use of ANSI SQL's inner join operation

to perform transformations of data between a database and an XML document. This can translate relational data from a database into a hierarchical structure, suitable for storage as XML. Bourret [BOU99] describes two different approaches for mapping between an XML document and a database. In the first, template-driven mapping, there exists no predefined mapping between document and database, but commands are embedded in templates that get processed by the data transfer middleware. In the second type of mapping, model-driven, a data model gets imposed on the structure of the XML document and this gets mapped to the structures in the database and vice versa.

E. CHAPTER SUMMARY

This chapter has explored the use of XML for transporting data to and from relational database systems. The greatest advantage of using XML in this role is its flexibility. This flexibility brings with it, however, a number of important considerations, each of which need to be evaluated prior to designing a solution.

IV. ANALYSIS OF EXISTING XML TECHNOLOGY AND COTS TOOLSETS

The intent of the previous chapters has been to set the stage for the discussion that follows. A description of XML and its advantages and disadvantages as a tool for interoperability is a necessary preface to any analysis of how and where XML can be applied to solving data interoperability issues. This chapter covers the how and where by looking at relevant technologies and tools that currently exist and that can be applied toward resolving the issues expressed early in this document. An important part of this will be a description of the status of these tools and technologies, since many of the XML-related specifications and standards are still in their early stages.

A. RELEVANT TECHNOLOGY

One of the original design principles behind XML was that it support a wide variety of applications. A number of other design principles were directed at minimizing the complexity of XML and making it easy to use [BRA98]. A consequence of this is that although XML can be applied as a solution in many different application domains, it cannot alone provide the entire solution for most of the problems within these domains. To address this, there have been a number of related technologies and specifications that have

been developed to meet the requirements in these areas. This section discusses many of those technologies that are relevant for the exchange of data between relational databases. While this list is far from the full spectrum of XML solutions, it is a representative subset that can be applied directly to our discussions here.

1. Extensible Style Language Transformations (XSLT)

Extensible Style Language Transformations (XSLT), which are part of the recently approved XSL specification, specify transformations that can be performed on XML documents. In particular, this mechanism takes one XML document and transforms it into another XML document based on the static mapping information contained in a style sheet. This can be for the purpose of display or, more interesting for our purposes, to convert data to different DTD or schema formats. An example of using XSLT to perform this type of conversion is given in the *Track_list* example in section II.B.2, where the *coordinates* element from one schema exists as separate *latitude* and *longitude* elements in another schema. By applying an XSLT transformation and using an XSL stylesheet to specify the mapping between the schemas, the transformation can easily be performed.

In data sharing applications, transformations via XSLT might be applied at one or more points during the data transition. One use would be to apply changes to XML data

so that it will conform to a desired data format. As an example, consider a data source that produces XML data tagged with data types that are similar, but different, from those required by the format of the consumer. By applying an XSL template, and running the data through a transformer, we can produce valid data for the consumer without modifying the data source. This can be a cost effective way to create standardizing software wrappers for legacy systems.

Another possible use is to utilize XSLT to reduce some of the data from a specific dataset, or to completely reorganize a source document. This use will be applied later to the production of valid messaging formats from larger sets of XML data pulled from a database.

XSLT currently has the status of a *Recommendation* to the W3C [XSL99] and as such, it is relatively stable. Numerous products conform to the specification and have been developed specifically for performing translations using XSLT. A comprehensive list of these products can be found at the W3C web site for XSLT [XSL99]. There are also a large number of products, such as the BizTalk products discussed below, that have a larger scope, but which use XSLT at their primary transformation engine.

2. Extensible Query Language (XQL)

This section briefly describes the Extensible Query Language [XQL98]. Because this research focuses more on

utilizing XML for data transport and interoperability, and less on its use as a mechanism for storage, we do not cover details here. XML queries do become important, however, when data from a relational database is truly exported as XML or in the case that the data is stored within a true XML database. The discussion here focuses on XQL since it is likely to become the predominant query language for XML.

XQL performs the same function for the hierarchical XML structure as SQL performs for a relational database, in that it permits data access and manipulation. It has also been designed to assist with integration of multiple XML data sources. Of interest here is the design requirement that calls for the ability to perform queries on streams of XML data for the purpose of filtering, in a similar manner to the usage of Unix filters. This could be used for either extracting data from the streams, or for transforming the data stream to compress it.

The XQL language is still in the early stages of development at the W3C, although it is being based on other query languages that reached some level of stability. Both the query requirements and the data model have been recently submitted as a *Working Draft* to the W3C. There are a number of products that possess some form of either an XQL processor or one that handles a variation of the language.

These products are likely to change, however, as the standards solidify.

3. Simple Object Access Protocol (SOAP)

The Simple Object Access Protocol (SOAP) [BOX00] is an open messaging architecture, designed to transmit data from sender to receiver on top of the Hypertext Transfer Protocol (HTTP). SOAP is typically used to combine messages to create a request/response pattern. The contents of SOAP messages can be of any format, although the expected use is one or more XML documents.

While SOAP is not necessarily related to interfacing with databases, it is mentioned here because it has features that make it very attractive for use in the communication between different systems. It uses HTTP as its underlying transport protocol, for which numerous reliable COTS products exist. It has encoding features that can assist with encoding messages in binary format prior to transmission, as well as handling multiple XML document instances to be combined into the same message.

4. XML Schemas

As explained in sections II.B.2, III.B, and III.C.7, the DTD portion of the XML specification falls short when it comes to fully describing the set of properties that come with strongly typed data. It also, therefore, does not make it easy to validate these properties of the data when they

exist. The result has been a complete lack of standardization among the various implementations that must represent strong typing within XML.

This has led to a number of attempts to define a standard to address these problems. These efforts have now coalesced into a single standardization group, called XML Schemas, which plans to replace the DTD altogether for the use in applications requiring these additional capabilities.

Although the future use of XML as a data interchange mechanism seems to lie with one of these new approaches, this is an area that is still evolving. Most of the existing COTS tools have little or no support for XML Schemas and require the use of a DTD, although this situation is quickly changing.

B. COTS TOOLS FOR DATA EXTRACTION AND TRANSLATION IN XML

The COTS tools that are of most interest for this evaluation fall into one of two categories: middleware or parts of specific Database Management Systems. The first category of tools are relevant to utilizing a standard method for interfacing with a database such as Open Database Connectivity (ODBC) [ODB95] or Java Database Connectivity (JDBC) [JDB00], and then performing the translation from data elements to XML using a middleware solution that handles the mapping. In certain cases, when the data can not be exported directly via ODBC/JDBC due to legacy or

security issues, some middleware solutions can still be used with success by using an existing API or some different method for extracting and updating the data.

The second category of tools involves utilizing the features of the different database management systems for accessing and storing data as XML. Although there are several types of databases in this category, including relational, object, and true XML databases, only relational databases will be considered here since they are used by the legacy systems that are of interest.

As might be expected, the tools evaluated in this section vary from small tools, targeted to address one specific area, to larger environments that possess a great deal of functionality across a wide spectrum. The tools evaluated here provide only a sampling of the tools that are currently available in each category. One of the tools discussed in greater depth, Sybase Application Server Enterprise (ASE) version 12, represents the category that is specific to a single Database Management solution. This DBMS was chosen since it can provide a solution for data exchange from GCCS-I3, which is covered in greater detail in Chapter V.

1. Middleware Tools

The term ``middleware'' covers a large category of tools. In this context, the term is used to refer to XML

tools that lie between the source and destination databases, and provide data translation, manipulation, and mapping services. Data translation is a basic requirement for achieving interoperability in an environment where the data sources are directly modified.

In some cases, the use of third party middleware tools that do not currently have XML capabilities are worth investigating for integrating data from different sources. This would be feasible if the data is already exported as XML from the database, or if another mechanism can be used for relational to XML mapping. This option is worth considering primarily due to the power and functionality that exists within this category of tools.

One example is *Data Joiner* [DAT00] from IBM. It can provide integration capabilities on a large scale, such as transparent SQL access and relational joins across multiple different DBMSs. It also provides comprehensive APIs for working with data that can not be exported via standard access, such as ODBC and JDBC.

Another comprehensive tool for performing translations between different types of database formats is Microsoft's *Data Transformation Services* [DTS00]. This tool can operate independently of the DBMS to perform translations between a large number of formats. It has both offline and online processing modes, and while it does not yet have XML

capability as part of the translation services, it can be used in conjunction with *Active Data Object* (ADO) technology to provide this capability.

One type of schema mapping and data translation tool is Microsoft's BizTalk products and specifications. BizTalk is a comprehensive set of products that provides a number of services at runtime, including document validation against a set of business rules, translation of data formats, schema transformation, document transportation, and tracking and logging capabilities. Its transformation and mapping capabilities are built on top of an XSLT engine, which can perform mappings between a number of different formats.

Similar to other middleware products, the BizTalk server provides the processing functionality which maps data to an XML stream based on a mapping provided by each organization. The schema used by the BizTalk framework is currently implemented in XML Data Reduced (XDR) [LAY98] and Microsoft has promised that the XML Schema standard will be used when it is finalized and released by the W3C. This can provide a significant advantage over the use of a DTD for specifying the schema and for document validation, since it will handle many of the data constraint concerns described previously in Chapter III.

One difference between BizTalk and some of the other middleware frameworks, is that BizTalk is designed as an

end-to-end product, and handles the transmission of XML streams from source to destination over a number of different possible transport protocols. In the typical scenario, both the data source and destination would have BizTalk server platforms. The application layer would contain the business rules specific to that organization, would be responsible for retrieving the data as well formed XML, and would format it for handling by the server. The server then validates the documents, and processes them for transmission to the destination BizTalk server. The destination then performs XSLT translations and mappings into the proper destination format.

This framework is attractive because it contains many of the elements that need to be in place for end-to-end data sharing. This includes data translation and mapping, transport, and a schema representation designed to easily map to a relational schema without loss of metadata. It is, like many of the XML products, still in flux and dependent upon related specifications to be finalized.

XML-DBMS is a model-driven, open source middleware solution for moving data between relational databases and XML [XDB00]. This tool is discussed here because it is representative of a number of the open source tools that are available, some of which have the same approach as this tool.

XML-DBMS has been implemented in both Java and Perl, and consists of an API to a set of packages that provide services for extracting data from a relational database into an XML format, and for taking data already in XML and inserting it into a relational database. The product uses ODBC and JDBC interface standards for accessing the data, so it can be used with Sybase, Oracle, SQL Server, or any other database server that has JDBC or ODBC drivers. In order to perform the mapping, an object view of the targeted XML format is developed. This object view is then mapped to the relational schema by taking the object properties, represented by XML elements and attributes, and linking them to specific columns in the database. This mapping is then performed at runtime whenever data is moved to or from the database. This method has the advantage of requiring no modification to the database.

2. XML Enabled Databases

A number of the vendors of larger database management systems, including Sybase, Oracle, Informix, IBM, and Microsoft, have been integrating XML capabilities directly into their respective database systems. Although each of the systems is implementing XML in different ways, and sometimes for different uses, there are many similarities in the functionality provided. This section lists some of the

ways that this XML functionality is being provided and can be utilized.

In general, the XML capabilities being integrated into databases fall into two broad categories: document-centric and data-centric. Document-centric capabilities to focus on the storage and access of documents, which can be characterized by an irregular structure and larger grained database representation. An example of this might be a product user's manual, which can be stored in its entirety, or in parts, as XML in the database. In contrast, data-centric refers to a more regular structure where each XML element has a corresponding data element within the relational database. While the lines between these two categories are not always clear, the focus here is primarily on data-centric XML functionality.

There are a number of advantages to using XML extensions to existing databases rather than third party middleware. In many cases, performance enhancements can be expected since better optimization can often be performed by the database vendors because they have access to the underlying structure of the database. Additionally, since data can often be stored or maintained in memory in XML format, instead of the tags and hierarchical relationships being established when the data is requested, gains in performance and architectural simplicity can result. An

example of this is the use of database *XML views*, which can be used to maintain a subset of data in the desired XML format. Queries are then executed against the view instead of requiring joins between the different database tables. This capability currently exists only in a small number of database systems.

Disadvantages to the use of database XML extensions include the fact that not all database systems currently have the same level of capability, and there are a number of differences in the ways that the capabilities are being implemented. So, while it might be possible to implement a third party mapping and translation capability in a similar manner across a set of differing systems, the effort and complexity of developing directly to each of the individual systems might be extensive. Much of the XML functionality in these systems is also currently either in beta form or still in the process of refinement, so each database system should be evaluated on a case-by-case basis.

Most major database systems now have some method available for exporting data as XML. Relational database systems that have integrated XML capability typically provide this capability in three different forms. One form allows XML-formatted documents to be generated from the individual data elements stored in the database. Another form involves extracting the data and structure from an XML

document for insertion into a database. The third form of functionality allows entire XML documents to be stored as a single entity within the database. This discussion deals only with the first two forms and their specific application in the Sybase Application Server Enterprise (ASE) [SYA00] database, although comparisons to implementations in other database management systems will be explored.

Within Sybase ASE Version 12, this capability is provided through use of Java tools and the Java interfaces to the DBMS. This means that in order to transfer data to or from the database as XML, the Sybase Java API must be invoked from custom code. The functionality of the API is limited to performing basic mapping functions and is similar to that of some of the middleware products discussed. Details on the process can be found in [SYB00].

Sybase ASE currently lacks some of the XML capability found in other large DBMSs, such as Oracle 8i, DB2, and SQL Server. Specifically, these other DBMSs provide additional transformations on query results through the use of XSLT. Additional capabilities, such a publishing views as XML and facilities for conducting XML queries are not currently available within Sybase ASE.

3. Additional Tools

a) Parsers

Virtually every tool that works with XML requires an XML parser. The parser is a software component that takes an XML document as input, reads and interprets the structure of the document, and then returns the result to the application for manipulation.

There are two basic types of parsers, one that produces a complete data tree as output and another that is event based. Parsers using the tree model typically produce an entire structure representing the document in memory prior to allowing any operations on the data. Once the document has been completely parsed, the resulting structure is passed on to the application either for direct use, or in the form of an API based on the Document Object Model (DOM) [DOM00].

With the event based parser, the application registers specific events that it is interested in, and it is then notified of these events as the parsing is taking place.

There are numerous parsers available, either for standalone use or integrated as part of another product. Many of the products listed already have integrated XML parsers, so individual products will not be discussed here.

b) XML Editors

An important part of any work with XML is an XML editor. While not specifically used for database work, a good XML editor can assist the developer greatly with producing valid XML documents for testing and performing certain types of transformations.

Only recently have XML editors approached the capability of editors found in other disciplines, but some of the work in this area has begun to redefine the role of an XML editor. The functionality available includes parsing, validating and editing not only XML, but also XSL, DTD, DCD, and other schema dialects. Another helpful function is the automatic generation of DTD or the various XML Schema dialects from XML source. Other capabilities include two-way translation between XML and tabular formats and XSL translations. These capabilities together provide more of the Integrated Development Environment (IDE) approach found with many of today's mature development environments.

c) Compression Technology

One area of concern with respect to the use of XML has been its verbosity. Like any textual markup language, XML contains a lot of redundant information and carries with it a great deal of overhead in terms of the metadata. This has brought about the need for compression technology to

reduce the size of the XML output. There are a number of products now available that address this need.

The wireless community has needed to address the same problem, and has developed a specification for handling XML compression in a standardized manner. The *Wireless Application Protocol Binary XML Content Format Specification* [WAP99] was developed specifically to reduce the transmission size of XML documents, in order to allow a more efficient transfer of data in XML format. The specification addresses low level details, such as byte-ordering and character encoding, that normally do not need to be handled by XML applications. This work could easily be leveraged to provide the same type of service for database to database transfer.

C. CHAPTER SUMMARY

There is a wide range of COTS tools and technologies for XML, many of which offer solutions for interfacing with relational database systems. Choosing the proper solution, in many cases, can be difficult because there can be a number of potential solutions, each which have different, but overlapping, functionality. This is clear when evaluating the various middleware tools and the XML-enabled database systems, which provide much of the same functionality. There are also a number of significant

variations in how they are integrated, so each must be closely evaluated.

V. XML TRANSPORT FROM GCCS-I3

A. OVERVIEW

Up to this point, we have discussed in general terms methods for applying XML and its related technologies to interoperable data interchange. Here, we will look at more of the specifics. This chapter focuses on the use of XML technologies and COTS products that have been previously discussed, and a process for applying them to one specific database architecture, GCCS-I3.

The first part of this Chapter outlines the steps involved. It includes a section that lays out the design goals for the process, a description of the steps involved in the process, and a description of the GCCS-I3 MIDB schema, which are used for illustration. Following this is a description of a data model and examples of an XML-to-relational-database mapping that could be used. The second part of the chapter applies some of the XML COTS tools and technologies to assist with the process. Finally, an assessment is made of the advantages and disadvantages to the data exchange process.

B. STEPS FOR END-TO-END DATA EXCHANGE

1. Discussion

Early efforts to utilize XML for data sharing within the DOD have included XML-MTF [MTF00] and XML-CIX [INR00] messaging. While these efforts represent a good first step towards data interoperability through an increased use of COTS tools, they will still be subject to many of the limitations of the messaging standards on which they are based (see section II.B.6). Basic MTF and CIX messaging, as they currently exist, are also too restrictive to allow the level of data interchange necessary to address the needs of a Joint Battlespace Infosphere, as described in [JBM00]. XML will provide us the opportunity to easily expand upon these messaging standards, and when communicating with systems that require legacy messaging, transform the same data into a valid MTF or CIX message.

A comparison can be made here to work that has been done within the METCAST weather reporting system. The Weather Observation Definition Format (OMF) [OMF00] is a recent application of XML by SPAWAR PMW-185 to address shortcomings in *weather observation reports*. The reports are issued in a number of different messaging formats that are similar in nature to DOD tactical messaging formats. The set of problems that needed to be addressed can be summed up as a basic inability to extend the messages to

provide additional information in cases in which it is needed. Some of the information needed for interpretation of the messages was maintained externally to the messages themselves, presenting problems when this information was not available.

OMF was developed to provide annotations that would extend the existing weather reporting formats. In a similar manner, the process described and assessed below is designed to create a data sharing environment that is usable by existing military systems, yet extensible enough to accomplish interoperability objectives.

2. Design Goals

A number of the design goals that are necessary for this process are similar to those for making the move from a legacy message structure to XML, such as those expressed in [MTF00]. The main differences would be related to the emphasis here on database to database transfer and a lack of adherence to all details of a specific message format.

A basic set of design goals for our process follow:

- The data exchange process must handle multiple databases, each with a different schema, running under different DBMSs, and containing data, some of which has the same or similar semantics.
- The schemas for the existing databases cannot be modified.
- Where appropriate, data must be easily transformed between different message formats, including, but not limited to, USMTF, CIX, NATO Allied Data

Publication Number 3 (AdatP-3), and Theater Ballistic Missile (TBM) track format.

- Operations that act on the XML schema must allow the schema to change over time.
- The XML schemas must be simple to construct using data element definitions derived from a central repository.
- Use of standardized technologies and COTS tools must be used wherever possible.

3. Process

The design of an architecture for transfer of data between the databases can be broken down into a series of steps, some of which, in practice, might be combined to form a single step. Here, they will be addressed as distinct steps in order to remain implementation independent.

The primary step of interest, since it will have the most effect on the XML toolset to be used, is to develop the data model and method for performing relational to XML mappings. This will be discussed in some level of detail in relation to the Modernized Integrated Database (MIDB). Other steps, some of which will be covered in lesser detail include the following:

- Establish a common vocabulary.
- Analyze legacy systems to determine the subset of data to be shared.
- Analyze legacy systems to obtain data structures, semantics, etc.

- Determine mechanism for accessing the data (e.g. stored procedures/triggers, special APIs, ad hoc queries, etc.)
- Determine an XML Data transformation engine
- Determine mechanism and protocol for data transport.
- Determine tools that can be used to assist with or provide the capabilities needed for each of the items above. This includes determining the amount of custom code that will be required and the amount of risk involved with specific tools and technologies.

While each of these steps are important to the overall process, the ones that will be covered here involve developing a data model, interfacing with the database, and providing a mechanism for communication. In doing so, the large number of choices that need to be made will be reduced to a smaller subset. The process will be greatly simplified here and many of the details will ultimately need to be filled in, but the purpose here is to lay some groundwork for how this task can be achieved using available COTS products and existing technologies. Figure 1 shows a view of the overall architecture.

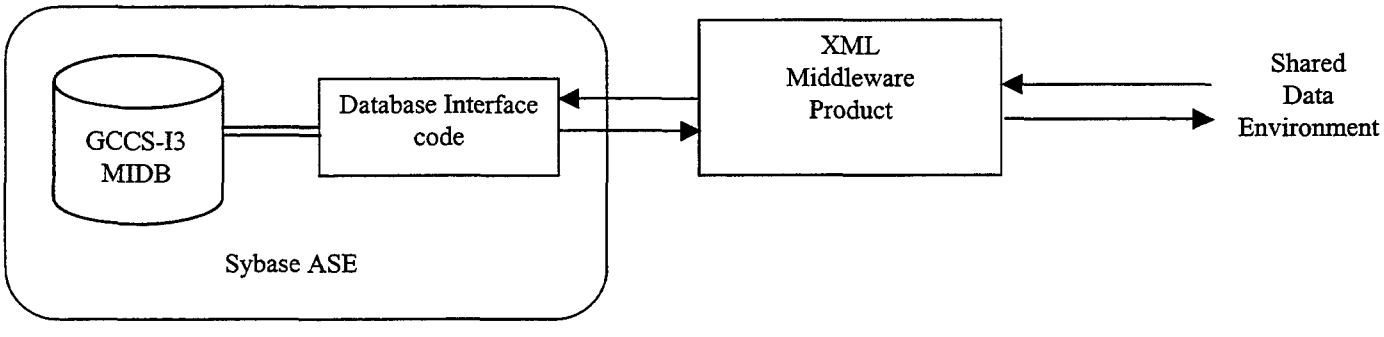

Figure 1: GCCS-I3 Data Sharing

4. GCCS-I3 MIDB Segment Schema

The Modernized Integrated Database (MIDB) serves as the primary data repository for general intelligence data within the DOD. There are a number of aspects about the MIDB schema, as it currently exists, that will affect the method used for extracting data from it.

One consideration is the version of Sybase ASE in use. The XML feature set provided by Sybase only exists in Sybase version 12, while the current version of the MIDB requires the use of Sybase 11. While middleware tools outside of the DBMS can be used to eliminate versioning problems, this decision needs to be made in advance.

The example set of data to be utilized references the following tables from the MIDB [MID98]:

- **TGT_MSN** - contains information about the missions against targets.
- **TGT_OBJ** - contains information for a military operation involving targeting.
- **TGT_LIST** - contains information on a prioritized, validated target set.
- **TGT_DTL** - refers to a specific target.
- **FAC** - contains data about a facility or an installation.

A subset of the elements from TGT_MSN are described below. The full descriptions of all elements can be found in [MID98].

MSN_ID	varchar(15), NULL
A unique identifier for the mission.	
OPERATION_NAME	varchar(54), NULL
The name used to describe an exercise or live set of military missions and activities.	
CLASS_LVL	char(1), NOT NULL
Highest classification level of the data contained within the record. Permissible values: U (Unclassified), C (Confidential), S (Secret), T (Top Secret)	
CODEWORD	char(1), NULL
Indicates the appropriate control channels associated with a physical records classification. Permissible values: 0 (collateral), 1-3 (SI-1), 4-7 (TK-1)	
MSN_NAME	varchar(30), NOT NULL
Name of the mission.	

The tables, elements, and constraints being used in the sample data are fully described in [MID98]. A full example in XML format is given in Appendix A. The corresponding DTD for the example is given in Appendix B. The same

representation, but in XDR schema format, as used by BizTalk Framework, can be found in Appendix C.

5. Data Model and Mapping

One of the goals here is to provide a data model that will allow mapping to the common DOD messaging formats, but without the full spectrum of associated constraints. This will be similar in nature to the data model introduced for the *Observation Markup Format* introduced in [OMF00].

Examples will be given of the data model as a DTD and also as an XML Data Reduced (XDR) schema. Use of a DTD requires extensions to the XML Specification in order to represent data characteristics such as data type, repetition constraints, and other similar constraints. Some other characteristics, such as designating uniqueness or designating values as being required, are part of the specification.

Extending the DTD will mean that some of the data validation will not be performed by standard COTS tools, but additional checking will need to be performed in external custom code. While extensions are not necessary for any of the XML Schemas, such as XDR, the specifications for these standards have not yet been solidified, so the availability of tools is limited and subject to change.

The method used here to extend the DTD is to designate, for each element, a set of fixed attributes with values that

describe the properties of the element. An example of this, which describes a Mission_ID element as having a datatype of *string* with a maximum size of 15, is:

```
<!ELEMENT Mission_ID (#PCDATA) >

<!ATTLIST Mission_ID dtype NMTOKEN #FIXED `string'
dsize NMTOKEN #FIXED `15' >.
```

This set of fixed attributes will be associated with every PCDATA (character-data-only) element. An additional set of attributes can be easily added to express other properties of an element that need to be checked. The datatypes utilized, for simplicity, are those from the XML-Data [LAY98] submission to the W3C. The full DTD listing can be found in Appendix B.

The example above, described using XDR, would be:

```
<ElementType name="Mission_ID" model="closed"
content="textOnly" dt:type="string"
dt:maxLength="15"/>.
```

Note that XDR is expressed in XML syntax, and the element description consists of an empty element with multiple attributes for describing the data properties. The attribute *model* refers to the ability to extend the element data description. The *content* attribute describes the content of the element as only text, only other elements, mixed, or empty. The *dt:type* attribute describes the data

type. Additional information can be found in [LAY98] and the full XDR listing is in Appendix C. Both these representations provide the same set of data constraints. The primary difference between the two, as discussed in section III.C.7, is the support provided by current COTS applications and the corresponding amount of custom code required.

In the relational to XML mapping, tables can be represented as elements which can only contain other elements. Columns can be represented as either PCDATA elements or elements that contain other elements, for the case in which the column expresses a relationship. The decision could have been made to represent columns as attributes, but representation as an element was chosen to provide greater flexibility.

It would also be possible here to express relationships within the data as primary key/foreign key fixed attributes. This would have the effect of flattening out the XML structure, providing a closer mapping to the relational structure and reducing data redundancy. The structure described, however, provides a more natural transition to messaging formats.

C. APPLICATION OF EXISTING COTS TOOLS AND TECHNOLOGIES

This section explains potential use of the various COTS product types for a number of the steps listed above in

order to achieve the design goals listed. Additionally, it covers the use of XML related technologies as they might be applied to assisting with this process.

As indicated by the previous discussion of the current state of XML technology and the challenges that exist when applying it to interfacing with database systems, there are a number of decisions that must be made in order to design an architecture that can support data exchange via XML. Since many of the specifications and tools that form the basis for XML are still in flux, there are many areas where choices need to be made. Most of these choices are dependent on the existing architectures of the systems involved.

1. Analyze Data Structures and Semantics

The goal of this step is to understand data structures, relationships, and rules that apply to the data well enough to map the data to a different schema without a loss in meaning. Much of this work must be done manually. The level of effort required may depend on a number of things, including amount and quality of documentation, complexity of the data relationships, type of data storage, and amount of business knowledge that is buried in applications.

One concern with the MIDB is the use of ``tie'' tables throughout the database for establishing relationships between tables instead of using primary/foreign key

relationships. This has the affect of not only increasing the number of joins required in order to obtain meaningful results, but also can make the transition to XML format more complex.

2. Data Access

As outlined above, the method for data access from the database needs to be determined. Most of the existing data propagation from the MIDB is provided via a series of database triggers and stored procedures, so this would be the expected method. A typical scenario might be that one or more updates occur to the data identified for XML transport, which fires a trigger that is in turn responsible for passing the data through the XML mapping and transformation engines.

One additional consideration is the method of interfacing with the data, which can take a number of different forms, and while the tools to be utilized may be affected by the method used, this will only be briefly discussed here since the XML transformations can be transparent to the data interface.

One interface method could involve ad hoc queries through an ODBC or JDBC interface. These queries could interface directly with the data or with stored procedures that implement the data queries. A number of the tools discussed in chapter IV provide APIs for this type of

interface. While the mapping capabilities of these tools typically can be used regardless of the data access method, certain advantages would result from using one of the standard interfaces.

Another interface method is to utilize existing application interfaces, such as those described in [GCC98], which provides an application interface to GCCS-I3 MIDB. Utilizing the published API would allow use of any intelligence (i.e. business rules) that is built into the API. Use of the API may also sometimes be a requirement due to security or other reasons, although with GCCS-I3 this is not the case. However, this is likely to greatly restrict the types of queries that can be processed, the result set that can be returned, and may in some cases negatively affect performance.

One method of publishing the data as XML is to use database *views*. Views allow data to be presented in a number of logical combinations that are independent of the underlying representation of the data. This is exactly what is needed in order to produce data as XML. By using a view to produce data structures that are the same as, or similar to, that of the targeted XML format, much of the data transformation has been done prior to extraction from the database. Combining views with a ``publish as XML'' feature in the database can allow virtually all of the

transformation to XML to occur at the server side, reducing the complexity and processing requirements of the client. This can also greatly improve performance at transaction time, since the actual transaction would not involve any of the overhead associated with data transformations and mapping.

Unfortunately, Sybase is not among the database vendors that currently support publishing views as XML. The use of views can be made, however, by publishing the subset of data that will be available for XML transactions. This will simplify the overall process and remove the overhead that would normally be required for performing joins and preparing the data for extraction.

3. XML Data Transformation Engine

One of the design steps addresses the need to transform data sets to one or more of the standard message formats. XSLT can be integrated here as part of the middleware solution for performing this translation. Both USMTF and CIX messaging formats currently have XML extensions. Applying XSLT with an appropriate stylesheet would allow the XML data to be transitioned to one of these formats.

Another type of transformation that could be made using XSLT would be to take the data to a display format. An example of this would be the ability to pull targeting data

out of the MIDB for mission planning purposes, and format the data for display within a web browser.

Inserting an XQL processor in the XML data stream provides yet another method for filtering or transforming the XML data stream.

4. Method and Protocol for Data Transmission

An important consideration in the widely distributed DOD operating environment is how to distribute the data once it is in XML format. Two of the technologies discussed in Chapter IV would provide this capability, and both work in a similar manner.

Both the Simple Object Access Protocol (SOAP) and Microsoft's BizTalk framework have communication facilities for transmission and reception of XML data. They both perform message routing through the use of external wrappers on messages with XML content. These would be valid mechanisms for data transportation.

D. ASSESSMENT OF EFFECTIVENESS

It is possible to utilize XML and its associated technologies to greatly reduce the barriers to data interoperability. This issue is very important in the face of past failures in this area. It is clear from the previous assessment, however, that this is still not an easy task, there are many choices to be made, and the process is not risk free.

One conclusion that can be drawn about each of the products and technologies that have been reviewed, is that they are relatively new to the market and they each represent the first generation in their respective categories. The result of this has been a lack of satisfactory case studies from which to draw conclusions. This fact, although it will certainly change over time, will increase risk for investments in products and technologies.

It is also evident from the study that there is no single product or technology that can be expected to accomplish each of the goals. Further, it appears that no single vendor seems to dominate in this area, and that each of the individual products work well together. The overall affect from this is an increased modularity within the architecture and decreased risk from not having a reliance on any single product or technology.

The most established set of products and technologies appear to be the middleware tools for performing data mapping and translations. This is true in terms of the number of products available and in their functionality. The XML-enabled DBMSs are mostly still in the early stages of adopting XML and, in the case of Sybase, appear to be lacking in functionality.

It is clear that some of the steps listed in Section B.4 will not be easily accomplished. Analyzing the existing legacy structures is never an easy task, even when using

COTS products to assist with evaluations. Much of the necessary information is buried in application code that can be very time consuming to analyze.

One of the challenges that must be overcome in translating between the extended messaging discussed here and standard message formats is handling the set of restrictions that standard messaging formats place on the structure of the messages. One example is the allowable line length and message length for the OTH-G format. OTH-G message lengths are limited to 100 lines, and line lengths are restricted to 69 characters [JWI00].

In surveying the COTS tools available on the open marketplace, it is clear that there are important distinctions that need to be made between the diverse set of requirements that exist within the DOD when it comes to data interoperability, and the more narrowly focused requirements presented by standard business applications. While many tools are being developed specifically to address data interchange between heterogeneous database systems, tools that are designed for use in the business to business arena are not always suited to handle some of the requirements that arise due to the size and diversity of data within the DOD.

One example of these differences is in the area of data access. While many of the COTS tools may require some standard method for interfacing with a database, such as

ODBC or JDBC, this may not be possible, or even allowed, when interfacing with certain DOD databases which require the use of a specific API for all data access.

E. CHAPTER SUMMARY

Although the tools and technologies do exist for providing shared data access from a legacy environment, it is clear that there is no general solution and that each system must be evaluated separately. It is also evident that the choices may not be clear and the maturity of the tools and the specifications on which they are based will be a big consideration.

Although the technology is still evolving, there is a great deal of work being done with XML, both within the DOD and in the business community. It will be important to look at parallel efforts in different fields. The Weather Observation Definition Format is one example of a parallel effort that can lend valuable insight.

VI. CONCLUSION AND FUTURE RESEARCH POSSIBILITIES

This thesis assesses some of the techniques and existing technologies for applying XML to the exchange of data between different database systems. While the standards and specific implementations that have been discussed represent the state of current technology, this work by no means represents the final chapter. XML and its associated standards are a moving target, with new uses and strategies for use being developed daily. This document can, however, provide a description of a process for the use of XML in data exchange and the problems that must be addressed.

The first part of this thesis identifies several of the issues affecting data interoperability within the DOD. The primary issue is that legacy systems, originally developed to address a single set of requirements, are very difficult and costly to modify to share data with other systems. At the data level, integrating differences in schemas and in the rules applied to those schemas is very difficult to achieve.

XML is well suited to help solve these issues. One of the primary driving factors behind the use of XML is that it provides the capability to develop common centralized schemas, without modifying the legacy database schemas.

This is critical since it addresses one of the main problems with past approaches to data integration.

Applying the use of XML is, by necessity, a phased approach. Recent developments have been directed at adapting standardized DOD messaging solutions to use XML. This provides immediate advantages, since operational systems that already rely on these messaging solutions can be easily adapted to utilize XML. This alone, however, will not provide data sharing to the extent required and it does not leverage the full advantage that can be provided by XML. The next step in the process involves detailed analysis of existing systems, the development of a common schema, and the application of data translations and mappings for each system.

The evaluation of XML tools and technologies and their application to a subset of the GCCS-I3 MIDB schema provided some insight into where problems exist and which questions remain unanswered. The primary observation that stood out, as might be expected with a relatively new technology, is that the tools are greatly mixed in terms of functionality and maturity. Another observation, which is really a characteristic of the XML design, is that multiple tools will probably need to be applied in order to accomplish the task.

An important question that remains to be answered is whether the application of available tools provides an

adequate level of performance. This has been a criticism of XML in the past, and, although there are ways to improve the overall efficiency, this will require close evaluation. Another question that should be answered is what the timeframe would be for the design and implementation of a specific approach.

The next logical step is to apply some of the tools and technologies discussed to sharing a subset of data between two systems. This would initially take the form of a set of requirements and design specifications, followed by some prototype work. As suggested in this thesis, this work should take a step beyond the transmission and reception of USMTF or CIX messaging, although it would be good to utilize a filter to transition subsets of data to these message formats.

As specifications become solidified and the product base matures, new approaches to solving the data interoperability problem may surface. While it is highly unlikely that there exists a silver bullet approach that will solve all the problems, XML has the capability to greatly reduce costs and simplify efforts to create a viable data sharing solution.

APPENDIX A (XML Document Listing)

```
<?xml version="1.0" encoding="UTF-8"?>
<Target_Mission>
     <Mission_ID>152-XX-221</Mission_ID>
     <Operation_Name>Tandem Thrust</Operation_Name>
     <Classification_Level>T</Classification_Level>
     <Codeword>5</Codeword>
     <Mission_Name>Strike Package 322</Mission_Name>
     <Target_Objective>
          <Country>IQ</Country>
          <Execution_Date>20000927</Execution_Date>
          <Functional_Production_Area>FUELS</Functional_Production_Area>
          <Priority_Objective>3</Priority_Objective>
          <Record_Status>E</Record_Status>
          <Domain_Level>SI</Domain_Level>
          <Eval>2</Eval>
          <Originating_Agency>EA</Originating_Agency>
          <Objective_Name>Airfield in Area 301</Objective_Name>
     </Target_Objective>
     <Target_List>
          <Operation_Name>Tandem Thrust</Operation_Name>
          <Classification_Level>T</Classification_Level>
          <Date_Created>20000825194500</Date_Created>
          <Date_Last_Changed>00000000000000</Date_Last_Changed>
          <Domain_Level>SI</Domain_Level>
          <Target_List_ID>12226</Target_List_ID>
          <Target_List_Status>A</Target_List_Status>
          <Target_List_Type>JTL</Target_List_Type>
          <Target_List_Name>Area 301 Tamino Airfield
Desig</Target_List_Name>
          <Production_Level>S</Production_Level>
          <Record_Status>E</Record_Status>
          <Target>
               <Affiliation>H</Affiliation>
               <Country>IQ</Country>
               <Classification_Level>T</Classification_Level>
               <Condition>COM</Condition>
               <Coordinates>325218290N1170928640E</Coordinates>
               <Coordinate_Basis>2</Coordinate_Basis>
               <Coordinate_Derivative>PM</Coordinate_Derivative>
               <Date_Created>19991011160000</Date_Created>
               <Date_Last_Change>00000000000000</Date_Last_Change>
               <Hardness>M</Hardness>
               <Height>320.0</Height>
               <Domain_Level>SI</Domain_Level>
               <Elevation>2040</Elevation>
               <Elevation_Confidence>100</Elevation_Confidence>
               <Target_Name>Control Tower</Target_Name>
               <Evaluation>1</Evaluation>
               <Radius>125.0</Radius>
               <Review_Date>20000825190000</Review_Date>
               <Release_Mark>MQ</Release_Mark>
               <Facility>
```

```
                <Access>CLRMO</Access>
                <Activity>ATC</Activity>
                <BE_Number>1014-8Z-3967</BE_Number>
                <Category>40812</Category>
                <Evaluation>1</Evaluation>
                <Facility_Name>Tamino Control Tower</Facility_Name>
                <Facility_ID>32008</Facility_ID>
                <Location_Name>Tamino Airfield</Location_Name>
                <Primary_Mission>DQ</Primary_Mission>
                <Relative_Ranking>1</Relative_Ranking>

<Population_Area_Proximity>14</Population_Area_Proximity>
                <Record_Status>E</Record_Status>
                <Review_Date>20000825190000</Review_Date>
                <Graphic_Agency>DIA</Graphic_Agency>
                <Graphic_Country>US</Graphic_Country>
            </Facility>
        </Target>
        <Target>
            <Affiliation>H</Affiliation>
            <Country>IQ</Country>
            <Classification_Level>T</Classification_Level>
            <Condition>COM</Condition>
            <Coordinates>325177560N1170823930E</Coordinates>
            <Coordinate_Basis>2</Coordinate_Basis>
            <Coordinate_Derivative>PM</Coordinate_Derivative>
            <Date_Created>19991011160000</Date_Created>
            <Date_Last_Change>00000000000000</Date_Last_Change>
            <Hardness>H</Hardness>
            <Height>20.0</Height>
            <Domain_Level>SI</Domain_Level>
            <Elevation>2040</Elevation>
            <Elevation_Confidence>100</Elevation_Confidence>
            <Target_Name>Bunker</Target_Name>
            <Evaluation>1</Evaluation>
            <Radius>235.0</Radius>
            <Review_Date>20000825190000</Review_Date>
            <Release_Mark>MQ</Release_Mark>
            <Facility>
                <Access>CLRMO</Access>
                <Activity>STG</Activity>
                <BE_Number>1014-8Z-3976</BE_Number>
                <Category>40812</Category>
                <Evaluation>1</Evaluation>
                <Facility_Name>Bunker for A/C Storage</Facility_Name>
                <Facility_ID>32010</Facility_ID>
                <Location_Name>Tamino Airfield</Location_Name>
                <Primary_Mission>DQ</Primary_Mission>
                <Relative_Ranking>2</Relative_Ranking>

<Population_Area_Proximity>14</Population_Area_Proximity>
                <Record_Status>E</Record_Status>
                <Review_Date>20000825190000</Review_Date>
                <Graphic_Agency>DIA</Graphic_Agency>
                <Graphic_Country>US</Graphic_Country>
            </Facility>
        </Target>
```

```
        <Target>
            <Affiliation>H</Affiliation>
            <Country>IQ</Country>
            <Classification_Level>T</Classification_Level>
            <Condition>COM</Condition>
            <Coordinates>325377680N1171033970E</Coordinates>
            <Coordinate_Basis>2</Coordinate_Basis>
            <Coordinate_Derivative>PM</Coordinate_Derivative>
            <Date_Created>19991011160000</Date_Created>
            <Date_Last_Change>19991205132000</Date_Last_Change>
            <Hardness>H</Hardness>
            <Height>0.0</Height>
            <Domain_Level>SI</Domain_Level>
            <Elevation>2040</Elevation>
            <Elevation_Confidence>100</Elevation_Confidence>
            <Target_Name>Runway</Target_Name>
            <Evaluation>1</Evaluation>
            <Radius>2000.0</Radius>
            <Review_Date>20000825190000</Review_Date>
            <Release_Mark>MQ</Release_Mark>
        </Target>
    </Target_List>
</Target_Mission>
```

APPENDIX B (Sample Data DTD)

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Access (#PCDATA)>
<!ATTLIST Activity
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "3"
>
<!ELEMENT Activity (#PCDATA)>
<!ATTLIST Affiliation
      dtype NMTOKEN #FIXED "char"
>
<!ELEMENT Affiliation (#PCDATA)>
<!ATTLIST BE_Number
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "10"
>
<!ELEMENT BE_Number (#PCDATA)>
<!ATTLIST Category
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "5"
>
<!ELEMENT Category (#PCDATA)>
<!ATTLIST Classification_Level
      dtype NMTOKEN #FIXED "char"
>
<!ELEMENT Classification_Level (#PCDATA)>
<!ATTLIST Codeword
      dtype NMTOKEN #FIXED "char"
>
<!ELEMENT Codeword (#PCDATA)>
<!ATTLIST Condition
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "4"
>
<!ELEMENT Condition (#PCDATA)>
<!ATTLIST Coordinate_Basis
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "2"
>
<!ELEMENT Coordinate_Basis (#PCDATA)>
<!ATTLIST Coordinate_Derivative
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "2"
>
<!ELEMENT Coordinate_Derivative (#PCDATA)>
<!ATTLIST Coordinates
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "21"
>
<!ELEMENT Coordinates (#PCDATA)>
<!ATTLIST Country
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "2"
```

```
>
<!ELEMENT Country (#PCDATA)>
<!ATTLIST Date_Created
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "14"
>
<!ELEMENT Date_Created (#PCDATA)>
<!ATTLIST Date_Last_Change
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "14"
>
<!ELEMENT Date_Last_Change (#PCDATA)>
<!ATTLIST Domain_Level
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "2"
>
<!ELEMENT Domain_Level (#PCDATA)>
<!ATTLIST Elevation
      dtype NMTOKEN #FIXED "float"
>
<!ELEMENT Elevation (#PCDATA)>
<!ELEMENT Elevation_Confidence (#PCDATA)>
<!ATTLIST Eval
      dtype NMTOKEN #FIXED "char"
>
<!ELEMENT Eval (#PCDATA)>
<!ATTLIST Evaluation
      dtype NMTOKEN #FIXED "char"
>
<!ELEMENT Evaluation (#PCDATA)>
<!ATTLIST Execution_Date
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "8"
>
<!ELEMENT Execution_Date (#PCDATA)>
<!ELEMENT Facility (Access, Activity, BE_Number, Category, Evaluation,
Facility_Name, Facility_ID, Location_Name?, Primary_Mission?,
Relative_Ranking?, Population_Area_Proximity?, Record_Status,
Review_Date, Graphic_Agency, Graphic_Country)>
<!ATTLIST Facility_ID
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "14"
>
<!ELEMENT Facility_ID (#PCDATA)>
<!ATTLIST Facility_Name
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "54"
>
<!ELEMENT Facility_Name (#PCDATA)>
<!ATTLIST Functional_Production_Area
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "5"
>
<!ELEMENT Functional_Production_Area (#PCDATA)>
<!ATTLIST Graphic_Agency
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "15"
```

```
>
<!ELEMENT Graphic_Agency (#PCDATA)>
<!ATTLIST Graphic_Country
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "2"
>
<!ELEMENT Graphic_Country (#PCDATA)>
<!ATTLIST Hardness
      dtype NMTOKEN #FIXED "char"
>
<!ELEMENT Hardness (#PCDATA)>
<!ATTLIST Height
      dtype NMTOKEN #FIXED "float"
>
<!ELEMENT Height (#PCDATA)>
<!ATTLIST Location_Name
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "54"
>
<!ELEMENT Location_Name (#PCDATA)>
<!ELEMENT Mission_ID (#PCDATA)>
<!ATTLIST Mission_ID
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "15"
>
<!ATTLIST Mission_Name
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "30"
>
<!ELEMENT Mission_Name (#PCDATA)>
<!ATTLIST Objective_Name
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "54"
>
<!ELEMENT Objective_Name (#PCDATA)>
<!ATTLIST Operation_Name
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "54"
>
<!ELEMENT Operation_Name (#PCDATA)>
<!ELEMENT Originating_Agency (#PCDATA)>
<!ATTLIST Population_Area_Proximity
      dtype NMTOKEN #FIXED "char"
>
<!ELEMENT Population_Area_Proximity (#PCDATA)>
<!ATTLIST Primary_Mission
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "4"
>
<!ELEMENT Primary_Mission (#PCDATA)>
<!ATTLIST Priority_Objective
      dtype NMTOKEN #FIXED "smallint"
>
<!ELEMENT Priority_Objective (#PCDATA)>
<!ATTLIST Production_Level
      dtype NMTOKEN #FIXED "char"
>
```

```
<!ELEMENT Production_Level (#PCDATA)>
<!ATTLIST Radius
      dtype NMTOKEN #FIXED "float"
>
<!ELEMENT Radius (#PCDATA)>
<!ATTLIST Record_Status
      dtype NMTOKEN #FIXED "char"
>
<!ELEMENT Record_Status (#PCDATA)>
<!ATTLIST Relative_Ranking
      dtype NMTOKEN #FIXED "int"
>
<!ELEMENT Relative_Ranking (#PCDATA)>
<!ATTLIST Release_Mark
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "2"
>
<!ELEMENT Release_Mark (#PCDATA)>
<!ATTLIST Review_Date
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "14"
>
<!ELEMENT Review_Date (#PCDATA)>
<!ELEMENT Target (Affiliation?, Country?, Classification_Level,
Condition, Coordinates, Coordinate_Basis, Coordinate_Derivative,
Date_Created, Date_Last_Change, Hardness?, Height?, Domain_Level,
Elevation?, Elevation_Confidence, Target_Name, Evaluation, Radius?,
Review_Date, Release_Mark?, Facility?)>
<!ELEMENT Target_List (Operation_Name?, Classification_Level,
Date_Created, Date_Last_Change, Domain_Level, Target_List_ID,
Target_List_Status, Target_List_Type, Target_List_Name,
Production_Level, Record_Status, Target+)>
<!ATTLIST Target_List_ID
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "14"
>
<!ELEMENT Target_List_ID (#PCDATA)>
<!ATTLIST Target_List_Name
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "54"
>
<!ELEMENT Target_List_Name (#PCDATA)>
<!ATTLIST Target_List_Status
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "3"
>
<!ELEMENT Target_List_Status (#PCDATA)>
<!ATTLIST Target_List_Type
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "3"
>
<!ELEMENT Target_List_Type (#PCDATA)>
<!ELEMENT Target_Mission (Mission_ID?, Operation_Name?,
Classification_Level, Codeword?, Mission_Name, Target_Objective,
Target_List)>
<!ATTLIST Target_Name
      dtype NMTOKEN #FIXED "string"
```

```
      dsize NMTOKEN #FIXED "54"
>
<!ELEMENT Target_Name (#PCDATA)>
<!ELEMENT Target_Objective (Country?, Execution_Date?,
Functional_Production_Area?, Priority_Objective?, Record_Status,
Domain_Level, Eval, Originating_Agency, Objective_Name)>
```

APPENDIX C (Sample Data XDR)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="http://schemas.biztalk.org/BizTalk/g9boxjl2.xsl"
type="text/xsl"?>
<Schema name="Targetlist-schema" xmlns="urn:schemas-microsoft-com:xml-
data" xmlns:dt="urn:schemas-microsoft-com:datatypes">
    <ElementType name="Access" model="closed" content="textOnly"
dt:type="string" dt:maxLength="9"/>
    <ElementType name="Activity" model="closed" content="textOnly"
dt:type="string" dt:maxLength="3"/>
    <ElementType name="Affiliation" model="closed" content="textOnly"
dt:type="char"/>
    <ElementType name="BE_Number" model="closed" content="textOnly"
dt:type="string" dt:maxLength="10"/>
    <ElementType name="Category" model="closed" content="textOnly"
dt:type="string" dt:maxLength="5"/>
    <ElementType name="Classification_Level" model="closed"
content="textOnly" dt:type="char"/>
    <ElementType name="Codeword" model="closed" content="textOnly"
dt:type="char"/>
    <ElementType name="Condition" model="closed" content="textOnly"
dt:type="string" dt:maxLength="4"/>
    <ElementType name="Coordinate_Basis" model="closed"
content="textOnly" dt:type="string" dt:maxLength="2"/>
    <ElementType name="Coordinate_Derivative" model="closed"
content="textOnly" dt:type="string" dt:maxLength="2"/>
    <ElementType name="Coordinates" model="closed" content="textOnly"
dt:type="string" dt:maxLength="21"/>
    <ElementType name="Country" model="closed" content="textOnly"
dt:type="string" dt:maxLength="2"/>
    <ElementType name="Date_Created" model="closed" content="textOnly"
dt:type="string" dt:maxLength="14"/>
    <ElementType name="Date_Last_Change" model="closed"
content="textOnly" dt:type="string" dt:maxLength="14"/>
    <ElementType name="Date_Last_Changed" model="closed"
content="textOnly" dt:type="string" dt:maxLength="14"/>
    <ElementType name="Domain_Level" model="closed" content="textOnly"
dt:type="string" dt:maxLength="2"/>
    <ElementType name="Elevation" model="closed" content="textOnly"
dt:type="float"/>
    <ElementType name="Elevation_Confidence" model="closed"
content="textOnly" dt:type="i1"/>
    <ElementType name="Eval" model="closed" content="textOnly"
dt:type="char"/>
    <ElementType name="Evaluation" model="closed" content="textOnly"
dt:type="char"/>
    <ElementType name="Execution_Date" model="closed"
content="textOnly" dt:type="string" dt:maxLength="14"/>
    <ElementType name="Facility" model="closed" content="eltOnly"
order="seq">
        <element type="Access" minOccurs="0" maxOccurs="1"/>
        <element type="Activity" minOccurs="1" maxOccurs="1"/>
        <element type="BE_Number" minOccurs="1" maxOccurs="1"/>
```

```
        <element type="Category" minOccurs="1" maxOccurs="1"/>
        <element type="Evaluation" minOccurs="1" maxOccurs="1"/>
        <element type="Facility_Name" minOccurs="1" maxOccurs="1"/>
        <element type="Facility_ID" minOccurs="1" maxOccurs="1"/>
        <element type="Location_Name" minOccurs="0" maxOccurs="1"/>
        <element type="Primary_Mission" minOccurs="0" maxOccurs="1"/>
        <element type="Relative_Ranking" minOccurs="0" maxOccurs="1"/>
        <element type="Population_Area_Proximity" minOccurs="0"
maxOccurs="1"/>
        <element type="Record_Status" minOccurs="1" maxOccurs="1"/>
        <element type="Review_Date" minOccurs="1" maxOccurs="1"/>
        <element type="Graphic_Agency" minOccurs="1" maxOccurs="1"/>
        <element type="Graphic_Country" minOccurs="1" maxOccurs="1"/>
    </ElementType>
    <ElementType name="Facility_ID" model="closed" content="textOnly"
dt:type="string" dt:maxLength="14"/>
    <ElementType name="Facility_Name" model="closed" content="textOnly"
dt:type="string" dt:maxLength="54"/>
    <ElementType name="Functional_Production_Area" model="closed"
content="textOnly" dt:type="string" dt:maxLength="5"/>
    <ElementType name="Graphic_Agency" model="closed"
content="textOnly" dt:type="string" dt:maxLength="15"/>
    <ElementType name="Graphic_Country" model="closed"
content="textOnly" dt:type="string" dt:maxLength="2"/>
    <ElementType name="Hardness" model="closed" content="textOnly"
dt:type="char"/>
    <ElementType name="Height" model="closed" content="textOnly"
dt:type="float"/>
    <ElementType name="Location_Name" model="closed" content="textOnly"
dt:type="string" dt:maxLength="54"/>
    <ElementType name="Mission_ID" model="closed" content="textOnly"
dt:type="string" dt:maxLength="15"/>
    <ElementType name="Mission_Name" model="closed" content="textOnly"
dt:type="string" dt:maxLength="54"/>
    <ElementType name="Objective_Name" model="closed"
content="textOnly" dt:type="string" dt:maxLength="54"/>
    <ElementType name="Operation_Name" model="closed"
content="textOnly" dt:type="string" dt:maxLength="54"/>
    <ElementType name="Originating_Agency" model="closed"
content="textOnly" dt:type="string" dt:maxLength="2"/>
    <ElementType name="Population_Area_Proximity" model="closed"
content="textOnly" dt:type="char"/>
    <ElementType name="Primary_Mission" model="closed"
content="textOnly" dt:type="string" dt:maxLength="4"/>
    <ElementType name="Priority_Objective" model="closed"
content="textOnly" dt:type="i1"/>
    <ElementType name="Production_Level" model="closed"
content="textOnly" dt:type="char"/>
    <ElementType name="Radius" model="closed" content="textOnly"
dt:type="float"/>
    <ElementType name="Record_Status" model="closed" content="textOnly"
dt:type="char"/>
    <ElementType name="Relative_Ranking" model="closed"
content="textOnly" dt:type="i1"/>
    <ElementType name="Release_Mark" model="closed" content="textOnly"
dt:type="string" dt:maxLength="2"/>
```

```
    <ElementType name="Review_Date" model="closed" content="textOnly"
dt:type="string" dt:maxLength="14"/>
    <ElementType name="Target" model="closed" content="eltOnly"
order="seq">
        <element type="Affiliation" minOccurs="0" maxOccurs="1"/>
        <element type="Country" minOccurs="0" maxOccurs="1"/>
        <element type="Classification_Level" minOccurs="1"
maxOccurs="1"/>
        <element type="Condition" minOccurs="1" maxOccurs="1"/>
        <element type="Coordinates" minOccurs="1" maxOccurs="1"/>
        <element type="Coordinate_Basis" minOccurs="1" maxOccurs="1"/>
        <element type="Coordinate_Derivative" minOccurs="1"
maxOccurs="1"/>
        <element type="Date_Created" minOccurs="1" maxOccurs="1"/>
        <element type="Date_Last_Change" minOccurs="1" maxOccurs="1"/>
        <element type="Hardness" minOccurs="0" maxOccurs="1"/>
        <element type="Height" minOccurs="0" maxOccurs="1"/>
        <element type="Domain_Level" minOccurs="1" maxOccurs="1"/>
        <element type="Elevation" minOccurs="0" maxOccurs="1"/>
        <element type="Elevation_Confidence" minOccurs="0"
maxOccurs="1"/>
        <element type="Target_Name" minOccurs="1" maxOccurs="1"/>
        <element type="Evaluation" minOccurs="1" maxOccurs="1"/>
        <element type="Radius" minOccurs="0" maxOccurs="1"/>
        <element type="Review_Date" minOccurs="1" maxOccurs="1"/>
        <element type="Release_Mark" minOccurs="0" maxOccurs="1"/>
        <element type="Facility" minOccurs="0" maxOccurs="1"/>
    </ElementType>
    <ElementType name="Target_List" model="closed" content="eltOnly"
order="seq">
        <element type="Operation_Name" minOccurs="0" maxOccurs="1"/>
        <element type="Classification_Level" minOccurs="1"
maxOccurs="1"/>
        <element type="Date_Created" minOccurs="1" maxOccurs="1"/>
        <element type="Date_Last_Changed" minOccurs="1" maxOccurs="1"/>
        <element type="Domain_Level" minOccurs="1" maxOccurs="1"/>
        <element type="Target_List_ID" minOccurs="1" maxOccurs="1"/>
        <element type="Target_List_Status" minOccurs="1"
maxOccurs="1"/>
        <element type="Target_List_Type" minOccurs="1" maxOccurs="1"/>
        <element type="Target_List_Name" minOccurs="1" maxOccurs="1"/>
        <element type="Production_Level" minOccurs="1" maxOccurs="1"/>
        <element type="Record_Status" minOccurs="1" maxOccurs="1"/>
        <element type="Target" minOccurs="1" maxOccurs="*"/>
    </ElementType>
    <ElementType name="Target_List_ID" model="closed"
content="textOnly" dt:type="string" dt:maxLength="14"/>
    <ElementType name="Target_List_Name" model="closed"
content="textOnly" dt:type="string" dt:maxLength="54"/>
    <ElementType name="Target_List_Status" model="closed"
content="textOnly" dt:type="string" dt:maxLength="3"/>
    <ElementType name="Target_List_Type" model="closed"
content="textOnly" dt:type="string" dt:maxLength="3"/>
    <ElementType name="Target_Mission" model="closed" content="eltOnly"
order="seq">
        <AttributeType name="xmlns" dt:type="string"/>
        <attribute type="xmlns"/>
```

```
        <element type="Mission_ID" minOccurs="0" maxOccurs="1"/>
        <element type="Operation_Name" minOccurs="0" maxOccurs="1"/>
        <element type="Classification_Level" minOccurs="1"
maxOccurs="1"/>
        <element type="Codeword" minOccurs="0" maxOccurs="1"/>
        <element type="Mission_Name" minOccurs="0" maxOccurs="1"/>
        <element type="Target_Objective" minOccurs="1" maxOccurs="*"/>
        <element type="Target_List" minOccurs="1" maxOccurs="*"/>
    </ElementType>
    <ElementType name="Target_Name" model="closed" content="textOnly"
dt:type="string" dt:maxLength="54"/>
    <ElementType name="Target_Objective" model="closed"
content="eltOnly" order="seq">
        <element type="Country" minOccurs="0" maxOccurs="1"/>
        <element type="Execution_Date" minOccurs="0" maxOccurs="1"/>
        <element type="Functional_Production_Area" minOccurs="0"
maxOccurs="1"/>
        <element type="Priority_Objective" minOccurs="0"
maxOccurs="1"/>
        <element type="Record_Status" minOccurs="1" maxOccurs="1"/>
        <element type="Domain_Level" minOccurs="1" maxOccurs="1"/>
        <element type="Eval" minOccurs="1" maxOccurs="1"/>
        <element type="Originating_Agency" minOccurs="0"
maxOccurs="1"/>
        <element type="Objective_Name" minOccurs="1" maxOccurs="1"/>
    </ElementType>
</Schema>
```

LIST OF REFERENCES

[ABI00] Abiteboul, S., Buneman, P., and Suciu, D., *Data on the Web: From Relations to Semistructured Data and XML.* Morgan Kaufmann Publishers, 2000.

[BOS99] Bosworth, A., Layman, A.,and Rys, M., "Serializing Graphs of Data in XML," http://www.biztalk.org/Resources/canonical.asp, Microsoft Corporation, 1999.

[BOU99] Bourret, R., "XML and Databases," Technical University of Darmstadt, http://www.informatik.tu-darmstadt.de/DVS1/staff/bourret/xml/XMLAndDatabases.htm, September 1999.

[BOX00] Box, D., et al., "SOAP: Simple Object Access Protocol," http://msdn.microsoft.com/xml/general/soapspec.asp, March, 2000.

[BRA98] Bray, T., "The Annotated XML Specification," http://www.xml.com/axml/axml.html, September 1998.

[BUC00] Buck, L., "Modeling Relational Data in XML," Extensibility, Inc., http://www.extensibility.com/xml_resources/modeling.htm, March 2000.

[CAR00] Carnevale, R., "The Joint Common Database," February 2000.

[DAT00] IBM, "Datajoiner," document available online at http://www.software.ibm.com/data/datajoiner.

[DAV99] David, M., "SQL-based XML Structured Data Access," Web Techniques, San Francisco, June 1999.

[DOM00] Wood, L., et al., "Document Object Model (DOM) Level 1 Specification," W3C Working Draft, http://www.w3.org/TR/2000/WD-DOM-Level-1-20000929/, September 2000.

[DTS00] Microsoft, "Data Transformation Services Considerations," document available online at http://msdn.microsoft.com/library/psdk/sql/dts_adv_15.htm

[FGM98] "Functional Description Document: Track and Relational Data Synchronization", Version 1.0, FGM Inc., August 1998.

[FLO99] Florescu, D. and Kossmann, D., "A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database," The French National Institute for Research in Computer Science and Control, May 1999.

[GCC98] "Global Command and Control System (GCCS) Application Program Interface Reference Manual (APIRM) for NIPS Developers Segment (NDEV)," Version 3.1.1.0, prepared for the Defense Information Systems Agency, April 1998.

[HAS00] Hayes, G., Pipher, J., and Davis, S., "Joint Common Catalog (JCC): Refined Concept and Implementation Status," *Proceedings for the Federal Database Colloquium and Exposition*, September 2000.

[HOD00] Hodges, A. and Buck, T., "Defense Information Infrastructure (DII) Common Operating Environment (COE) Data Access Services," *Proceedings of the Federal Database Colloquium and Exposition*, September 2000.

[INR00] "Implementation Guidelines for Interoperability with the GCCS-COP for JWID 2000", Inter-National Research Institute, Inc., May 2000.

[ISO86] "Information Processing – Text and Office Systems – Standard Generalized Markup Language (SGML)," ISO 8879, 1986.

[ISO97] "Framework for the Specification and Standardization of Data Elements," http://pueblo.lbl.gov/~olken/X3L8/drafts/draft.docs.html, 1998.

[JBM00] Myers, R. and Brunn, B., "Joint Battle Management Initiative (JBMI) Assessment Plan Draft," August 2000.

[JDB00] JavaSoft, "JDBC Data Access API," document available online at http://www.javasoft.com/products/jdbc/index.html.

[LAY98] Layman, A. et al., "XML-Data," World Wide Web Consortium (W3C) Note, http://www.w3.org/TR/1998/NOTE-XML-data-0105/, January 1998.

[MID98] "Modernized Integrated Database: Database Design Document," Defense Intelligence Agency, June 1998.

[MTF00] "XML-MTF Mapping, Third Public Working Draft," XML-MTF Development Team, August 2000.

[NRC99] "Realizing the Potential of C4I: Fundamental Challenges," Committee to Review DOD C4I Plans and Programs, National Research Council, 1999.

[ODB95] Microsoft, "ODBC Related Standards", http://ourworld.compuserve.com/homepages/VBrant/stds.htm, 1995.

[OMF00] "Weather Observation Definition Format (OMF)," http://zowie.metnet.navy.mil/~spawar/JMV-TNG/XML/OMF.html, Space and Naval Warfare PMW-185, May 2000.

[QUI00] Quin, L., *Open Source XML Database Toolkit: Resources and Techniques for Improved Development*, Wiley Computer Publishing, August 2000.

[REN96] Renner, S. and Scarano, J., "Migrating Legacy Applications to a Shared Data Environment," *Proceedings of the Federal Database Colloquium and Exposition*, August 1996.

[ROS00] Rosenthal, A., Frank, M., and Renner, S., "Getting Data to Applications: Why We Fail, How We Can Do Better," *Proceedings of the Federal Database Colloquium and Exposition 2000*, September 2000.

[ROS97] Rosenthal, A., Edward, S., and Renner, S., "Toward Unified Metadata for the Department of Defense," *IEEE Metadata Workshop*, April 1997.

[SCH00] Schneider, J., Cherinka, R., Cokus, M., and Molloy, M., "TBMCS-JBI Using XML-MTF Information Objects: An XML Technology Report for the Joint Battle Management Integration (JBMI) Assessment," The Mitre Corporation, January 2000.

[SYA00] Sybase Inc., "Adaptive Server Enterprise 12.0," document available online at http://www.sybase.com/products/databaseservers/ase/.

[SYB00] "Using XML with the Sybase Adaptive Server SQL Database," Technical Whitepaper, Sybase Inc., January 2000.

[WAP99] "WAP Binary XML Content Format," http://www1.wapforum.org/tech/documents/SPEC-WBXML-19991104.pdf, November 1999.

[XDB00] Bourret, R., "XML-DBMS," document available online at http://www.rpbourret.com/xmldbms/index.htm.

[XML98] "Extensible Markup Language 1.0," W3C Recommendation, http://www.w3.org/TR/REC-xml, February 1998.

[XQL00] "XML Query Requirements," W3C Working Draft, http://www.w3.org/TR/xmlquery-req, August 2000.

[XQL98] Ishikawa, H., Kubota, K., Kanemasa, Y., "XQL: A Query Language for XML Data," http://www.w3.org/TandS/QL/QL98/pp/flab.txt, November 1998.

[XSL99] "XSL Transformations," World Wide Web Consortium (W3C), http://www.w3.org/TR/xslt, November 1999.

NPS SE Ph.D. Program
22 December, 2000

Dissertation Proposal
CAPT Paul E. Young

I. Title. Integration of Heterogeneous Software Systems Through Computer-Aided Resolution of Data Representation Differences

II. Goals and proposed new contribution.

A. Introduction. Past acquisition and development practices in the Department of Defense (DoD) have led to the procurement of numerous special purpose, non-interconnected software-intensive systems for application areas varying from embedded weapon system software to logistic management systems. Advances in computer communications technology, the recognition of common areas of functionality in related systems, and an increased awareness of how enhanced information access can lead to improved capability, are driving an interest toward integration of current stand-alone systems to meet future system requirements. In addition, the integration of Commercial Off-the-Shelf Software (COTS) and Government Off-the-Shelf Software (GOTS) with existing legacy systems offers an attractive alternative for enhancing the capabilities of these systems without incurring the expense and time required for a new software development.

A prime difficulty in achieving interoperability among heterogeneous components of a composite system is that the component systems were developed independently, without any requirement for interoperability. Thus systems have different architectures, different hardware platforms, different operating systems, different host languages and different data representations. Short of redeveloping a new system using the consolidated requirements from the various component systems and a common architecture, hardware platform, operating system, host language, etc. (a cost prohibitive approach), a means must be devised to achieve the goal of component interoperability in the face of expected limited acquisition budgets.

One major impact of the independent development of system components targeted for integration is the potential for differences in representation of data shared between the systems. These representation differences can be in the form of different physical representations, accuracy tolerances, range of values allowed, terminology used, structural representations, and methods for capturing data semantics. [KM98] In addition, there must be a logical mapping between data elements of the involved systems to ensure that the context in which data is referred is the same on all interoperable systems.

To overcome the impediments identified above, we propose to explore ways to answer the following question in a positive way. Given N heterogeneous systems, can we resolve the differences in data representation and ensure consistency in data mapping to

enable interoperability between the systems? The research goal is to provide an automated and / or computer-aided methodology to aid in the resolution of data representational differences between systems targeted for integration in order to enable system interoperability.

B. Significance of the problem and its potential impact.
Integration of heterogeneous legacy systems is currently an essentially manual, labor intensive, costly evolution. The ability to automate part or all of this integration process holds the promise of providing enhanced capability at significant time and cost savings. In addition, the same methodology for integrating heterogeneous legacy systems can be applied to the integration of COTS and GOTS components with existing systems to enhance their capability while minimizing cost, an attractive possibility for this era of continuously shrinking defense budgets.

C. Proposed advances to the state-of-the-art.
Current state-of-the-art for integration of heterogeneous systems involves manually resolving differences in data representation and mapping for each interface between systems, in an inherently customized manner. The first step in advancing the state-of-the-art is to develop a general formal model that captures the attributes of the data elements comprising the interface and the operations required to reconcile data element differences between the interfaced systems. The formal model can then be used as follows. First, the model can be used to instantiate a specific instance of the model for the systems being integrated. Then, the model's instantiation will serve as the basis for automating the process for resolving representational differences.

We will explore ways of using this common structure to provide computer aid. Automation support of heterogeneous system integration has the potential to significantly enhance the integration process and should lead to significant savings in time and cost. Potential areas for automation include:

- identification of data types and elements comprising the interface between systems,
- assistance in identifying component system data elements that represent the same real-world object,
- assistance in defining the operations required to resolve differences in data representation when the implementation of a real-world object varies between the component systems, and
- providing data conversion between component systems where differences in data representation occur, using the operations defined above.

In summary, the proposed advances to the state-of-the-art include:

- definition of a general formal model for capturing the relationships between data elements shared between components in a federated system and for identifying the operations required to resolve representational differences between data elements where they exist,
- defining an approach for discovering related data elements on different component systems, and
- automation of the process for resolving data representational differences between related data elements.

III. Research strategy and proposed approach.

A. Proposed approach.

System integration as producer-consumer relationship

Before contemplating the integration of two or more heterogeneous systems, one should be able to answer the question of why should the selected systems be interconnected. The obvious answer to this question is that two systems should be connected only if one system has data that is of interest to another or if there exists the potential for this to occur. This requirement to share data defines a producer-consumer relationship between a system that provides the data (the producer) and a system that uses that data (the consumer). It may be the case that a system is both a data provider and a data user. In this situation, a separate producer-consumer relationship is defined for each data element concerned.

Legacy system integration focus

The focus of my research is on methods for providing computer aid to the integration of existing heterogeneous systems. By contrast, design of a new system from a consolidated set of requirements should not require such methodologies to achieve data interchange if properly designed from the start. The focus on legacy systems in my research results in additional restrictions that must be taken into consideration.

Foremost among these restrictions is the desire that the proposed methodology eliminates or at least minimizes any requirement for modification of the underlying legacy system code. Legacy system modification, whether of COTS or GOTS products, is a difficult, costly, and oftentimes impossible proposition. This restriction leads to a focus on the existing external systems interfaces for the integration of legacy software. Thus, my primary focus will be on reconciling the representational differences between data elements that are contained in the external interfaces of systems to be integrated. The obvious limitation to this approach is the reality that even though a system may contain a data element that may be of interest to another system, if both systems' external interfaces do not provide access to that data element, the interchange cannot occur. In this instance, for integration to proceed, modification of one or both of the systems' external interfaces would be required to provide access to the desired data.

Two-phased integration process

The goal of enabling data interchange between systems is the attainment of a federation of cooperating, autonomous, heterogeneous software systems, akin to the Federated Database Systems (FDBS) specified in [HMS94]. Achievement of such a federated system is accomplished using a two-phased process. In the first phase, accomplished prior to runtime, a formal model for capturing the relationships between producer and consumer data elements is defined. In the second phase, the model developed prior to runtime is used to automatically translate between heterogeneous data representations.

Pre-runtime phase

In the pre-runtime phase, a model for establishing relationships between producer and consumer types is defined. This model, termed a Consolidated Type Hierarchy (CTH),

utilizes an object-based model to capture relationships between data elements in producer and consumer systems, translations required to convert between different representations used by a producer and consumer system, and information used to establish relationships between producer and consumer data elements.

The CTH model consists of three basic object types, as depicted in Figure 1. The first object type, the ProducerType, contains the producer's view of an exported data element in the form of a ProducerTypeSchema. Similarly, a ConsumerType contains the consumer's view of a data element being imported in the form of a ConsumerTypeSchema. In order for the interchange of data between a producer and a consumer to have meaning, the exported data element and the import data element must both be representations of the same real-world object. Any differences in the representation of the real-world object are resolved by introduction of a third object type in the CTH, termed a ConsolidatedType, which provides a "standard" or normal representation for the real-world object, as captured in the ConsolidatedTypeSchema. In a sense, two or more different representations of the same real-world object are *consolidated* into a standard ConsolidatedType that is added to the CTH.

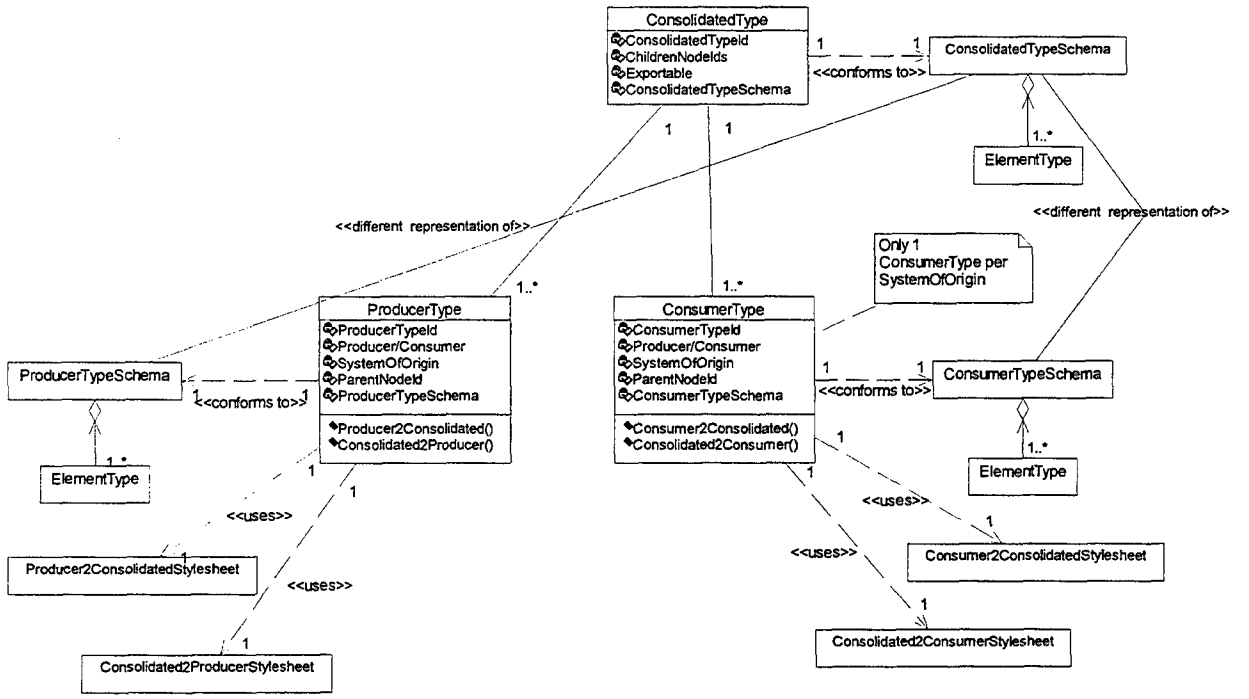

Figure 1

The ConsolidatedType maintains a *representative of* relationship with the producer and consumer types that depict the same real-world object. The *representative of* relationship is defined to mean that the ConsolidatedTypeSchema contains a representative of each of the elements contained in every related ProducerTypeSchema and ConsumerTypeSchema. The ConsolidatedType representation can be a mirror of either the ProducerType or the ConsumerType representation or it can be different from either.

The key consideration is that the ConsolidatedType representation be of sufficient precision that the meaning and precision of the data is preserved when converting between different representations.

There can exist a one-to-many relationship between both a ConsolidatedType and a ProducerType and between a ConsolidatedType and a ConsumerType. However, limitations in the run-time translation architecture may restrict the cardinality of this relationship. Allowing a one-to-many relationship between a ConsolidatedType and a ProducerType enables composition of data elements from multiple producers in response to a consumer request.

In addition to capturing the relationships between producer and consumer system data elements, the CTH contains translations required to convert a producer's representation of an object to the consumer's representation. This translation is accomplished in two steps. First, the producer's representation is converted to the consolidated type representation. Then, the consolidated type representation is converted to the consumer's representation. These translations are captured as part of the consumer and producer type objects.

Finally, the CTH contains attributes used to determine the data type relationships between consumer and producer systems. These attributes include both structural and semantic information about a data element. This information is used to bring computer aid to solving the problem of establishing whether two data elements are representations of the same real-world object.

Use of the eXtensible Markup Language (XML) for implementing the CTH
In order to assist the data transfer and conversion process, use of a standard method for representation of the abstract CTH model is proposed. One possible representation, the eXtensible Markup Language (XML) offers a mechanism to separately identify the elements in the abstract data model along with methods required for implementing the conversion process.

XML is a markup language used for describing documents that contain structured information. Structured information contains both content, generally referred to as data, and a description of what role the content plays in the document. A markup language provides a mechanism for capturing the structure in a document. Markup, in the form of "tags", serves both to delimit the data contents and provide descriptive information about data in an XML document. The XML specification defined in the World Wide Web Consortium (W3C) Extensible Markup Language (XML) 1.0 Recommendation of 10 February 1998 (Second Edition dated 6 October 2000) defines a standard way to add markup to documents. [Wal98] [BPS98]

The CTH model is represented as an XML document, with ConsolidatedTypes, ProducerTypes, and ConsumerTypes represented as XML elements. XML elements are delimited using tags and serve as the basic building blocks of an XML markup. Tags

consist of an element type name and specific characters used to distinguish the element type name, with each element requiring a start-tag and an end-tag.

The data elements represented by the ConsolidatedTypes, ProducerTypes, and ConsumerTypes are extracted from the external interfaces of the systems being integrated and conform to an XML schema defining the interface. An XML schema uses a set of rules expressed using XML syntax to specify the structure and permissible values of XML documents. An XML document that conforms to the syntactic rules of XML is said to be *well-formed* XML. A document that also conforms to the vocabulary defined in a referenced XML schema is considered to be *valid*. Development of the CTH model assumes that the XML schema for the produced and consumed types is available for use in constructing the CTH document. If the external interface for the component systems to be integrated is not defined using XML, then translation from the existing format used by the component system, such as USMTF, OTH-T Gold, etc., to XML will be required.

Messages translated by a producer system or received by a consumer conform to the XML schema that defines the allowable data elements for the producer and consumer external interfaces, respectively.

The first function of the CTH model, capturing the relationships between producer, consumer, and consolidated types, is accomplished through the use of XML's capability to define a hierarchical structure for the CTH document or by features of the XML schema that allow links between document elements.

The second aspect of the CTH model function, capturing the translations required to transform a data element from a producer to a consumer representation, is accomplished through the use of eXtensible Stylesheet Language Transformation (XSLT) stylesheets. As stated in W3C references, XSLT "is a language for transforming XML documents into other XML documents". [http://www.w3.org/TR/xslt11/] In the CTH application, an XSLT stylesheet defines the translations required to convert from a producer to a consumer representation as a sequence of data element attribute mappings and functional transformations. Given a producer message and the appropriate XSLT stylesheets, the proposed translator will invoke an XSLT engine to transform the data element into the proper consumer representation.

Finally, structural and semantic information about a data element that can be used to determine the data type relationships between consumer and producer systems can be stored in the CTH as part of the ConsumerType and ProducerType elements. Additional relational information, such as might be provided by a common ontology from which to map component system data elements could be modeled outside the CTH with reference to the appropriate terms in the ontology provided as part of the ConsumerType and ProducerType elements.

Consolidated Type Hierarchy development

The Consolidated Type Hierarchy is constructed for a federation of heterogeneous systems from the data elements defined in each system's external interface. Construction

of the CTH is an incremental process involving the following computer-aided human activities: 1) *Registration*, 2) *Discovery*, 3) *Consolidation*, and 4) *Reconciliation*. *Registration* provides the means for adding producer and consumer data elements to the Consolidated Type Hierarchy. Registration utilizes the *Discovery* process to assist the system designer in determining whether there are any producer data elements relevant to a consumer element being registered. Once determined by the system designer that a producer and consumer data element are both representations of the same real-world object, the *Consolidation* process establishes the required relationships between the producer and consumer objects. Finally, in the *Reconciliation* process, the system designer is aided in defining the mapping and translation functions necessary for reconciling representational differences between producer and consumer types.

Data element Registration begins with the producer data elements. These are elements that are identified in a system's external interface as being exported. This segment of the Registration process is straightforward- the component system's external interface is searched to find those elements that the system will export. These are incrementally added to the CTH document as ProducerType objects.

Registration of consumer elements follows producer element registration. The first step in consumer element registration is determining whether there are any producers of the data elements being requested for import by the consumer. The system designer is aided in this determination by the Discovery process. The Discovery process utilizes information about data elements to determine whether two types are different representations of the same real-world object and therefore can be used to share information between the two systems being integrated. Methodologies for making this determination can be separated into three categories: information retrieval approaches, methods using structural information about a data element, and approaches using semantic information about an element.

Information retrieval approaches, such as used by traditional information retrieval systems and by web-based search engines, rely on such methods as keyword matching, subject classification, and term relationships to find information relevant to a retrieval request. These approaches, while useful, have their limitations for solving the Discovery problem. Keyword matching is limited by different systems' use of synonyms and homonyms in naming data elements. In using synonyms, different systems can use different names for data elements that represent the same real-world object. Conversely, using the same name to represent different objects, such as use of the term "fire" to represent both a burning object and a weapon's discharge, is an example of homonym use. Lack of a canonical naming scheme by component systems reduces the effectiveness of keyword matching approaches for solving the Discovery problem. In subject classification approaches, a set of pre-defined subject terms would be used to categorize data objects. The use of subject classification tends to be difficult to administer. Subject categories must be defined and objects categorized according to this definition. The number of categories required to integrate component systems can be quite large, even if done on a domain-specific basis. Building term relationships from existing documents, such as thesaurus-group generation, concept networks for concept

retrieval (vice keyword retrieval), and latent semantic indexing by singular value decomposition, have similar limitations to the keyword matching and subject classification approaches discussed above. [KM98]

Other methods use structural information about a data element to determine whether two elements are related. This structural information can include 1) relationships between an element being compared and other elements from the same system, 2) the meaning and resemblance of an element's attributes, and 3) the similarities of elements based on the percentage of occurrences of common attributes. The relationship of an element to other elements in the same system can be used for comparing whether a producer and a consumer represent the same real-world object. Whether an element is a complex object or part of a complex object, whether an element is part of a subtype to supertype relationship, or what methods might be defined for the element may all be used to compare producer and consumer elements. As noted by Garcia-Solaco, Saltor, and Castellanos, "two classes are (to be) integrated only if they are similar and the specializations in which they participate as subclasses are similar as well." [GSC95, p.512] The meaning of an element's attributes can be approximated in terms of its type, cardinality, integrity constraints and allowable operations. Then based on this meaning, two elements can be compared to see if they are equivalent. Finally, heuristics can be used to determine the similarity of objects based on the percentage of occurrences of common attributes between the objects. [HMS94]

Finally, the most promising yet most challenging methods for determining whether two elements are related use semantic information about the elements. Semantic information methods include 1) methods that define export object characteristics in terms of a common ontology, 2) methods that capture the behavior of objects using predicate logic, 3) knowledge-based systems, 4) use of pre- and post-conditions in the specification of a type's methods to enable comparison of behaviors of two types, and 5) path based methods. See [HM99] and [Sin98] for more details.

The goal of my research is to find an appropriate engineering solution to the Discovery process that provides a filtered list of potential producer elements matching a consumer element registration request. The envisioned solution will either provide a unique approach for providing the match or use some combination of the above approaches to assist the system designer in matching consumer and producer elements.

The Consolidation process is used to unite a consumer object with a producer object satisfying the consumer request. This is accomplished through the addition of a ConsolidatedType to the CTH document for each producer-consumer pairing identified during the Registration and Discovery processes. The relationship between the consolidated type and producer and consumer types defines a tuple of the form {$ProducerType_1$, ... $ProducerType_n$, ConsolidatedType, $ConsumerType_1$ .. $ConsumerType_m$} and is used to establish links between the ConsolidatedType and the appropriate ProducerType and ConsumerType nodes in the CTH.

As the last step in defining the CTH for the federated system, the Reconciliation process is used to add the mapping and translation functions required to convert between data element representations to the CTH document. Separate mappings/ translations are required to convert between a producer representation and the respective consolidated type representation and from the consolidated type representation to the appropriate consumer type representation.

CTH development environment

One of the benefits of the CTH model is that it readily supports application of computer aid to building a CTH document that defines a specific federation of component systems. It is expected that computer aid can be applied in the following areas:

a) registration of producer and consumer types,
b) discovery of produced type(s) satisfying a consumer type request,
c) creation of consolidated types as the canonical representation of a producer-consumer relationship, capturing the relationship between consolidated, producer, and consumer types, and
d) development of translations to convert a producer representation of a data element to its appropriate consumer representation.

For producer and consumer type registration, it is envisioned that the CTH development environment will utilize the schemas for a component's external interface to provide a graphical representation of the exported and imported types to the system designer. Then, using a "click-to-select" approach, as a first step the designer will select those export types to be added to the CTH document. Based on the designer's selection, the development environment will automatically register the selected export type as a ProducerType and add it to the CTH XML document.

Upon completion of export type registration, the designer will select the import types to be registered. For each import type selected, the discovery process will provide a rank-ordered list of producer types that are candidate alternative representations for the import type. The designer can then select from one of the provided alternates or conduct a manual search of the entire set of producer types to designate a producer/consumer pairing.

For each designated producer/consumer pairing, the CTH development environment will create a ConsolidatedType to provide the "standard" representation of the real-world object captured by the producer and consumer types. The development environment will assist the designer in defining the sub-elements and attributes for the ConsolidatedType and defining the ConsolidatedType to ProducerType and ConsolidatedType to ConsumerType relationships at the sub-element and attribute level. These relationships will provide a mapping between sub-elements and attributes and may include functional transformations to convert between different representations, such as miles to kilometers. The development environment will automatically record these relationships in the CTH XML document.

The relationships defined in the previous paragraph provide the basis for the automatic generation of the XSL stylesheets needed by the ProducerToConsolidated and ConsolidatedToProducer operations to convert from a producer's representation of a data instance to a consumer's. These stylesheets can be automatically generated using the previously defined relationships between producer, consolidated, and consumer types, along with the XML schemas for these types.

Runtime phase

The Consolidated Type Hierarchy (CTH) document constructed during the pre-runtime phase for a specified federation of component systems is used to resolve the data representational differences between data elements from the different systems. Reconciling representational differences is accomplished at runtime by a *translator* that serves as an intermediary between component systems.

The translation function is anticipated to be implemented as part of a *software wrapper* enveloping a producer or consumer system (or both) in a message-based architecture, or as part of the data store (actual or virtual) in a publish/subscribe architecture. A software wrapper is a piece of software used to alter the view provided by a component's external interface without modifying the underlying component code.

For either type of architecture, the function of the translator is similar, as is indicated in Figure 2. In both cases the decision of which producer type should be linked to which consumer type and what translations are required to convert an instance of the producer type representation to an instance of the consumer representation, is determined by the CTH. The resolution of which producer types can be used as a data source for which consumer types is determined by the existence of a producer-consumer tuple [ProducerType(s)-ConsolidatedType-ConsumerType(s) pairing] in the CTH. Only when there is producer-consumer tuple in the CTH can the specified producer be a data source for the matching consumer. This is an important concept for insuring that only those producer-consumer relationships specified by the system designer can be used to implement a translation. If such a tuple exists in the CTH, then the ProducerType, ConsolidatedType, and ConsumerType components of the tuple define nodes of a tree rooted at the ConsolidatedType. Traversal of this tree from the ProducerType node to the ConsolidatedType node defines a translation path to convert an instance of the producer type to an instance of the consumer type. The ProducerToConsolidated and ConsolidatedToConsumer operations associated with the ProducerType and ConsolidatedType nodes, respectively, define the required translations along the path.

The primary difference between a message-based architecture and a publish/subscribe architecture is the location of the translator functionality. In a message-based architecture, the translator functionality can be incorporated into a software wrapper enveloping the producer system, the consumer system, or both. In a publish/subscribe architecture, the translator functionality would be realized as part of the data store implementation.

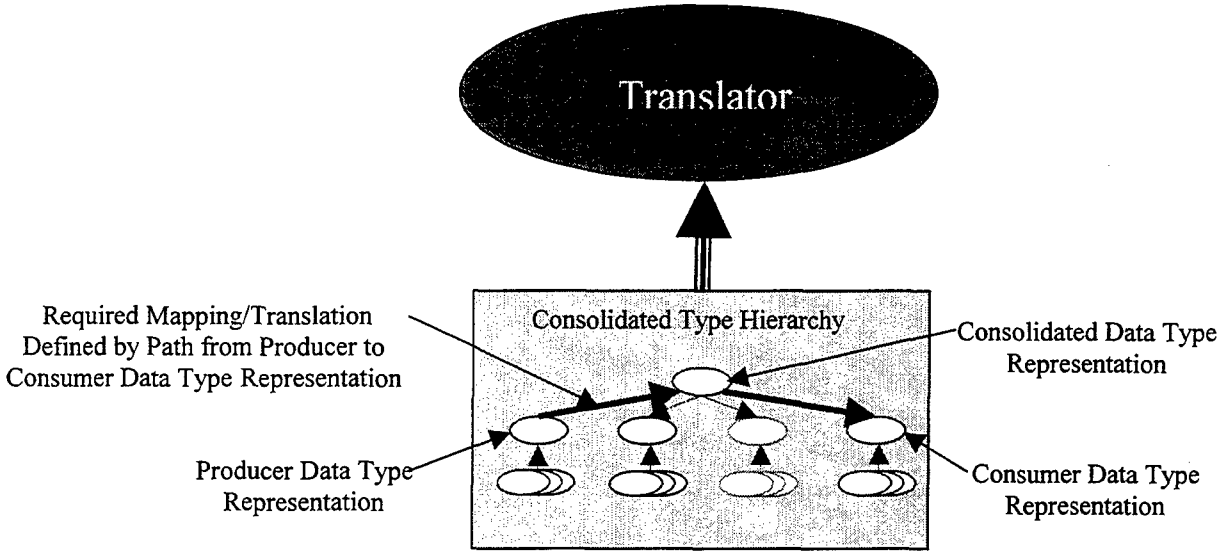

Figure 2

Translator function.
A message-based architecture with translator functionality located in both the producer and consumer system wrappers is selected for illustration in Figure 3 and for explanation as follows. A producer system constructs an outgoing message conforming to the allowable message set grammar provided for the system's external interface. The allowable message grammar is specified in the form of an XML schema for the particular system's external interface. The producer system may optionally validate the outgoing message against the XML schema through the use of an XML parser. Upon transmission, the wrapper encapsulating the producer intercepts the outgoing message.

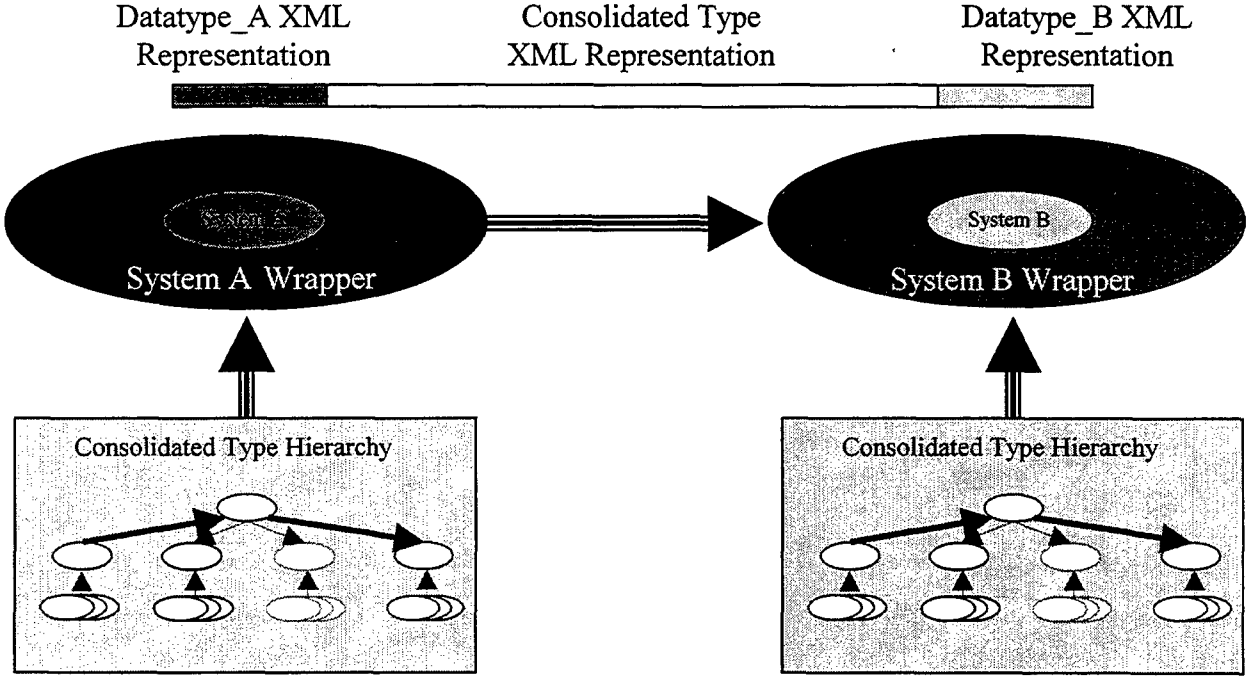

Figure 3

The embedded translator parses the outgoing message to determine the message's constituent types. For each type instance, the translator locates the indicated producer type node in the CTH. The translator then uses the ProducerToConsolidated operation associated with that type instance to convert the instance to an instance of the corresponding consolidated type. This process is repeated for each data element instance contained in the transmitted message. The resultant message is then forwarded to the consumer system.

On the consumer system, the above process is executed in reverse. The wrapper encapsulating the consumer captures the incoming message. The embedded translator parses the incoming message to determine the message's component types. For each type instance, the translator locates the indicated consolidated type in the CTH. From the consolidated type, the translator traverses the CTH to locate the corresponding consumer type for the destination system. The translator then uses the ConsolidatedtoConsumer operation associated with the consumer type to convert the instance to an instance of the consumer type. This process is repeated for each consolidated type instance contained in the incoming message. The translated message is then forwarded to the consumer system for utilization.

Implementing the translator functionality on both the producer and consumer systems provides an advantage over either a producer-end only or consumer-end only implementation. That advantage is due to the isolation of the impact of changes to a component system to only the effected system and its accompanying wrapper. Changes to one system do not require other systems or their wrappers to be changed. Another potential advantage is that only a system-specific subset of the Consolidated Type Hierarchy translation operations need be implemented for each component system wrapper. Only those operations required to translate to or from the types contained in the system's external interface, as specified by the XML schema for the interface, are required to be implemented for that system.

The operation of the translator in a publish/subscribe architecture is similar to the operation in a message-based architecture described above. The primary difference is in where the translation functionality is implemented. In a publish/subscribe architecture the most logical location for implementation of the translation functionality is as part of the data store. This data store may be a physical store with an integrated data management capability, or it may be a virtual store where the data management function is provided by some arbitrator with actual data storage remaining on the producer systems. The virtual store approach is akin to the method implemented for the world wide web where an internet browser provides the data management function and actual data storage is maintained on host producer systems.

As illustrated in Figure 4 below, translator operation in a publish/subscribe architecture incorporating a physical data store is envisioned to occur as follows. Data publishers export data in the publishers' native format to the physical data store. If the publisher does not export data using an XML message representation, a wrapper around the publisher system can be used to convert the exported data into an XML representation of the native format. Data received by the data store is retained in the XML representation

of the publisher's native format. Data subscribers transmit a request for data to the data store in the subscriber's native format. Again, if the subscriber does not utilize XML for its data representation, a wrapper around the subscriber can convert the data request into its equivalent XML representation.

Just as with the message-based system, a Consolidated Type Hierarchy document would have been constructed prior to run-time defining the allowable producer/consumer data type relations. Upon receipt of a subscription request the translator, implemented as part of the physical store's data handler, would locate the data type of the subscription request in the CTH, determine the producer type that corresponds to the request type, and issue a request to the data handler for the published data using the XML representation of the producer's native format. Upon locating the requested published data, the translator uses the ProducerToConsolidated operation associated with the producer type to convert the instance to an instance of the corresponding consolidated type. From the consolidated type, the translator traverses the CTH to locate the corresponding consumer type of the subscription request. It then uses the ConsolidatedtoConsumer operation associated with the consumer type to convert the instance to an instance of the consumer type. The translated data instance is then forwarded to the system that issued the subscription request.

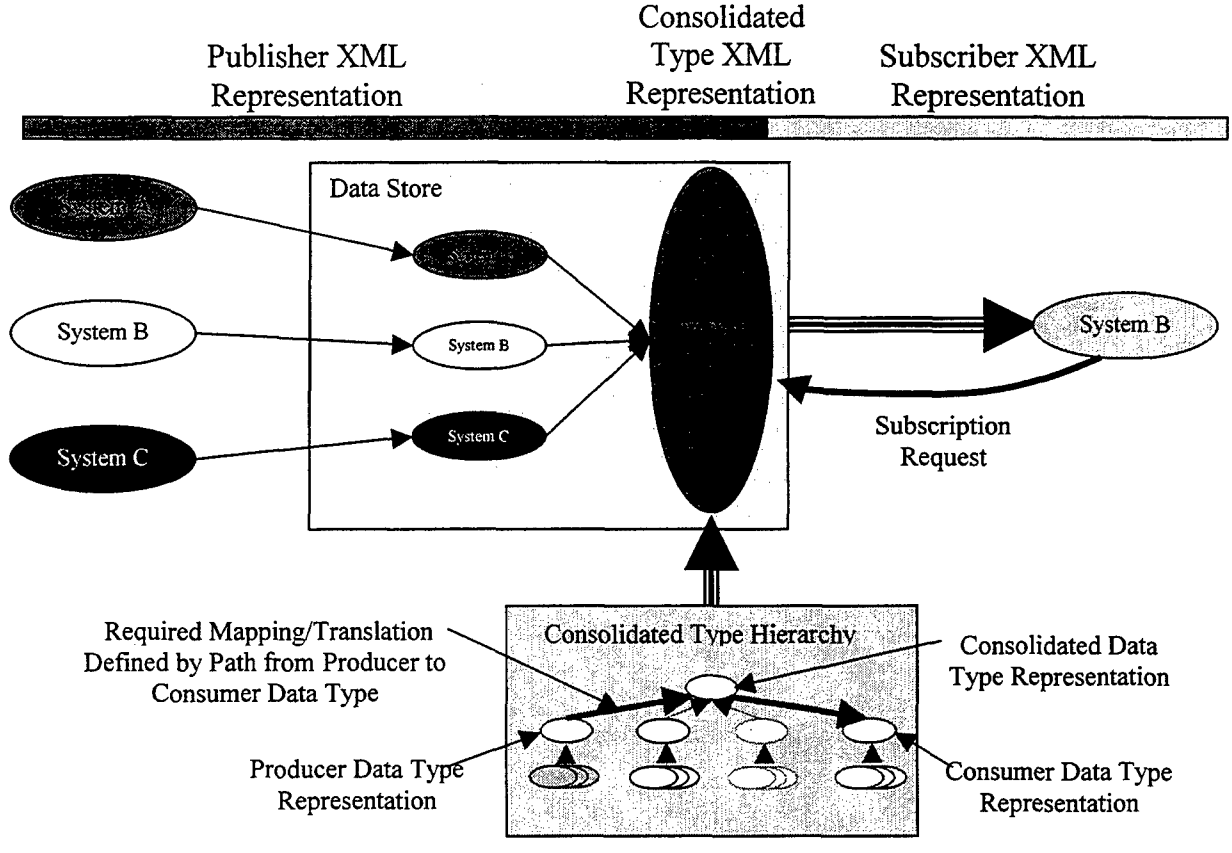

Figure 4

Conclusion

In summary, the Consolidated Type Hierarchy model serves as the foundation for automating a process for resolving data representation differences between autonomous, heterogeneous software systems. From the CTH model, a federation-specific hierarchy is developed for the included component systems. Computer aid is applied in the development of this hierarchy to assist the system designer in locating relevant data producers and consumers and in defining the translations required for resolving data representation differences between systems. Finally, the resulting producer-consumer relationships and translation definitions are used to automate resolution of data representational differences in the federation.

B. Methods to substantiate new contributions including proposed experiments, measurements or theoretical analysis.
   1. Discovery algorithm efficiency and effectiveness
      a. Measurement of precision and recall of results obtained from computer-aided discovery of produced types satisfying consumer type request.
   2. CTH development environment effectiveness
      a. Measurement of percent of stylesheet generated automatically versus percent requiring manual construction.
   3. Translator correctness
      a. Use of XML parser to validate result of translation against consumer type schema.

C. Expected delivery of products, if any.
   - Specification of a general formal model to be used for the resolution of data representational differences between heterogeneous systems.
   - Architecture and implementation for a development environment used to construct Consolidated Type Hierarchy for a federated system.
   - Architecture and implementation for a messaging system translator.

IV. Assessment of previous work.

[WO00] "Respectful Type Converters"

Summary

In converting objects of one type to another, Wing and Ockerbloom introduce the concept of *respects* to describe how a converter C may retain some of the original behavior of an object of type A when converting it to some other type B, denoted C: A → B, while some of A's behavior is changed. The information that is preserved during the conversion is given in terms of another object of type T and the conversion is said to *respect* type T "if the original object of type A and the converted object of type B have the same behavior when both objects are viewed as a type T object".

Wing and Ockerbloom's definition of the *respects* relationship exploits the Liskov and Wing notion of behavioral subtyping to show that type T reflects the preserved behavior in a conversion from type A to type B. Under behavioral subtyping, "if S is a subtype of T, users of T objects cannot perceive when objects of type S are substituted for T objects". Thus Wing and Ockerbloom point out that for a converter C: A → B to respect a type T, it is

necessary that T be a common ancestor of both types A and B in a supertype-subtype hierarchy. However, they also show that this is not sufficient. A converter C: A → B does not necessarily respect every common ancestor T of types A and B. In order to determine whether a converter C: A → B respects a given ancestor T in a type hierarchy, they provide the following definition:

"DEFINITION OF RESPECTS RELATION: Let C: A → B be a converter function, a partial function mapping values of type A to values of type B. Let T be an ancestor of both A and B in a given type hierarchy. Then converter C respects T if for each method m of T, $\forall a \in dom(C)$:

1. $m.pre_T[\alpha(a) / x_{pre}] \Leftrightarrow m.pre_T[\beta(C(a)) / x_{pre}]$ and
2. $m.post_T[\alpha(a) / x_{pre}, \alpha(a) / x_{post}] \Leftrightarrow m.post_T[\beta(C(a)) / x_{pre}, \beta(C(a)) / x_{post}]$"

where α and β are abstraction functions used to relate the value spaces of types A and B to their ancestors, respectively, in the hierarchy rooted at type T (depicted below).

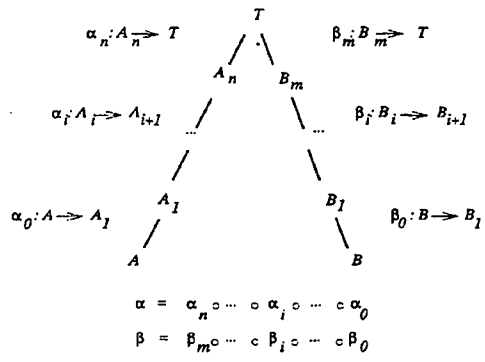

Fig. 6. Two compositions of Abstraction Functions.

NEED TO SHOW THAT EVERY CONVERSION C: A → B IN THE CTH RESPECTS A COMMON ANCESTOR.

If a given ancestor T in a type hierarchy is respected by a converter C: A → B, then T contains the behavior of type A that will be preserved when converted to type B. The remainder of A's behavior not captured by type T will be lost when converting from type A to B. If no ancestor T (actual or virtual) of A and B can be found such that C: A → B respects T, then the conversion from A to B cannot be effected directly.

Applicability or relationship of work to dissertation topic

The basic premise of the paper is to show how to determine whether a converter C: A → B respects a given ancestor T in a type hierarchy containing types A, B and T. How can this be used for our efforts?

Given two types, A and B, we want to convert an object of type A to type B, preserving as much of A's behavior as possible. In our Consolidated Type Hierarchy, we introduce a consolidated type T as the parent of two types A and B if A and B represent two different models of the real-world object represented by type T. We also define converters C_{T1}: A → T and C_{T2}: T →B such that the mapping from the subtype to the supertype and from the supertype to the subtype respect the supertype T. (see p. 591) Conversion from A to B can then can be conducted using converter C_{T1}: A → T and converter C_{T2}: T →B. (See p.589: "In general, if types A and B are "below" type T by a composition of subtype and representation relations and α is the abstraction function from A to T and β is the abstraction function from B to T and β is invertible, then $\beta^{-1} o \alpha$ is a conversion from A to B that respects T.")

On p.586, the author indicates that Ockerbloom implemented an instance of the Typed Object Model (TOM) that allows users in a distributed environment to store types and type conversion functions, to register new ones, and to find existing ones. This appears to closely resemble the Registration and Discovery aspects of creating the Consolidated Object Hierarchy from my Dissertation Proposal. A review of Ockerbloom's dissertation is advisable.

Wing's paper also distinguishes between the conversion of abstract types and the conversion of concrete instances of those abstract types. In the Consolidated Object Hierarchy, we are defining relationships and conversions between abstract types. We then use those relationships and conversion routines to effect the conversion from one concrete implementation of an abstract type to another concrete implementation as part of the wrapper process.

[KM98] Dynamic Classification Ontologies: Mediation of Information Sharing in Cooperative Federated Database Systems

Summary

In their paper, Kahng and McLeod address the use of a common ontology to resolve information-sharing issues due to the heterogeneity of component systems comprising a Cooperative Federated Database System (CFDBS). Their focus is on resolving *semantic heterogeneity*, which they define as differences in data representation due to the independent specification of data in the component systems. The authors cite five principal incompatibilities between objects that comprise semantic heterogeneity:

category:	different realization of the same or similar real-world entities
structure:	different objects of a compatible category may have different structures such as an attribute of one database being represented as an object in the other
unit:	two objects with a compatible category and structure may use different units of measure

terminology:	the use of different names for the same object (synonyms) or the same name for different objects (homonyms) lead to incompatibilities
universe of discourse:	the meaning of data may be hidden in the context and not explicitly specified

The authors concentrate on attempting to resolve semantic heterogeneity between systems caused by category incompatibilities. Their approach to resolving semantic heterogeneity is to adopt a common ontology as the basis for mutual understanding.

Kahng and McLeod define ontology as "a collection of concepts and interconnections to describe information units". They use a common ontology in the CFDBS to describe information exported from component information sources. Exported information is first extracted from the information source and then translated from the local data model to a common data model. Semantic heterogeneity among the exported information is then resolved by mapping it into a common ontology. Export, import and discovery mediators use the common ontology to facilitate information sharing among the components of the CFDBS.

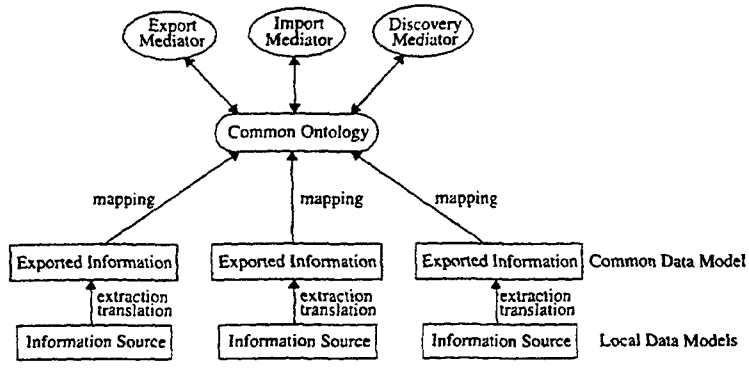

Figure 3. Information sharing in the CFDBS.

The authors discuss a number of different possibilities for use as a common ontology- the use of an integrated database schema that supercedes all component database schemas with mappings between the integrated and component schemas; the use of natural language keywords to describe and compare component data types; the use of pre-classified subjects (an extension of the keyword approach); the use of relationships among terms such as that provided by thesaurus-group generation; and the use of classification to subdivide component types into different classes. They chose classification as their approach to constructing a common ontology. The ontology used in their implementation of a CFDBS is termed a *Dynamic Classificational Ontology* (DCO).

A DCO is composed of a *base ontology* and a *derived ontology*. The base ontology is generally a static ontology for general description and classification of exported objects and is independent of specific concepts to be exported. The derived ontology is more dynamic, based on the base ontology and the population of exported concepts.

Knowledge in the DCO is used by mediators provided for information sharing. The export mediator utilizes the knowledge in the DCO to help components compose the description of concepts that they export. The component exports the concept by submitting an entry using the schema of the base ontology as a template. A discovery mediator is used to retrieve concepts relevant to a discovery request from those previously exported using the export mediator. The discovery mediator computes a *relevance factor* (RF) for each exported concept based on the discovery request. This relevance factor can be used to determine which concepts best match that of the discovery request.

Preliminary results of application of the DCO methodology to a medical information retrieval system have indicated that the precision and recall of document searches can be significantly improved over traditional search schemes.

Applicability or relationship of work to dissertation topic

During the consolidation phase of constructing the Consolidated Type Hierarchy (CTH), a common ontology such as that provided by the DII/COE Namespace Registry or Conceptual Model of the Mission Space (CMMS) could be used to define the consolidated types added to the CTH when resolving representational differences between component types. This would have the added benefit that later additions of components compliant with the selected ontology to the integrated system would greatly simplify the discovery and consolidation processes used to form the CTH.

[LAS98] Web Metadata: A Matter of Semantics

Summary

This article provides an overview of the World Wide Web Consortium (W3C) published Resource Description Framework (RDF) and describes how RDF can be used to provide metadata to capture the semantics of a Web resource. In the context of this article, Lassila defines metadata as "machine-understandable descriptions of Web resources." A Web resource is any object that can be addressed using a Uniform Resource Identifier (URI). By enabling the semantics of objects to be expressible and exploitable, RDF supports interoperability between applications that exchange machine-understandable information on the Web.

RDF provides a model for expressing instances of metadata regarding a resource. RDF is based on a concrete formal model utilizing directed, labeled graphs to relate semantic information about a resource to that resource. In the model, a resource is described through a collection of *properties* called an RDF Description. Each *property* has an associated *property type* and *value*. In the graph, the resources and values form the vertices of the graph, with the properties naming the edges between a resource and its associated value.

RDF is an application of XML, extending the XML model and syntax to provide descriptive information about resources. RDF uses XML schema information to define the *property types* for a particular resource. This schema information can come from a predefined *namespace* as identified using the XML Namespace facility or from the default namespace specified for the RDF instance.

Figure 1 below shows an example of RDF syntax that describes a Web resource.

```
<?xml:namespace ns="http://www.w3.org/TR/WD-rdf-syntax" prefix="RDF"?>
<?xml:namespace ns="http://purl.org/metadata/dublin_core" prefix="DC"?>

<RDF:RDF>
   <RDF:Description about="http://www.some.org/smith">
      <DC:Creator>John Smith</DC:Creator>
   </RDF:Description>
</RDF:RDF>
```

Figure 1.

This RDF instance describes a Web resource pointed to by the URI *http://www.some.org/smith* as specified by the *about* attribute of the <RDF: Description> tag. This resource has one property defined for it- Creator- whose value is the string "John Smith" indicating that "John Smith" is the creator of this particular resource. The example also incorporates the use of the XML *namespace* facility to identify the schema that defines the RDF syntax for naming a resource property, and to uniquely identify the property that provides semantic information about the resource. In this case the <RDF: Description> tag indicates that the <Description> tag is taken from the RDF namespace whose schema is located at *http://www.w3.org/TR/WD-rdf-syntax*. In addition, the indicated resource has one property defined for it- Creator- that is taken from the DC namespace located at *http://purl.org/metadata/dublin_core.*

In the model for this RDF instance, the Web resource pointed to by the URI *http://www.some.org/smith* is depicted as a vertex in the graph, with the *Creator* property depicted as a named edge connecting the resource node to the vertex containing the value "John Smith" of the Creator property. Figure 2 provides a pictorial representation of the RDF model.

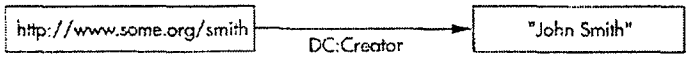

Figure 2. A graph generated from the example in Figure 1.

In addition to providing basic property/value information for a resource, RDF can provide property/value metadata for other metadata. Also, RDF property values can be complex objects, with a property having one or more subproperties. Finally, although RDF does not contain any predefined vocabularies for authoring metadata, it does permit the use of a central attribute registry.

Another capability that RDF provides is the extensibility and shareability of the underlying schema used to define the property/value pairs for a relationship. Schema extensibility enables you to define new schema by defining incremental additions to a base schema, instead of defining a completely new schema as with XML's DTDs. Schema shareability allows reuse of definitions and supports establishment of domain-specific definitions.

How then, does RDF support interoperability between applications that exchange machine-understandable information on the Web? According to the author, the key is RDF's capability for shared schemata to build common ontologies for the Web. He states that standardized metadata can provide a solution to the lack of machine-understandable semantics.

Applicability or relationship of work to dissertation topic

In trying to resolve representational differences between data types on heterogeneous software systems, a key problem toward automation of the integration process is the ability to recognize two different types as representing the same real-world object. Intuitively, the solution that promises the highest precision in recognizing two types as being related is one that utilizes semantic information about the types to determine the relationship. Thus, the ability to capture semantic information about the types from which to perform a comparison is attractive. Extension of the article's use of RDF from providing semantic information about a Web resource to that of being able to provide semantic information about a type, might prove fruitful. Although types are not currently envisioned to be separately referenced via a URI, modifications to the current type DTD definition or some other approach may enable use of RDF to provide semantic information about a type for use in solving the discovery problem.

Weaknesses

Although the article does point out how semantic information about a Web resource can be captured using RDF, the author provides little explanation of how this semantic information can be used to solve the interoperability problem on the Web. He does propose that standardized metadata provides "a solution to the lack of machine-understandable semantics" and that RDF's ability to share schemata can support the development of concept ontologies which can be used for standardizing metadata. However, he does not provide any details on how standardized metadata can be used to address the interoperability problem.

[HMS94] Object Discovery and Unification in Federated Database Systems

Summary

This paper discusses the formation of a Federated Database System (FDBS) from a set of autonomous, heterogeneous database components. The interest in heterogeneous databases arises from the desire to share information between existing databases which were independently developed with their own specific goals and criteria. Maintaining component database autonomy results from the desire for individual database owners to retain control over the organization and information release of their data in a sharing environment.

According to the authors, the key to achieving interoperability in a FDBS is largely dependent on two capabilities: 1) the ability of a component to identify and locate relevant non-local information for its use (the discovery problem) and 2) the ability to integrate such information into the component's local system framework (the unification problem). The authors attempt to solve the discovery and unification problems with their Remote-Exchange experimental system, the architecture of which is illustrated below.

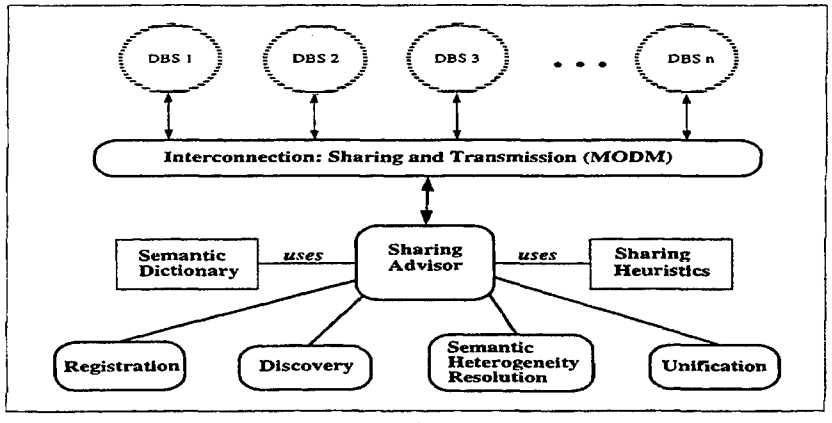

Figure 2: The Remote-Exchange sharing architecture

Remote-Exchange contains a component called the *sharing advisor* which is responsible for addressing the discovery and unification problems. The sharing advisor provides four services to the components of the federation which are used to identify relevant non-local (type) objects for incorporation into the local component's environment. These services are *Registration*, *Discovery*, *Semantic Heterogeneity Resolution*, and *Unification*. The sharing advisor utilizes a *semantic dictionary* to manage knowledge about objects that the components export and a set of *sharing heuristics* to assist in establishing relationships between exported objects.

Registration is used by a component to inform the sharing advisor about information it is willing to share with other components in the federation. Such information is added to the semantic dictionary in a bottom up fashion, generating a hierarchy of concepts exported by the components. The sharing advisor utilizes information from the sharing heuristics to establish the relationships in the concept hierarchy.

The sharing advisor utilizes the Discovery process to locate type objects in remote components that are related to a particular type object in the local component. Once a related component is found, any semantic discrepancies that may exist between components are resolved by the sharing advisor's Semantic Heterogeneity Resolution process. Finally, the non-local objects are unified with the corresponding local objects by means of the Unification process. Following unification, references to non-local type objects are treated as if the type object resided locally.

Applicability or relationship of work to dissertation topic

The approach for determining relationships among heterogeneous databases may be applicable to our planned efforts to resolve data representational differences encountered during the integration of legacy systems. Data types in an interface and data objects in a database are equivalent concepts. This work may be useful in 1) identifying data elements in the interface where representational differences occur and 2) resolving those representational differences through definition of a generalized abstract data type which data types in the interface are instances of.

In their description of Remote-Exchange's semantic dictionary, the authors describe the bottom-up establishment of the semantic dictionary concept hierarchy indicating that "the set of properties belonging to a concept at a particular level (is) represented as the union of properties of all its subconcepts". (p.8) This approach has been considered for our Consolidated Type Hierarchy in order to 1) enable the possibility of defining new type as the composite of more than one child, and 2) potentially simplify the structure of the Consolidated Type Hierarchy by only having two levels in any tree in the hierarchy.

The authors' approach to resolving semantic heterogeneity between type objects employs a local lexicon for each component which specifies the relationship between its local types and a global set of commonly understood concepts. (p.10) One potential source for the global set of commonly understood concepts could be the use of a common ontology such as the DII/COE namespace registry for use in discovery/consolidation process for the Consolidated Type Hierarchy.

Weaknesses

Sharing Advisor's Semantic Heterogeneous Resolution process establishes relationships between local and remote objects (equal, kindof, collectionof, etc.), but does not discuss method for resolving potential representational differences between the objects (such as may be found with our gridPosition / latLongPosition example). Semantic Heterogeneous Resolution may allow us to establish the relationships
<gridPosition> InstanceOf <Position> and
<gridPosition> Equal <latLongPosition>,
but more info is needed to utilize our gridPosition in a system that is expecting a latLongPosition for use in its algorithms.

Method for overcoming weakness or improving on previous results

In addition to establishing the relationships between local and remote objects, a method must be provided to translate between different representations of related objects.

[HM99] Resolution of Representational Diversity in Multidatabase Systems

Summary

Hammer and McLeod present an approach for sharing data among components of a loosely coupled federated database system. Their Remote-Exchange architecture accomplishes data sharing between a local and a non-local system by folding the non-local data into the local system's data schema. In order to effect the importation of non-local data into a local system, the following must occur. First, a common model for describing the sharable data must be established. Second, any semantic and representational differences between the local and non-local data schema must be resolved.

The issue of a common model is addressed through Remote-Exchange's Core Object Data Model (CODM) which provides a common data model for describing the structure, constraints, and operations for sharable data. Through CODM relationships between local and non-local data objects are established as a precursor for data sharing.

CODM includes a Remote Sharing Language (RSL) to provide a "standardized interface to the conceptual schemas of the participating components." RSL provides primitives to obtain structural information about component databases in order to capture the relationships between data objects (resolution) and to enable importation of relevant remote objects into a local database schema (unification).

A foundation of the author's approach for determining the relationships between local and non-local data objects is the ability to provide semantic information about the sharable objects in each component in addition to the syntactic information provided by an object's schema. The semantic information is captured in the form of a *local lexicon* which defines a relationship between objects in the local database and a list of commonly understood terms from either a general-purpose ontology (GPO) or one of several domain-specific special-purpose ontologies (SPOs). Object representations from two different local databases can be considered related if they each contain a local lexicon that relates both to the same ontology term.

Remote-Exchange uses an object's syntactic information along with the semantic information contained in the local lexicon to establish relationships among entries in different lexicons. This information is captured in the form of a concept hierarchy that depicts relationships between objects in different databases, and is stored in a global repository called a *semantic dictionary*. In addition to the concept hierarchy, the semantic dictionary also contains the general-purpose and special-purpose ontology information and the relationship descriptors that are used by the local lexicons to describe relationships between local database objects and terms from the ontologies.

Relationship information stored in the semantic dictionary concept hierarchy is used by Remote-Exchange's *Sharing Advisor* to identify data that is sharable between federation databases. A sharing tool then imports sharable data from a non-local component into the appropriate place in the local type hierarchy for use.

Applicability or relationship of work to dissertation topic

The Consolidated Type Hierarchy (CTH) envisioned in my research effort serves a similar purpose as the author's CODM. The CTH is used to establish a relationship between data types on different systems that represent the same real-world data object much as the CODM defines relationships between local and non-local data objects

The registration and discovery process used in the Remote-Exchange project is closely related to the process envisioned for resolving data representational differences during the integration of heterogeneous software systems. The communication involved in the integration of heterogeneous software systems can be thought of as a producer-consumer relationship. Integration of two systems is generally undertaken where one system produces some data object that another system can use. The determination of what data objects are being produced in an interface is analogous to the registration process of Remote-Exchange; each producer system "registers" those data objects it will export by means of the consolidated type hierarchical model. The problem of finding appropriate producer objects

for a system wanting to import data object(s) to its system is then analogous to the discovery process in Remote-Exchange; the consumer system wants to locate all data objects provided by the producer(s) that are related to the object(s) it wants to consume.

Discovery in the Consolidated Type Hierarchy could benefit from a methodology such as that provided by Remote-Exchange's local lexicon. The location of related producer types to satisfy a consumer request will undoubtedly require comparison using more than just a type's syntax. The local lexicon also allows a level of semantic comparison that may help to eliminate candidates that might otherwise be incorrectly considered.

The relationship between Remote-Exchange and the proposed research diverges at this point. Remote-Exchange integrates the discovered remote object into its local framework. Our integration process will need to provide a means for resolving any representational differences between the discovered "producer" and the "consumer".

[YS94] Interfaces, Protocols, and the Semi-Automatic Construction of Software Adaptors

Summary

A growing area in object-oriented software development is the trend of constructing software applications from parts- fully functional components are connected together to form a new composite application. While acknowledging the potential benefits of such an approach, Yellin and Strom address two principal challenges facing object-based component composition: 1) how can you specify component interfaces such that you can determine from the interface specification whether two different components will work properly together if connected? and 2) for two different components that are functionally compatible but whose interfaces are not type compatible, can you provide adaptors to enable the components to work together?

In response to the first challenge above, the authors introduce the use of an augmented interface description for the components to be combined, called a *collaboration specification*. A collaboration specification contains (1) an *interface signature* describing both messages being sent and messages being received by a component and (2) *protocols* defining the legal sequence of messages that can be exchanged between components. Sequencing constraints are defined in terms of a finite state grammar specifying a set of states and a set of transition rules between states, where there is one transition for each message that can be sent or received from a particular state.

The collaboration specification for two components can then be used to determine protocol compatibility between the two components. Comparison of the collaboration specifications of two components will determine if the two components will work together when connected.

In response to the second challenge of enabling two components that are functionally compatible but were not designed to compatible collaboration specifications to interact, the authors introduce an intermediary between the two components called an *adaptor*. The adaptor is modeled as a finite state machine that has interfaces to the two components that want to collaborate. When one component sends a message to another functionally

compatible mate, the adaptor intercepts the message and translates it into a form that is protocol compatible with the collaboration specification of the second.

Given the collaboration specifications for two different components that are desired to be integrated, the authors define a semi-automated methodology to synthesize a well-formed adaptor consistent with the specifications, or if no such adaptor exists, determine that it the case. The methodology starts by defining an *interface mapping* between the interface signatures of two collaboration specifications. From the *interface mapping*, the authors sketch an algorithm that will produce an adaptor that is valid with respect to the interface mapping or determine that no such adaptor exists.

Applicability or relationship of work to dissertation topic

A key issue which must be resolved in attempting to provide interoperability between a federation of heterogeneous computer systems is how do you ensure the correct sequencing of data producers and consumers? The idea of the addition of a *protocol* defining the legal sequence of messages that can be exchanged between components to a component interface and the use of that protocol information to define adaptors between components that enforce protocol compatibility between components may help in resolving this issue.

Weaknesses

The types of protocols provided in the paper's examples appear similar to the type of handshaking protocols you would expect in a communications network application, i.e., the message exchange occurs between two components and is defined by a sequence of query/response type actions. The kind of protocol expected in a federation of heterogeneous computer systems would differ from that provided in the examples. In a federation of heterogeneous computer systems one would likely expect a sequencing of messages involving many components, with no direct query/response interaction between adjoining components.

[YS97] Protocol Specifications and Component Adaptors

Summary

In an update of their previous work appearing in the ACM OOPSLA 1994 Conference, *Interfaces, Protocols, and the Semi-Automatic Construction of Software Adaptors*, Yellin and Strom strengthen the theory provided for enhanced interface specifications, protocol compatibility and component adaptors. In this article they provide three theorems which summarize their work on protocol specifications and component adaptors:

"THEOREM 2.3.2. *There exists an algorithm for checking protocol compatibility.*

THEOREM 3.3.1. *There exists an algorithm for checking whether an adaptor A is compatible with protocols P_1 and P_2.*

THEOREM 4.3.1. *The adaptor synthesis algorithm will either produce a valid adaptor w.r.t. the interface mapping I or will correctly conclude that no such adaptor exists.*"

Proofs of these theorems and related lemmas provide a sound foundation for possible application of this material to my dissertation efforts.

[CMK98] A Protocol Based Approach to Specifying Interoperability between Objects

Summary

Cho, McGregor, and Krause provide an overview of several techniques for determining the interoperability of two components and propose a new technique for determining component compatibility. The authors define interoperability as the ability of two software modules to communicate and cooperate with each other. Thus, their definition for interoperability extends beyond simply enabling the exchange of information between two modules. It also entails the capability to interact and jointly execute tasks.

The authors review six existing techniques for achieving interoperability: Zaremski and Wing's protocol specification approach, the Polylith system, CORBA IDL, Formal Connectors, Software Adaptor, and the PROCOL approach. In reviewing these techniques, they pose the following questions:
"How can a component be specified so that it can interoperate with another?
How can we identify a module with which we want our application to interoperate?
How can we make one module interoperate with the other if they are not compatible?"
From these questions, nine criteria are provided for comparing the techniques' interoperability response:

1. Module- the level of software modules considered, whether function, object or component
2. OO Support- whether the technique supports object-oriented or procedural languages
3. Multi-language- whether the modules are heterogeneous or homogeneous
4. Additional Specification- whether separate specification required for each component
5. Separate Specification- whether the specification is provided within the module or separate from the module
6. Specifications needed at- whether the specification is needed in both modules or only one
7. Semantic match- whether the specification includes semantic interoperability or only syntactic interoperability (assumed)
8. Protocol match- whether the technique utilizes protocols to determine interoperability
9. Matcher (Adaptor)- whether the technique provides an adaptor to enable interoperability between two components that are functionally compatible but whose signatures do not match

The results of the comparison are provided in Table 1 below.

	Specification Matching	Polylith	CORBA IDL	Formal Connector	Software Adaptor	PROCOL
Module	function, component	module	component	component	component	object
OO Support	possible	no	yes	yes	yes	yes
Multi-language	no	yes	yes	generic	no	no
Additional Specification	signature, pre/ post, protocol	MIL	IDL	WRITE arch. language	(state machine-like) protocol	method guards
Separate Specification	no within module	yes	yes	yes	yes	no (within object)
Specifications needed at	both sides	both sides	server only	separate component	both sides	both sides
Semantic Match	yes	no	no	yes	no	no
Protocol Match	yes	no	no	yes	yes	yes
Matcher (Adaptor)	none	software bus	none	none	adaptor	no

Table 1. Categories in module interoperation

Traditionally, interoperability checking is done by comparing the interface signatures of two components to see if the components can successfully work together. The interface signature generally specifies the allowable messages that can be sent or received from a component. This in itself is not sufficient to ensure interoperability between two components. In addition, the sequence of message transmission between the two components must be taken into account to ensure that messages are transmitted and received when expected by each component. The sequence of allowable messages defines a *protocol* which must be compatible with the protocol defined for a connecting component in order for the two components to be interoperable.

The authors next introduce DOSAEE (Domain-Oriented Software Analysis and Engineering Environment) and propose a new protocol specification language to form what they term the Interoperable Component Model (ICM). DOSAEE is a graphical tool that uses protocols, components, and patterns to create a domain architecture for a software system. ICM builds on DOSAEE and provides a more complete specification that aids in the design of component interactions. ICM provides sufficient information to determine whether two components can interoperate successfully. The authors envision ICM as a step toward the achievement of plug-and-playable software modules.

Applicability or relationship of work to dissertation topic

The paper provides a good overview of various techniques used to achieve interoperability between different component systems. Of particular interest to my dissertation is the discussion on software adaptors. A software adaptor provides a bridge between two components with functionally compatible but type incompatible interfaces. The software adaptor compensates for differences in message and parameter names, parameter orders, or numbers of states. An adaptor intercepts outgoing messages from a component and translates the message to the format and content expected for an incoming message of the other component.

Another area where this paper may be related to my dissertation is in the area of protocols. The authors define a protocol as the sequence of messages involved in the interaction of two components. One issue I am looking at is the sequencing of messages among all components in the federated system and how this required sequencing can be captured in the consolidated object hierarchy model. Perhaps some variation of the protocol concept can be used.

Weaknesses

A primary motivation between trying to achieve interoperability between heterogeneous systems is the desire to capture the investment in legacy software. Such systems generally are procedurally-oriented and do not utilize object-oriented methodologies. Therefore, the paper's focus on object oriented systems excludes a large number of systems which are primary drivers in the push to achieve system interoperability.

[Sin98] Unifying Heterogeneous Information Models

Summary

The article presents an automated approach to unifying heterogeneous information models based on machine-processable metadata specifications. The approach is realized in the author's Tesserae system, which uses the metadata information to dynamically resolve representational differences. The metadata descriptions are used to infer relationships among data objects, using these relationships to unify heterogeneous data representations into a common object data model. Then, using the common object data model, the system dynamically decides how to handle requests for data from heterogeneous sources at runtime.

The Tesserae system consists of a number of information sources and consumers, and a central information integrator, the Tesserae Information Engine (TIE), which resolves differences between information requests and the different information providers.

As its primary means of resolving the heterogeneity between different information sources, the TIE maintains a body of information about the information sources and prospective data consumers. This *metainformation* includes intelligence on the contents of available information sources as well as descriptions of consumer interests. In addition, the metainformation contains definitions of the vocabulary used by both the information sources and data consumers.

The approach used by the Tesserae system is for information providers to provide a knowledge base that defines the vocabulary used by the provider in terms of a standard vocabulary. Then, where differences between the vocabularies of an information provider and consumer exist, the information in this knowledge base can be used to resolve the heterogeneity between terms.

The information in the knowledge base is captured using a metadata language, which the system uses to automatically resolve differences between information providers and consumers at runtime. The metadata language consists of three components: "a vocabulary, a content language known as KIF (Knowledge Interchange Format), and a communication wrapper language named KQML (Knowledge Query and Manipulation Language)."

The vocabulary consists of a universally agreed-upon set of words, where each word has a single meaning, thus providing the standard vocabulary for use by all components. The KIF content language utilizes first-order predicate calculus to define the metadata for unifying the heterogeneous information models using a body of terms, functions and rules to describe the relationship of the component vocabulary to the standard. KQML defines a communication layer on top of KIF that is used to describe the type of request associated with the embedded KIF sentence.

TIE uses automated inference to handle a query or notification. The information contained in KQML and KIF sentences is used to resolve any mismatches in the information models of the sources. The procedure used by TIE is based on model-elimination, a variant of the backward-chaining procedure used in Prolog.

Applicability or relationship of work to dissertation topic
The approach outlined by Singh has potential for application in the Discovery process used to develop the Consolidated Type Hierarchy (CTH). Similar to the use of a common ontology outlined by Kang and McCleod in [KM98] Dynamic Classification Ontologies: Mediation of Information Sharing in Cooperative Federated Database Systems, use of a standard vocabulary to relate items in different components may prove of value in identifying different representations of the same real-world object for consolidation in the CTH.

Weaknesses
One weakness of the author's approach is the requirement to maintain a set of terms, functions and rules to relate a component's types to some standard vocabulary. The overhead required to provide such a body of information may prove to outweigh the benefits obtained from its use.

[RKM97] A Three-Layer Model for Schema Management in Federated Databases
Summary
The paper provides a framework for a three-layer architecture to create a federated system from a number of heterogeneous legacy systems. A principal goal of the three-layer architecture is to isolate both the federation and the component systems from changes to the other.

The three-layer architecture consists of the *Federation Layer*, which manages metadata at the federated system level, the *Component Layer*, which serves to isolate the federation metadata from the local database metadata, and the *Integration Layer* which contains the local database metadata.

Integration of the three layers is accomplished by the use of an object-oriented *canonical data model (CDM)* which allows local databases (LDBs) to interoperate with other LDBs even though they may use a different data model. This is accomplished by converting each LDB format into a CDM format, resulting in the creation of a new *component schema* that contains mappings to the original local schema attributes, at the Integration Layer level. These component schemas map to various *export schemas* at the Component Layer level, which in turn map to the *federated schemas* at the Federation Layer level.

By mapping the LDB format into a CDM format, and separating the resultant component schema from the higher level export and federated schemas, the authors achieve the desired result of isolating changes at either the federation level or the local database level from impacting each other. The composition of the federation can be changed without impacting the local databases making up the federation. Local databases can be added, deleted, or replaced in the federation without modifying the individual databases concerned. In addition, modifications can be made to the local databases without impacting the federation. Attributes can be added, deleted, or modified without rebuilding the schema.

Applicability or relationship of work to dissertation topic
Use of a canonical data model to convert the local database schema to a component schema is similar to the planned approach of using an intermediate representation to integrate two systems where differences in data representation exist. I also like their approach of treating the integrated system as a federation of individual databases. The layered approach as a means of preventing changes in individual members of the federation from impacting the entire federation and vice-versa is also an attractive approach.

Weaknesses
The paper provides a framework for a three-layer architecture to create a federated system from a number of heterogeneous legacy systems. It talks about the probability of differences in data representation between the various legacy systems, but doesn't specifically address the types of representational differences that might be possible, nor does it outline a method or model for resolving those representational differences. It does talk to using a canonical data model (CDM) format as an intermediate format for representing the data and the use of conversion routines to convert between the format of a local database and the CDM. However it doesn't provide many specifics on what the CDM consists of or how the conversion routines might be invoked.

[Blo92] Power Programming with RPC
Summary
Sun Microsystem's Open Network Computing group's Remote Procedure Call (RPC) mechanism uses a "single-canonical format for data representation" known as External Data Representation (XDR) to represent data structures in a machine-independent form. XDR was developed to enable "complying machines to share data regardless of compiler, operating system, or architecture differences." (p. 3) XDR provides both simple conversion routines, to convert between built-in C data types and their XDR external expression, and complex conversion routines, for handling such complex data types as vectors, arrays, unions, strings and pointers, or references.

In order for two heterogeneous systems to exchange data, the systems would invoke the appropriate XDR conversion filter to translate data into and out of the external data representation for the specified type. Thus, on the source end, a message originator would encode outgoing data into the XDR format using a source-specific XDR conversion filter and then, on the destination end, the XDR format message would be converted to the appropriate type using the destination-specific XDR conversion filter.

XDR can be used to resolve representational differences due to low-level data format hetereogeneities such as different byte ordering, floating-point representation, etc., resulting from compiler, operating system or system architecture differences. However, XDR cannot resolve differences caused by different data structures (use of attribute vs. element), terminology (use of synonyms and homonyms), or universe of discourse (semantics hidden in the context). [KM98]

[CS91] Semantic Enrichment of Database Schemas: An Object Oriented Approach

Summary

In this paper, the authors offer an approach for dealing with heterogeneity in a federated database system. As discussed, this heterogeneity can be manifested as *syntactic heterogeneity*, where different databases can be implemented using different data models, and *semantic heterogeneity* where, although the databases may adhere to the same data model, data structural differences occur due to different interpretations by different designers.

A common approach to resolving syntactic heterogeneity is the adoption of a canonical data model to which all of the component schemas are mapped. Then, once a common data model is defined for all of the component systems, relationships between data elements in the model can be defined using this model. These relationships are used to define the structure and contents of each database and can be used to assist in resolving the semantic heterogeneities between the component systems.

The authors use an object oriented data model, which they call BLOOM, as the canonical data model for their Federated system. A federated data schema is constructed from a group of local databases in a two step process. In the first step, the enrichment or conversion step, types, relations, objects, etc. of the local schema are converted to corresponding constructs in a component schema using the canonical data model. In the second step, the association or integration step, a federated schema is constructed from the component schemas.

The canonical model used for BLOOM utilizes a number of abstractions, specialization and specific dependencies to model the capabilities of the local schemas. The abstractions include classification, cartesian aggregation, cover aggregation, and generalization/specialization. BLOOM also includes four kinds of specialization: disjoint, complementary, alternative and general which express the numerical relationship between the superclass and subclass. Finally, BLOOM also includes constructs to represent both interest dependency (object is of interest only as long as another object is of interest) and existence dependency (existence of one object dependent on the existence of another object) relationships between objects.

The authors then show how to enrich a relational database schema using the BLOOM data model to produce an object oriented schema for use in the federated System.

Applicability or relationship of work to dissertation topic

Although primarily concerned with converting a relational database schema to an object oriented schema, the abstraction, specialization and specific dependency constructs used by the BLOOM data model may prove useful for application in the Consolidated Type Hierarchy (CTH) to assist in the discovery process. The additional relationships may help relate types being registered to existing types in the CTH.

Weaknesses

In their paper, Castellanos and Saltor demonstrate the enrichment or conversion of a local database schema to an object oriented component schema. They do nothing, however, to

demonstrate how the component schema is utilized to resolve representational heterogeneities in constructing a federated schema from component schemas.

[GSC95] A Structure Based Schema Integration Methodology

Summary

One approach to sharing data between a number of autonomous databases is to combine the databases into a federation by placing a Federated Database Management System (FDBMS) on top of the individual DBMSs. Because the component databases were most likely developed independently, with different designers, and different views on the part of the real world to be captured in the database, these component databases will be heterogeneous, both in their data models and DBMSs (syntactic heterogeneity) and in the composition and meaning of their schemas (semantic heterogeneity). The FDBMS often uses an integrated schema using a common data model to consolidate the schemas of the component databases as a means of addressing these heterogeneities.

One common problem encountered in the development of an integrated schema is how to determine which classes or types in one database are similar to which classes of another and subsequently how to resolve those differences. The author's offer an approach to solving that problem in the form of their structure based schema integration methodology. Their approach is based on enriching the schemas of the component databases to be integrated by capturing structural relationships between data classes in their BLOOM canonical data model. Then using the enriched BLOOM schemas, a search for similar classes is conducted during the *detection* phase, and similar classes are integrated into a federated schema during the *resolution* phase.

The first phase of the author's integration methodology is the *semantic enrichment* phase. In this phase, knowledge regarding generalization/specialization, aggregation, classification/instantiation, and dependency relationships among the database classes are made explicit in the form of an enriched BLOOM schema. (See [CS91] Semantic Enrichment of Database Schemas: An Object Oriented Approach for further information on the semantic enrichment phase.)

In the next phase, the *detection* phase, similarities that exist among the classes of the semantically heterogeneous database schemas are identified. In order to determine where similarities exist between the classes in two databases, the authors first outline a strategy to determine which classes from each database to compare, and then define the criteria by which similarity is decided. The strategy is guided at a *course* level by the generalization/ specialization relationships between classes which determines groups of classes to compare, and at a *finer* level by the aggregation relationships between classes in each group. If two classes to be compared don't share the same aggregation relationship, then *relaxation* can be applied to the class with the stronger aggregation abstraction to make in comparable with the weaker. These relaxations result in *penalizations* that are taken into account when computing the degree of similarity between the classes.

For objects being compared, a Degree of Similarity is computed based on the type of abstraction of the component classes, any relaxations performed on the class, and similarity between the components of the classes.

Finally, during the *resolution* phase, classes/objects are integrated only if they are similar and the specializations in which they participate as subclasses are similar as well.

Applicability or relationship of work to dissertation topic
The concepts presented in this paper have potential application in the discovery process used in constructing the Consolidated Type Hierarchy. While the authors' approach requiring the enrichment of the local database schema to include the structural relationships between objects is not the preferred way to go, the envisioned methodology for the discovery process will probably entail some level of structural comparison between objects.

Weaknesses
The authors' approach is based on analyzing the structural relationships between objects, generalization/specialization and aggregation/component patterns, and not on the actual meaning/usage of the objects involved. As no analysis is provided regarding the effectiveness of the proposed solution to the detection problem, it is questionable whether this will be sufficient for finding appropriate producer objects for a particular consumer. It is probable that other methods involving object syntax and semantics will be necessary to achieve sufficient precision and recall in matching producer types with consumers (such as those provided in the work by Steigerwald, Hermann, and Nguyen).

An additional weakness of the author's approach is the requirement for enriching a database to include the generalization/specialization and aggregation relationships. While feasible as part of a layered approach to implementing a federated system (see [RKM97] A Three-Layer Model for Schema Management in Federated Databases), the approach appears extremely complicated and difficult to implement.

[Pit97] Providing Database Interoperability through Object-Oriented Language Constructs
Summary
In this paper Pitoura makes a distinction between interconnectivity, the ability of systems to communicate and exchange information, and interoperability, the additional ability of systems to interact and jointly execute tasks. The methods outlined in the paper pertain to achieving the higher-level goal of interoperability.

Pitoura starts by listing the various causes of heterogeneity among database systems and provides examples of types of heterogeneity that might exist among database systems. The paper then introduces a methodology for achieving interoperability among heterogeneous systems.

The proposed methodology is based upon an object-based approach to integration. Key to the integrated system is a language, termed a multilanguage, through which the integrated data is defined and manipulated. The multilanguage is a unified language that serves as both a data definition and data manipulation tool. The multilanguage is defined in terms of

additional constructs which should be added to an existing object-oriented language vice as a new programming language.

An overview of Pitoura's object-oriented approach to integration is provided in Figure 1. The resources of the local systems that participate in the integrated system are modeled as objects and the services provided by those resources are modeled as the methods provided by those objects. The objects are termed *virtual* to indicate that they may not actually be stored as distinguishable entities, but that they may be formed from a combination of existing resources. From the user's perspective, the integrated system is composed of a number of global virtual objects, the use of whose resources is obtained through a corresponding global method. The fact that invocation of a global method results in execution of one or more underlying local methods is hidden from the user.

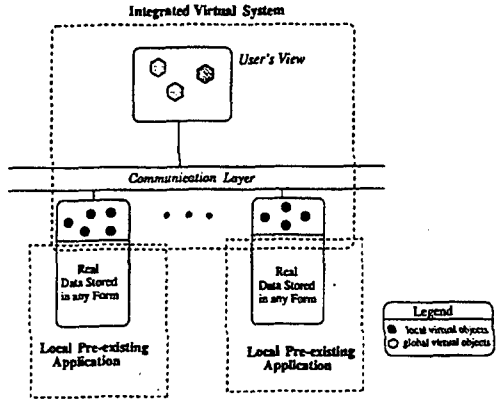

Figure 1. Distributed object architecture.

Implementation of the Integrated Virtual System is accomplished using a two-phase process as depicted in Figure 2. Grouping together objects of similar behavior and structure forms a class. Each system that participates in the integration provides a set of local virtual objects consisting of a set of *basic classes* and a set of *basic methods*. During design or compile time new *virtual classes* are defined by combining *basic classes* using the view mechanism of the multilanguage. Virtual classes are used to express relationships among basic classes and methods of pre-existing databases, and to resolve conflicts between classes and methods that exist because of their heterogeneity. Methods for the new virtual classes are formed as a combination of the methods on the basic classes. Then, during run-time, the user invokes the multilanguage to send queries via messages to the objects of the virtual class, which are translated into the appropriate basic methods for execution on the corresponding local system. The virtual system then combines the individual results and presents them back to the user as a consolidated response expressed in the multilanguage.

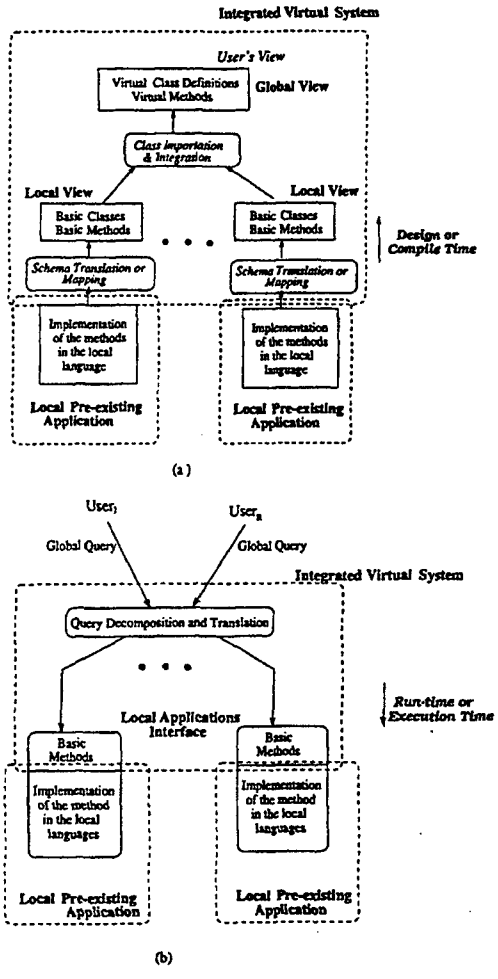

Figure 2. The use of the multilanguage (a) at design or compilation time, to define views and (b) at execution time, to express queries.

The multilanguage defined by the author

- is programming language-based in order to allow the integration of non-database systems
- is capable of combining and manipulating persistent pre-existing data
- provides for combining information from the component databases through the use of the view mechanism
- provides efficient data access and nonprocedural operations such as search and select
- provides constructs for specifying the control flow of each task and the interaction among concurrently executing tasks

Applicability or relationship of work to dissertation topic

There are two similarities between Pitoura's work and that proposed for my dissertation. First, Pitoura uses virtual classes to express relationships among basic classes and methods of pre-existing databases, and to resolve conflicts between classes and methods that exist because of their heterogeneity. This is similar to my use of consolidated types to resolve conflicts that exist between types on two systems being integrated. Second, Pitoura uses a two-phased approach to constructing the integrated virtual system. In the first phase, she

defines the virtual class definitions and virtual methods from the local systems basic classes and methods. In the second phase, user queries are mapped to the appropriate basic methods, using the hierarchy established between the virtual methods and local methods. This is similar to my proposed two-phased approach where the Consolidated Type Hierarchy is constructed prior to runtime and then used during runtime to effect translation between heterogeneous data types.

[KGF98] Exploitation of Database Meta-Data in Assisting Database Interoperation

Summary

Karunaratna, Gray, and Fiddian propose an approach for combining a number of autonomous, heterogeneous database systems into a loosely coupled federation.

The central concept of the author's approach is a four-layered Knowledge Base (KB) that provides the structure to unify a number of autonomous, heterogeneous databases into a unified multi-database system. The bottom, database layer maintains information about the component databases. Above this, the meta-data layer maintains schema information about the databases contained in the database layer, organizing the objects into clusters of semantically similar schema objects. The schemas in a cluster all refer to the same real-world data object. The next layer in the hierarchy, the concept layer, contains a number of concepts that map to the schema clusters at the meta-data layer, thus each concept refers to a different real-world object. Finally, the view layer contains meta-data defining different user-defined views of the data that map to one or more schema clusters at the meta-data layer.

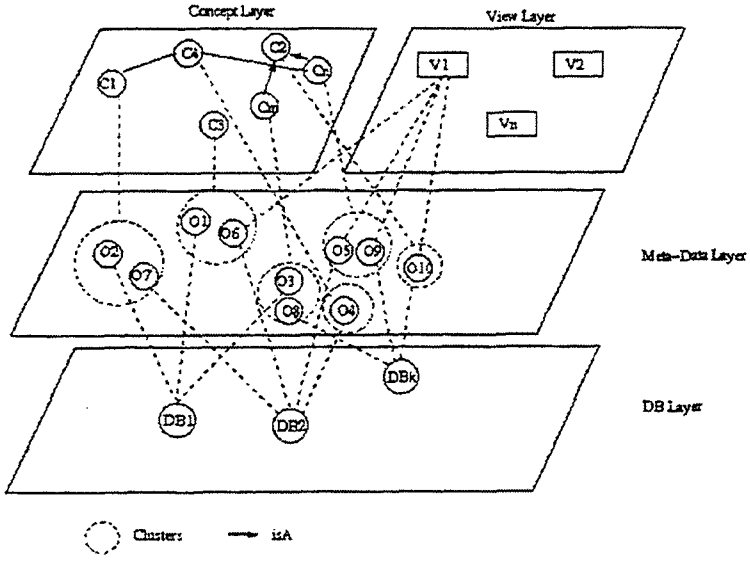

Figure 2: The Structure of the KB

Meta-data from component database schemas and the DB owners is used to build the knowledge base in a bottom-up fashion. Knowledge about a component database is derived as it is added to the federation. Thus, the knowledge base is continuously evolved as new databases join the federation.

The author's approach is not based on any pre-existing global taxonomy or dictionary such as a thesaurus, WordNet or some other ontology. Instead, a custom thesaurus containing terms related to the components comprising the federation is built in a bottom-up fashion from the database schemas making up the federation. The resulting custom-built thesaurus is stored as the various concepts at the concept layer. The authors are looking into the future possibility of using WordNet or some other ontological information to enrich the concepts contained in the Knowledge Base.

In addition, the authors provide an environment of software tools to aid the user in constructing the Knowledge Base. Major tools in the environment include the Meta-Data Extractor (MDE) which is used to extract schema information from the component DBs and DB owners, a Knowledge Server (KS) to maintain the Knowledge Base and a Graphical User Interface (GUI) used by the user to interact with the MDE and KS.

Applicability or relationship of work to dissertation topic

The information correlation problem, resolved by the authors by specifying concepts in the Knowledge Base to relate semantically similar objects, is closely related to our problem of trying to identify two types that are different representations of the same real-world object. Their solution of using a meta-model to represent the implicit relationships among different schema elements is similar to our concept of using the Consolidated Type Hierarchy to represent relationships between different representations of the same real-world object. In addition, the author's references to the future possibility of using WordNet or some other ontology to enrich the Knowledge Base may have relevance to our thoughts regarding the use of a common ontology such as the DII/COE Namespace Registry in creating consolidated types.

[RN91] Some Thoughts on Systems Integration: A Conceptual Framework

Summary

In building large, complex software systems, the requirement for systems integration is an issue whether you are constructing the system from existing components or applying an integrated approach to developing a system from scratch. In either approach, in order to be successful, you must be able to coordinate the integration and development activities in a managed and coordinated process. In order to define that process, Rossak and Ng propose that it is necessary to first identify all the elements of systems integration and then to organize these elements in a conceptual framework pertinent to the integrated systems' domain.

The authors propose a model of a layered approach to identify the components in the systems integration task. Their model contains three major components: enabling technologies, integration architectures, and global integration.

The mechanisms, systems, and tools that make construction of integrated systems possible are addressed under the enabling technologies component of their model. Enabling technologies include such technology areas as networking and communication technology. An integration architecture serves as the overall plan defining the basic layout of an integrated system. It is used to define the overall composition of the integrated system as well as the method of system decomposition, data storage, data communication and interprocess interaction. The integration architecture "guides the specification and the development of all the components which are going to be aggregated to constitute the integrated application." Global integration addresses the overall coordination and interoperation of the various system components from a system-level perspective.

The authors describe four elements of an integration framework which should be included in an integration architecture: the conceptual layout of the architecture, the mapping of the domain model into the architecture, the applied standards, and the implementation guidelines. The conceptual layout of the architecture is concerned with the specification of requirements and constraints for system components, communication technologies and data storage mechanisms. The methodology used for processing and handling data as well as the data communication model will vary depending on the application domain of the integrated system. This domain model must be incorporated into the integration framework. The standards used in the implementation of the generic integration architecture must also be included. Finally, implementation guidelines which describe the basis of the architecture in terms of existing methods and tools such as hardware restrictions, programming languages, coding standards, documentation guidelines, etc. should be captured in the integration architecture.

The authors identify three basic types of integration architectures: message-passing systems, systems with a central data repository, and generic systems. Message-passing systems focus on the communication aspects of the architecture, leaving the handling of data to the distributed components of the system. Systems with a central data repository resemble the architectures of large database-oriented systems with the following major difference- the integration architecture specifies a general model for an application domain whereas the database system specifies a point solution for the particular application. Generic systems enable specialization of a general-purpose solution to enable it to run in different application environments depending on the input set of initialization parameters.

There are two aspects to the global integration problem. From a technical point of view, the major integration issues are the fine tuning of the system architecture, the abstraction and reengineering of extant problem solutions, the resolution of interface problems, extending the capability of existing solutions where required, and enforcing architectural standards. The second aspect of the global integration problem involves semantic integration and domain modeling.

Semantic integration is required to due to often incompatible data structures and representations used by different system components. To overcome this problem, the authors propose the use of a system-wide meta data system to provide an integrated and standardized

view of the system's data resources. The meta data system enables the translation of data structures and semantics between the component system and the integrated meta model.

Once the elements and concepts for systems integration have been defined, the next logical step is to look at the methods for managing the integration tasks. Although there is no standard process model for system integration, two basic classes of approaches can be identified: the postfacto or bottom-up approach and an a priori top-down approach. Postfacto integration involves connecting nonstandardized and incompatible components into a cohesive integrated system. The authors identify software "glue" code as the means of coupling two or more given subcomponents. This software glue performs the following functions in integrating (possibly) heterogeneous components: call assist mechanisms, type conversions, protocol matching, and interlanguage and interprocess communications.

Top-down integration follows an incremental development approach, starting with an abstract conceptual requirement for the system and then incrementally deriving the functional and concrete system specifications, and finally a concrete system implementation.

Applicability or relationship of work to dissertation topic

This article doesn't support my research regarding data representational differences. It does provide some potential background information regarding the basic type of integration architectures (message-passing, central data repository, and generic systems). It also discusses the steps involved in system integration: 1) developing a generic model for the integration process, 2) defining the methods to handle this process, 3) developing the tools needed to support the different phases and tasks, and 4) defining the metrics, measurements and control structures to guide the integration process.

[YK95] An Object-Oriented Approach to Computer Integrated Systems

Summary

Yoon and King focus on the use of an object-oriented approach for defining, designing and developing Computer Integrated Systems (CIS). A CIS is a system of often heterogeneous subsystems incorporating some form of computer control, used in applications from manufacturing to aircraft flight control systems. The authors provide four major characteristics of CIS's:

"1. A CIS consists of heterogeneous subsystems which have been designed and manufactured according to different design principles by different vendors.
2. One or more components of a CIS can be added or deleted depending on the needs without damaging the integrity of the entire system.
3. Each subsystem is autonomous and capable of communicating with other subsystems.
4. A CIS could merge into a larger system." p.160

Based on these characteristics, the authors propose the use of objects as the fundamental elements of CIS's and view a CIS as a network of objects which are computing agents that communicate with each other. They define an object as a computational model for an entity that is represented as a set of data elements and a set of operations defined on those data elements. Objects are grouped into layers which form the subsystems that comprise the CIS.

The authors present three techniques for the representation of objects: algebraic, modular, and graphical. In the algebraic representation, an object is defined as a tuple consisting of a set of sorts or types, a set of operations defined on those sorts, and a set of equations defining the allowable semantics for the defined set of operations. The power of algebraic specifications is the ability to represent all major concepts in Computer Science in terms of the algebra. Additionally, a new algebra can be obtained from the sum or product of existing algebras, therefore providing an extension mechanism for defining a system from a set of fundamental components.

The modular representation has been used by various programming languages to represent objects and is equivalent to the algebraic one. The modular representation includes the *class* mechanism in C++ and the *package* in ADA.

A third equivalent representation, the graphical representation, was introduced as a visualization aid to the designer, as an alternative to the purely textual algebraic and modular representations. These representations consist of a graphical rendering of the objects in a subsystem as a series of interconnected vertices and edges.

Objects can be grouped into layers and subsystems using two operations: composition and union. Through composition, an object may import other objects, forming a new object that uses the imported objects and contains the types and operations of the imported objects. The union operation provides a mechanism to connect two or more independent objects, forming a network of communicating objects. A primary example of the union of two objects can be seen in the client-server model of subsystem interconnectivity.

Applicability or relationship of work to dissertation topic

The basic concept presented for the author's Computer Integrated System (CIS) is similar to the concept being explored for the integration of legacy, Commercial-Off-The-Shelf and Government-Off-The-Shelf software systems. The characteristics of CIS's provided:
"1. A CIS consists of heterogeneous subsystems which have been designed and manufactured according to different design principles by different vendors.
2. One or more components of a CIS can be added or deleted depending on the needs without damaging the integrity of the entire system.
3. Each subsystem is autonomous and capable of communicating with other subsystems.
4. A CIS could merge into a larger system." p.160
are similar to the characteristics envisioned for our integrated network of legacy systems. Therefore, the use of objects may be applicable as the basis for our approach as well.

Weaknesses

Other than illustrate how to view the CIS as a network of objects which are computing agents capable of communicating with other agents, the authors provide little insight into how differences in representation of data objects, my research problem, might be resolved.

V. Tentative chapter outline for dissertation.
   A. Chapter 1 – Introduction.
      1. Motivation- Mission Planning Example.
      2. Interoperability defined.
      3. Data representational differences between systems.
   B. Chapter 2 - Assessment of previous work.
      1. Early approaches to database interoperation.
      2. Other Interoperability Methods / Data Representation Methods.
      3. Methods for discovering relationships between data types.
   C. Chapter 3 – Development of general formal model for establishing relationships between data type representations.
      1. Consolidated Type Hierarchy defined.
      2. Using XML to represent Consolidated Type Hierarchy.
      3. Constructing Consolidated Type Hierarchy for federation of interoperating systems.
   D. Chapter 4 – Consolidated Type Hierarchy development automation.
      1. Areas for application of computer aid to the development of the Consolidated Type Hierarchy include:
         a. registration of producer and consumer types.
         b. discovery of relationships between producer and consumer types.
         c. development of translations to convert a producer representation of a data element to its appropriate consumer representation.
   E. Chapter 5 – Use of Consolidated Type Hierarchy for automated reconciliation of data element representational differences between heterogeneous systems.
      1. Message-based architecture translator.
      2. Publish/subscribe architecture translator.
   F. Chapter 6 - Conclusion, and recommendations for future work.

VI. Research plan and proposed schedule.
   A. Written Qualification Exam December, 1999
   B. Completion of Minor Area of Study (Computer Science):
      1. CS3650 Design and Analysis of Algorithms March, 1999
      2. CS3601 Theory of Formal Languages and Automata March, 1999
      3. CS4550 Computer Networks II December, 1999
      4. CS4800 Directed Studies (Computability Theory and Computational Complexity) March, 2000
      5. CS3450 Operating Systems March, 2000
   C. Complete assessment of previous work December, 2000
   D. Oral Qualification Exam January, 2001
   E. Advancement to Candidacy January, 2001
   F. Dissertation Defense August, 2001
   G. Graduation September, 2001

VII. List of references.

[ABK00] Anderson, R., Birbeck, M., Kay, M., et al., *Professional XML*, Wrox Press Ltd., Birmingham, UK, 2000.

[Bek97] Beker, H., "Graphical Embedded Real-Time Systems," *Dr. Dobb's Journal*, April 1997, pp. 54-65.

[BL91] Berzins, V., Luqi, *Software Engineering With Abstractions*, Addison-Wesley Publishing Company, Redding, MA, 1991.

[Blo92] Bloomer, J., *Power Programming with RPC*, O'Reilly & Associates, Inc., Sebastopol, CA, 1992.

[BPS98] Bray, T., Paoli, J., Sperberg-McQueen, C.M., "Extensible Markup Language (XML) 1.0 W3C Recommendation 10 February, 1998", http://www.w3.org/TR/1998/REC-xml-199802, 1998.

[CMK98] Cho, I., McGregor, J.D., Krause, L., "A Protocol Based Approach to Specifying Interoperability Between Objects", *Technology of Object-Oriented Languages, 1998.* TOOLS 26. Proceedings, 1998, pp. 84-96.

[CS91] Castellanos, M., Saltor, F., "Semantic Enrichment of Database Schemas: An Object Oriented Approach", *1st International Workshop on Interoperability in Multidatabase Systems*, IEEE Press, 1991.

[FHM91] Fang, D., Hammer, J., McLeod, D., "The Identification and Resolution of Semantic Heterogeneity in Multidatabase Systems", 1st International Workshop on Interoperability in Multidatabase Systems, IEEE Press, 1991, pp. 136-143.

[GSC95] Garcia-Solaco, M., Saltor, F., Castellanos, M., "A Structure Based Schema Integration Methodology", Proceedings of the Eleventh International Conference on Data Engineering, IEEE Press, 1995, pp. 505-512.

[Her97] Herman, J.S., *Improving Syntactic Matching for Multi-level Filtering*, Master's thesis, Naval Postgraduate School, 1997.

[HM99] Hammer, J., McLeod, D., "Resolution of Representational Diversity in Multidatabase Systems", *Management of Heterogeneous and Autonomous Database Systems*, Morgan Kaufman, 1999.

[HMS94] Hammer, J., McLeod, D., Si, A., "Object Discovery and Unification in Federated Database Systems", University of Southern California publications web site.

[HMS94a] Hammer, J., McLeod, D., Si, A., "An Intelligent System For Identifying and Integrating Non-Local Objects In Federated Database Systems", University of Southern California publications web site.

[HR98] Hasselbring, W., Roantree, M., "A Generative Communication Service for Database Interoperability"

[KGF98] Karunaratna, D.D., Gray, W.A., Fiddian, N.J., "Exploitation of Database Meta-Data in Assisting Database Interoperation", IEE Colloquium on Multimedia Databases and MPEG-7, Ref. No. 1999/056, pp. 12/1- 12/5.

[KLB93] Krämer93, Luqi, Berzins, V., "Compositional Semantics of a Real-Time Prototyping Language", *IEEE Transactions on Software Engineering*, Vol. 19, No. 5, May 1993, pp. 453-477.

[KM98] Kahng, J., McLeod D., "Dynamic Classificational Ontologies: Mediation of Information Sharing in Cooperative Federated Database Systems", *Cooperative Information Systems, Trends and Directions*, Academic Press, 1998.

[Las98] Lassila, Ora, "Web Metadata: A Matter of Semantics", *IEEE Internet Computing*, Vol. 24, July-August 1998, pp. 30-37.

[LB88] Luqi, Berzins, V., "Rapidly Prototyping Real-Time Systems", *IEEE Software*, September 1988, pp. 25-36.

[LBY88] Luqi, Berzins, V., Yeh, R., "A Prototyping Language for Real-Time Software", *IEEE Transactions on Software Engineering*, Vol. 14, No. 10, October 1988, pp. 1409-1423.

[LG97] Luqi, Goguen, J., "Formal Methods: Promises and Problems", *IEEE Software*, January 1997, pp. 73-85.

[LK88] Luqi, Ketabchi, M., "A Computer-Aided Prototyping System", *IEEE Software*, March 1988, pp. 66-72.

[LR91] Luqi, Royce, W., "Status Report: Computer-Aided Prototyping," *IEEE Software*, November 1991, pp. 77-81.

[LSH91] Luqi, R. Steigerwald, G. Hughes, F. Naveda, and V. Berzins, "CAPS as a Requirements Engineering Tool", *Proceedings of Requirements Engineering and Analysis Workshop*, Software Engineering Institute, Carnegie Mellon University, March 12-14, 1991, Pittsburgh, Pennsylvania, pp. 1-8.

[Luq89] Luqi, "Software Evolution Through Rapid Prototyping," *IEEE Computer*, May 1989, pp. 13-25.

[Luq90] Luqi, "A Graph Model for Software Evolution," *IEEE Transactions on Software Engineering*, Vol. 16 No. 4, August 1990, pp. 917-927

[Luq92] Luqi, "Computer-Aided Prototyping for a Command-And-Control System Using CAPS," *IEEE Software*, January 1992, pp. 56-67

[Luq93] Luqi, "Real-Time Constraints in a Rapid Prototyping Language," *Computer Language*, Vol. 18. No. 2, pp. 77-103.

[LS96] Luqi, Shing, M., "Real-Time Scheduling for Software Prototyping," Journal of Systems Integration, 6, 15-17 (1996), pp. 41-72.

[MIR93] Miller, R.J., Ioannidis, Y.E., Ramakrishnan, R., "Understanding Schemas", Proceedings of Third International Workshop on Research Issues in Data Engineering-Interoperability in Multidatabase Systems '93, 1993, pp. 170-173.

[Ngu95] Nguyen, D., *An Architectural Model for Software Component Search*, Ph.D. dissertation, Naval Postgraduate School, 1995.

[Pit97] Pitoura, E., "Providing Database Interoperability through Object-Oriented Language Constructs", Journal of Systems Integration, Volume 7, No. 2, August 1997, pp. 99-126.

[Pop98] Pope, A., *The CORBA Reference Guide*, Addison-Wesley Longman, Inc., Redding, MA, 1998.

[PR89] Pridmore, J., Rumens, D.J., "Interoperability- How Do We Know When We Have Achieved It?", Third International Conference on Command, Control, Communications and Management Information Systems, 1989, pp. 192-205.

[RK00] Regep, D., Kordon, F., "Using MetaScribe to Prototype an UML to C++/Ada Code Generator, *Proceedings Eleventh IEEE International Workshop on Rapid System Prototyping*, June 21-23 2000, pp. 128-133.

[RKM97] Roantree, M., Keane, J., Murphy, J., "A Three-Layer Model for Schema Management in Federated Databases", *Proceedings of the Thirtieth Hawaii International Converence on System Sciences*, Volume 1, 1997, pp. 44-53.

[RL] Ramesh, B., Luqi, "Process Knowledge Based Rapid Prototyping for Requirements Engineering",

[Ros98] Rosenberger, J., *Teach Yourself CORBA in 14 Days*, Sams Publishing, Indianapolis, Indiana, 1998.

[RN91] Rossak, W., Ng,. P.A., "Some Thoughts on Systems Integration: A Conceptual Framework", *Journal of Systems Integration*, Volume 1, Number 1, July 1991, pp. 97-114.

[SEK00] Shing, M., Eagle, C., Kreeger, G., Duranlioglu, I., "The Evolution of a Distributed Computer-Aided Prototyping System", Draft- Submitted to the RSP'2000.

[Sin98] Singh, N., "Unifying Heterogeneous Information Models", *Communications of the ACM*, Volume 41, Number 5, May 1998, pp. 37-44.

[Ste91] Steigerwald, R. A., *Reusable Software Component Retrieval via Normalized Algebraic Specifications*, Ph.D. dissertation, Naval Postgraduate School, 1991.

[Wal98] Walsh, N., "What is XML?", http://www.xml.com/pub/98/10/guide1.html, 1998.

[WO00] Wing, J.M., "Respectful Type Converters", *IEEE Transactions on Software Engineering*, Volume 26, Number 7, July 2000, pp. 579-593.

[YK96] Yoon, D.H.H., King, L.S., "An Object-Oriented Approach to Computer Integrated Systems", *Journal of Systems Integration*, Volume 6, Number 3, August 1996, pp. 159-179.

[YS94] Yellin, D.M., Strom, R.E., "Interfaces, Protocols, and the Semi-Automatic Construction of Software Adaptors"

[YS97] Yellin, D.M., Strom, R.E., "Protocol Specifications and Component Adaptors", *ACM Transactions on Programming Languages and Systems*, Volume 19, Number 2, March 1997, pp. 292-333.

INTERCONNECTIVITY VIA A CONSOLIDATED TYPE HIERARCHY AND XML

Brian J. Lyttle-Captain, United States Army
B.S., United States Military Academy, 1992
Master of Science in Computer Science-March 2001
and
Todd P. Ehrhardt-Lieutenant, United States Navy
B.S., San Jose State University, 1993
Master of Science in Software Engineering-December 2000
Co-Advisor: Valdis Berzins, Department of Software Engineering
Co-Advisor: Ge Jun, Department of Software Engineering
Second Reader: Paul E. Young, Department of Software Engineering

We propose building a software system that passes any message type between legacy Command, Control, Communications, Computer, Intelligence, Surveillance and Reconnaissance (C4ISR) systems. The software system presents significant cost savings to the Department of Defense (DoD) because it allows us continued use of already purchased systems without changing the system itself.

In the midst of the information age, the DoD cannot get information to the warfigher. We still maintain and use heterogeneous legacy systems, which send limited information via a set of common messages developed for a specific domain or branch of DoD. Our ability to communicate with one message format does not meet our needs today, though these stovepipe C4ISR systems still provide vital information. By combining these systems, we will have a synergistic effect on our information operations because of the shared information.

Our translator will resolve data representational differences between the legacy systems using a model entitled the Common Type Hierarchy (CTH). The CTH stores the relationships between different data representations and captures what is needed to perform translations between the different representations. We will use the platform neutral eXtensible Mark-up Language (XML) as an enabling technology for the CTH model.

DoD KEY TECHNOLOGY AREA: Command Control and Communications, Computing and Software

KEYWORDS: Interoperability, Interconnectivity, Legacy Systems, XML, Consolidated Type Hierarchy, Information Systems

I. INTRODUCTION

In today's combat environment, the United States military and its allies find themselves in the midst of the information age they helped start. Information and systems that use information abound in all parts of the services and all locations on the globe. No longer can the side with the best trained and best equipped force be confident of victory. If an opponent can conduct efficient information operations, they have a significant edge. An important fact is that information operations take place throughout the spectrum of combat, from peacetime operations such as refugee relief to armed conflicts similar to Operation Desert Storm. This fact implies we will always conduct information operations, regardless of the place or time.

Information operations are "Actions taken to affect adversary information and information systems while defending one's own information and information systems." [DTIC] Information systems are normally the computer systems that receive, manipulate, and disseminate information. From this definition of information operations we realize these operations are both offensive and defensive in nature. An astute information operator could use propaganda in an offensive manner to destroy the public support of his enemy. Or, the operator could publish incorrect information about an operation in order to deceive

the enemy. Properly conducted, information operations are a powerful combat force multiplier that can significantly increase our ability to shape the environment and influence decisions at all levels of combat.

To influence decisions, commanders and their staffs need the most up-to-date information available. This information comes from many different sources, but especially from computer systems. The Department of Defense (DoD) developed many of these computer systems over the last few decades before interoperability became a concern. Often systems cannot pass information to each other because they use incompatible message sets.

One agency within DoD that tries to solve joint war-fighting problems is the U.S. Joint Forces Command (JFCOM). A subordinate element of JFCOM is the Joint Battle Center (JBC) in Suffolk, Virginia, which tries to resolve Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) issues, especially between the various information systems. Part of their C4ISR involvement is the assessment of new technology to solve interoperability problems between the services.

Many of the established information systems use message formats that possess a structured, though limited method of communication. Information is passed via a set of messages contained in a message set. These sets are rigid by design

and cannot be changed. However, one format cannot satisfy the needs of the entire DoD, not to mention our allies.

Commanders need all possible information in order to make accurate and timely decisions. The various information systems contain valuable data, but it cannot reach the commander because of incompatible data formats between information systems. Thus, there exists a need to increase the flow of information to the commanders, yet save development time and costs due to budget constraints. We believe DoD can continue to use the legacy systems if some method is developed that allows message passing between the computer systems.

We seek to design a format that bridges the differences between all the message formats called the Consolidated Type Hierarchy (CTH). The CTH is formed from all the message formats contained in the network of information systems, thus allowing a free-moving flow of information to all systems that desire it.

One new technology that has emerged recently is the eXtensible Mark-up Language (XML). With roots in the publishing industry (the Standard Generalized Markup Language), XML is now used by the e-commerce industry to allow interoperability between a variety of databases in a near-real time manner. Though these applications are business oriented, the application of XML shows great

promise in solving some of the DoD interoperability problems. We used XML to implement the CTH in our thesis.

By using the CTH model, we believe DoD can start integrating the legacy computer systems with significant cost savings. Our results on a small set of messages show the concept has promise and hope for interoperability.

II. CURRENT STATE OF AFFAIRS

One of the main difficulties in information operations is the task of getting relevant information to the user in the correct format. Many of our current systems are heterogeneous systems that do not communicate outside of their own format. Thus, we need the ability to share data with computer systems that were developed for diverse user communities with very different data needs and requirements. We are currently limited to sending text messages common to the various computer systems, and some systems cannot even do that.

A. A MEGAPROGRAM

We can think of attempts to continually use legacy systems and their information as an example of megaprogramming[GW92]. Megaprogramming is a concept developed by the Defense Advanced Research Projects Agency (DARPA) as part of an effort to reuse systems that already exist. A megaprogram is a software program that utilizes commercial off the shelf (COTS), and government off the shelf (GOTS) software systems as if they were modules. The modules, or megamodules as the authors call them, are internally homogeneous, independently maintained software systems managed by a community with its own terminology, goals, knowledge and programming traditions. We call the

concepts, terminology, and interpretation associated with each domain specific megamodule an ontology.

Unlike the distributed federated databases used in [GW92], our legacy system megamodules possess only the ability to export information through a set of standardized messages. This constitutes a key difference between tying together legacy systems and the megaprogramming previously envisioned. Megaprogramming relies heavily on databases to furnish the ability to import and extract data from the heterogeneous systems, whereas our system must rely on the information sources to push the information out. We have no mechanism to actively query or pull information from the source. This limits our ability to access information within the megamodule.

Because some systems cannot automatically extract data from a distant machine, they are reliant on other machines to send regular updates consisting of any new data they find. This feature is unfortunate because the remote systems are not always configured to meet the needs of the other systems. In some cases operator action is required to send and receive information from the source. System operators must then rely on standard operating procedures (SOPs) for regular updates of information outside of our local system. This does not agree with the mega-programming concept as stated in the paragraph above. This makes reuse

of legacy systems a limited example of mega-programming, but still useful.

B. MESSAGE FORMATS

In previous years, information systems defined a set of messages for each system. This set of messages contained the information most commonly needed by consumer systems, and was often domain specific. One common message format used by many systems is the United States Message Text Format (USMTF). The U.S. and our NATO allies used USMTF to increase our ability to communicate tactical and other information. The format of USMTF is well established, but its fixed field format wastes bandwidth by sending empty information. Because USMTF messages require larger bandwidth capabilities than most land forces possess, the land forces use variants of USMTF. USMTF may also provide more information than the destination system needs.

Coalition Information eXchange (CIX) is a newer data message format constructed by Defense Information Systems Agency (DISA) with more capabilities than the Over the Horizon Gold (OTH-G) message format used by the Navy and Marine Corps. However, unless the receiving system can translate from CIX, the information is unused and useless.

To communicate between different message formats such as CIX and USMTF, current implementations use software programs called translators. The translator alters a system

message from one legacy computer system format into another format for a different legacy system. The translator is implemented via a third generation language such as C++ or Ada. Providing some way for different existing systems to share data presents an opportunity to save significant development costs in the design of replacement systems built to share data. Enabling systems to share data also saves end-user time, since data does not have to be entered by hand from one format or system into another.

However, making translators is a time consuming task when constructed manually.[Sin98] The programmer must map the systems' message types, find corresponding messages, find data within the message that can translate between systems, and finally code the translator from scratch. Once completed, the translator only works from one message format to another specific message format. Although these translators are better than the manual transportation of data between systems, their creation is time consuming and of limited use. Each translator is expensive because of the specialized knowledge contained in the two systems. This also causes maintenance problems when the programmer leaves or a heterogeneous system changes its message format.

At this time, we do not possess an automated way of resolving representational differences between systems. Thus, the programmer must still complete the mapping by hand. We seek to construct a translator that uses a pre-

runtime developed framework to perform run-time message translation. This method would enable reuse of common translation routines, and would be able to translate messages among many different formats.

C. BACKGROUND RESEARCH

Part of our research revealed the similarities between integration of heterogeneous databases and legacy system integration. Since message formats share data among systems, we can consider messages to be results from a database query. Many current commercial databases share data between heterogeneous systems connected via networks. Reconciling differences between databases must be done over several levels.

At the highest level, databases must be reconciled over different schemas. Database schemas define the structure of the data, and how each piece of data is related to each other, how it's organized. The differences include resolving the representations between the tables found in each database.[HMS] This representational heterogeneity is defined as "variations in the meaning in which data is specified (for the data) and (the way it is) structured in different components". It is a natural consequence caused by creating independent data structures.[HM99]

The next level of reconciliation involves the naming conventions used in each database. A major cause of

conflict is the use of homonyms and synonyms. Homonyms use the same word for different concepts, such as "fire." In one context, the phrase results in artillery rounds impacting on a target, while in another context, the phrase summons the fire department. Synonyms describe the same object, but use different terms. Soldiers commonly use position and location to mean the same place.

Representational differences make up a third level for reconciliation. As shown in Figure 2-1, one community may define a geospatial position using the Military Grid Reference System, while another defines the position using a latitude/longitude representation. Both methods define the same real world object, but implement different methods and possess different attributes.

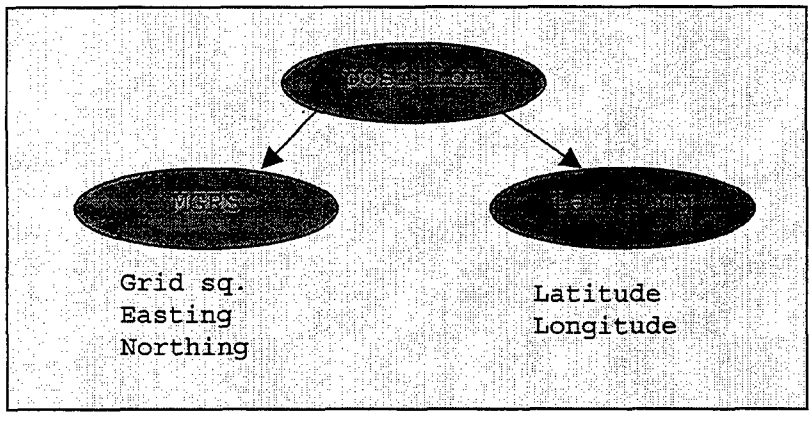

Figure 2-1 Different representations of the same location

Some other causes of differences in data representation include the low-level format of the data, such as precision or units of measurement.[KM98] Another cause is the range of values for a data type, which may vary from system to

system depending on the needs of the user and the hardware and software the user possesses. Older systems cannot represent larger numbers due to the size of the allocated memory or the processor used in the hardware.

D. PREVIOUS ATTEMPTS

Because of the many different systems and formats we are looking for a systematic way to construct translators, which opens the door to automation. This will save time, money, and results in more reliable communications.

In our search for a solution to the problem, we found several systems that try to achieve similar results.

One thing that almost all these systems or models have in common is the use of some kind of universal representation of data, or some universally agreed upon vocabulary. Most systems have these universally accepted terms and build on that in different ways.

1. Canonical Data Model

Roantree, Keane, and Murphy call their universal model a Canonical Data Model (CDM). This is similar to a universally agreed upon representation for a location. They introduced a model containing three layers. From top to bottom, the layers are: the Federation Layer, the Component Layer, and the Integration Layer. They use the lowest layer to isolate the effects of changes in a member database. The Integration Layer changes with the database in order to

maintain a consistent interface with the upper layers. Any time a change is made in the design or schema of a particular constituent database, its corresponding integration layer changes. [RKM]

2. Metadata

Another approach presented by Narinder Singh is to use metadata, which is information about data, to dynamically determine how to respond to a query. In this system, information providers must supply a description of the information they have to offer in terms of a standard vocabulary. This standard vocabulary is a list of universally agreed upon set of words, each word having a single meaning. Middleware provides access to the data sources. When a query is submitted from a user, the Tesserae Integration Engine dynamically creates a search plan and retrieves the information. [Sin98]

One drawback to this system is the time cost of creating a search plan on the fly. In a dynamic environment such as the web, the benefits would outweigh the costs; but, in our context there is no advantage to creating a search plan.

These previous methods have their merits, and we have tried to incorporate some of their achievements into our system. For example, it is apparent that in order to reconcile information from different databases, there has to

be at least some a priori agreement on what some of the terms mean. However, our context is different from the typical scenario in which databases are being integrated, since we don't have the ability to query data sources, and we don't want to assume the existence of a central data store.

E. RESPECTFUL TYPE CONVERSION

One of the most pertinent articles to our research is a paper written by Jeannette Wing and John Ockerbloom [JMJO00]. Their paper discussed the conversion of different types in such a manner that no data was lost. This pertains directly to interoperability because of the problems associated with data differences.

In their paper, Wing and Ockerbloom assume a normal subtype and supertype inheritance relationship, and call an instance of a type an object. The types follow what is known as the Liskov substitution principle, which is outlined in the article. The Liskov substitution principle says that the subtype inherits the attributes of the supertype, and an instantiated object of the subtype acts the same as the supertype when the supertype's method is invoked. A *respectful type converter* will convert two subtypes with a common supertype ancestor while preserving the behavior observable through the interface of the common ancestor supertype. [JMJO00]

Wing and Ockerbloom recognize type hierarchies may solve many interoperability issues by reducing the number of translators required from N^2 to 2*N translators.[JMJO00] They base their examples on an assumption that only one type will exist per file, which is unlikely to occur in our messaging system. A message may contain a position and a text message that have different supertypes. Unlike the paper, we must construct translators that contain many different functions because our messages will contain many different types.

Additionally, our system cannot actively retrieve information because of how the message systems are constructed. Rather, the information providers will push their data, as opposed to the data being pulled from its source. Therefore, a system that derives a search plan would not be appropriate.

F. THE EXTENSIBLE MARK-UP LANGUAGE (XML)

In order to construct our program, we needed a method that allowed us to express information in a manner independent of any platform yet still capture the meaning of the data. We found the eXtensible Mark-up Language (XML) met these criteria. Since XML is a fairly new language, we searched for current examples that utilized XML commercially and in DoD. In order to understand these examples and our thesis, we must first explain what XML is.

1. Meta-Language

XML is a meta language, which means it describes the data contained inside an XML document. XML separates the content of the document from the presentation of the data, which enables more programs to read the document. [PROXML] The separation occurs because XML only provides the means to describe the data, leaving presentation of the data to the receiver.

Mark-up tags surround the data in an XML document. The tags are very similar to Hyper-Text Mark-up Language (HTML) tags, with an important exception. While XML tags may use all but a small set of characters, HTML tags are predefined and restrictive. Unlike XML, the HTML language possesses functions that tell an HTML browser how to display the data.

Figure 2-3 is an example of how an XML document could describe a person. Note the document root mark-up tag entitled people, and how it surrounds the nested elements.

```
<people><!--This is a comment block-->
      <person>
         <firstName>Brian</firstName>
         <middleName>John</middleName>
         <lastName>Lyttle</lastName>
      </person>
      <person>
</person><!--This is an empty person element using open and close
            tags-->
</people>
```

Figure 2-3 Sample XML Document

2. XML Trees

XML works by forming a tree from the data contained in the XML document. The document must possess a root node in order for the parser to construct a tree from the elements within the document. Elements may be nested repeatedly beneath the root node, and may contain duplicate element names at the same level within the tree.

XML contains a powerful concept called a namespace that effectively allows homonyms. The namespace allows the writer to use the same name but with different associations, provided the writer distinguishes the namespaces. This allows the transformations and formatting functions at each viewer's platform to take the appropriate actions when parsing the document tree. [PROXML]

3. Parsers

In order to take actions on an XML document, we must be able to construct the tree in memory. The software program that constructs the document tree is called a parser. It is not responsible for presenting data to the user, unlike HTML. The parser ensures the document is "well-formed", which means the document obeys the XML syntax rules. XML parsers are powerful tools freely available from several sources. Both Internet Explorer 5.0 and Netscape's Mozilla 6.0 contain XML parsers in addition to HTML parsers. The IBM Apache Group (http://www.apache.org) wrote and provided

the source code for their Xerces processor for anyone to utilize for free. The Xerces parser is written in both C++ and Java, and is available for a variety of operating systems to include Windows, Linux, Unix, AIX, and Sun Solaris. The Xerces parser is the official parser of the World Wide Web Consortium (W3C) at this time, and is fully compliant with the approved W3C recommendations. It does not expand upon the approved requirements of the W3C for XML.

4. Validation

All of the parsers mentioned above are examples of a validating parser. Validating parsers verify the XML document obeys more stringent rules than the generic XML syntax. These rules are specified in a Document Type Definition (DTD) or a Schema. DTDs and schemas allow us to specify rules about what elements may appear in a document, the structure of the tree, and to a limited extent, what format (e.g. the order and number of occurrences) the elements must follow. DTDs and schemas serve the same purpose. They were designed to facilitate content checking, to some degree. Obeying the DTD ensures all users of our namespace can read our document using the same standard. The DTD is a W3C recommendation; schemas are only a W3C candidate recommendation. According to the W3C, "a Candidate Recommendation is work that has received

significant review from its immediate technical community. It is an explicit call to those outside of the related Working Groups or the W3C itself for implementation and technical feedback." Also, "a Recommendation is work that represents consensus within W3C and has the Director's stamp of approval. W3C considers that the ideas or technology specified by a Recommendation are appropriate for widespread deployment and promote W3C's mission." [W3C] However, schemas were designed to make up for some of the shortcomings of DTDs; and tools that support schemas are already on the market.

Schemas have several advantages over DTDs. Schemas allow open content models. An open content model provides extensibility to a schema. This means that I can reuse someone else's schema. If their schema doesn't contain all the elements I want to include in my schema, I can add elements. This allows greater reuse of schemas. Open content models are optional; however, and a closed content model can be specified in a schema if desired.

Schemas also provide some support for data types. Data types can be specified for elements and/or attributes. Beyond the typical data types found in common programming languages, the following data types are some of those supported: string, id, idref, nmtoken, nmtokens, entity, entities, enumeration, and notation.

Other advantages of schemas [MSDN]:

- Greater specificity of the number of occurrences of an element.
- Ability to specify if sub-elements must appear in a certain order.
- Accessible from Microsoft's Document Object Model.
- Schemas are well-formed XML documents (unlike DTDs, which have their own syntax).

We believe that although schemas are relatively new, their additional capabilities provide them a substantial advantage over DTDs. We recommend the use of schemas.

5. Transformation

If two users have different formats for their data, like many Defense organizations, we can transform the XML document using the eXtensible Stylesheet Language Transformation (XSLT). XSLT enables us to translate between vocabularies as well as merge existing resources. We can determine the correct stylesheet to use at runtime to dynamically translate between documents. We do not have to write procedural language code for most applications, although it may be necessary in some cases.

Stylesheets provide a major contribution toward achieving our goals. They are a part of the XML world, and as such, share many of the same benefits. They can be transferred using the ubiquitous hypertext transfer protocol (HTTP). They can be applied to XML documents by the XML

processors. The XML processors are COTS, and are available for free. Stylesheets can also refer to other stylesheets. Therefore, they can be used and reused in a modular way, also providing cost savings.

Internet Explorer 5.0 and the MSXML 3.0 parser allow the programmer to write procedural JavaScript functions in order to assist with transformation. We have not found any other free commercially available parsers that allow us to do this in a packaged format, though we can construct a parser from source code like Xerces and write functions in the same manner.

However, this requires a compiler for each target machine for the functions each programmer may write. Parsers perform much of the work contained by the XML language, and a good working parser should not be modified greatly. The commercial parsers such as Internet Explorer and Mozilla provide the functionality we need for this demonstration.

III. XML USAGE EXAMPLE SYSTEM

A. THE JBMI EXPERIMENT

One organization with XML experience is the Joint Battle Center (JBC) based in Suffolk, Virginia. JBC is part of Joint Forces Command (JFCOM), and is charged with finding joint solutions for Command, Control, Communications, Computer, Intelligence, Surveillance, and Reconnaissance Systems (C4ISR) inter-operability. In order to fulfill this mission, they conduct experiments with several organizations each year.

We witnessed Phase Two of an experiment entitled the Joint Battle Management Initiative (JBMI). JBMI sought to prove XML is a valid technology for improving inter-operability and inter-connectivity between systems. All four services provided computer systems for the experiment.

JBC defined two different levels of sharing information between systems in accordance with the Defense Information Infrastructure Common Operating Environment (DII COE). Interoperability at its highest level allows systems to import and export information as if the remote site were actually part of the user's system. Inter-connectivity is several steps lower, and allows systems to pass limited messages between different systems.

The computer systems at JBMI accurately reflected the problem in DoD today. The primary system was the Global Command and Control System (GCCS), which controls high level operational units across DoD. It specifically targets units the equivalent of an Army Brigade level or higher. It utilizes CIX as its means of message passing. The Navy and Marine Corps also sent their versions of GCCS, which are compatible with the other services' GCCS systems.

The U.S. Army provided a system entitled the Advanced Field Artillery Tactical Data System (AFATDS). AFATDS is a member of the Army Battle Control System set, and is the command and control system for all ground fire-support systems in both the Army and Marine Corps. AFATDS also interacts with our English and German allies using its own specific format developed many years ago. It can send and receive a limited number of USMTF messages.

In an interesting twist, JBC integrated two devices currently available on the commercial market. The first was a Palm Pilot V, which is a personal digital assistant. JBC programmed the simple USMTF Call for Fire and Observation Report messages into the PDA. They programmed the same ability into a cellular telephone, and communicated using the Wireless Application Protocol to the networked systems.

All the systems connected via a hardwire LAN into a web server. The web server allowed each unique system to subscribe to a message set or an individual message type

from the USMTF. As each legacy system produced a message, a software wrapper transformed the message into an XML formatted message. It then sent the XML mark-up message to the web server.

The web server received the message and removed the XML mark-up from the message. It parsed the message to discover the USMTF message type. The server then found a data directory specific to that message type, and saved the message. A Visual Basic monitor script periodically checked the directories for new information. If the monitor found new information, it checked a database to discover subscribers of that message type.

If a subscriber was found, it called upon functions constructed in Java code to transform the message into the appropriate type. If the destination system required the message in the HTML format, the XSLT processor was called to make the conversion. Most systems subscribed for an HTML representation of the USMTF message or email.

This system allowed the cell phone user to send a Call for Fire message to the AFATDS system via the web server. The AFATDS equipped unit could then provide indirect fire support onto the target. It also allowed the GCCS system to update its database, and the Air Force TCDB to enter the target information for use in plotting aircraft routes or further intelligence usage.

Other abilities included at this demonstration were comma-delimited files used in spreadsheets and word-processing documents. Since many of our allies do not have the funds required to make military specific information systems, they must rely on Commercial Off The Shelf (COTS) products.

An extremely useful application of COTS and XML was the target list used in the joint targeting process. Using AFATDS, a message containing a target list was sent to the web server. Upon receiving the message, the JBMI engine found the coalition subscribers that wanted a copy of the list. The engine translated the target list into a spreadsheet file, and sent it to the destination machine via email. Though the system lacked security restraints, it demonstrated the ability of XML to send various messages using COTS equipment.

Given the accomplishments of the JBMI engine, we knew XML presented a means to accomplish interoperability between systems. It allowed messages to transform from native legacy format into XML and then be used in a different system. However, the engineers were required to write source line code in Java to accomplish this. We believe using XML and other COTS tools along with a different methodology can accomplish interoperability between systems cheaper and faster than writing source code.

B. ASSUMPTIONS

We made several assumptions in our thesis. We assumed all the messages we received were well-formed XML documents and complied with a DTD for that specific message type. We assumed this because each system should send messages in the correct format, else it would not be fielded to the force. The parser would not read messages with incorrect formats because it would fail the validity check when a stylesheet or a DTD was applied to it. In a fielded system, a failed message would be returned to the sender with the appropriate error message. This service would take a small amount of time, and not impact the performance of the system. Additionally, we did not think we needed to check for transmission errors because the TCP/IP protocol stack conducts those error checks for us.

In our environment, we assumed an experienced software engineer would use the system. The messages will depart and arrive in an XML mark-up format of the original system message.

While we knew the translator system could be implemented either in a point-to-point system or in a publish/subscribe architecture, we chose to implement the point-to-point system. Although not as robust as the publish/subscribe architecture, the point-to-point implementation is sufficient as a first step for a proof of

concept. The point-to-point implementation can then form the basis for subsequent implementations. In the point-to-point system, each system possesses a copy of the translator and a means of communicating to the other system.

We assumed individual systems using this software would possess similar capabilities to our own, because our demonstration is based on the systems used by JBC during the JBMI exercise. That is, it would be a machine using Windows 95, Windows NT, or Windows 2000.

Given these assumptions and requirements, we can now describe the design of our system.

IV. THE CONSOLIDATED TYPE HIERARCHY

As we introduce you to the Consolidated Type Hierarchy (CTH), remember our goal: we are trying to achieve interoperability between legacy systems that have different views and representations of data. Our general approach is to set up a common framework that we can use in matching data sources with potential consumers. Translations will be defined in terms of the framework before run-time, and will be applied at run-time. Since the legacy systems we have in mind traditionally have shared their data through messages, we will consider the message formats they use rather than the data stores internal to the systems themselves. Before we explain what the CTH is, we will discuss what we need in order to create a CTH, the environment.

A. THEORY

1. System Schemas

Schemas provide a blueprint for the data to be shared. They can be thought of as Application Programmer Interfaces (APIs). Each message format will have its own schema. It is our way of knowing what data is contained within and provided by that data source or consumed by that recipient.

If we only had to be concerned with converting between two message formats, we could easily map data fields from one message format to the other. This simplified problem

would be trivial and not warrant further effort. However, as more formats are considered, the task becomes more complicated and requires considerably more work. If you had N different formats to reconcile with each other, N^2 direct mappings would be required. [JWJO00]

2. The Global Schema

The global schema is a global view of the data to be shared. It provides the context for data to be shared among systems. The elements of the system schema have a "kind-of" relationship with the elements of the global schema. For example, one element in the global schema might be a location. Although latitude-longitude and MGRS coordinates have different formats, they are both a kind-of location. They convey the same information.

The real purpose of the global schema is to capture the structure of composite types. If we were to send a list of locations, it would be meaningless. We must put information in its context. In other words, a position is an attribute of some other thing, like a ship, a tank, or an aircraft route. The global schema captures the contexts in which it is used.

3. Consolidated Types

Every element within the global schema is a consolidated type. In the example mentioned above, location

is a consolidated type and latitude-longitude and MGRS coordinates are legacy system subtypes.

Consolidated types are more than just an abstraction. Consolidated types must have a concrete representation in order to gain the advantages offered by having them. It's important to consider the physical representation of a consolidated type with care. Consolidated types are derived from pre-existing subtypes that are to be reconciled. Therefore, one method of choosing a representation would be to adopt the representation of one its subtypes. However, we would like to be able to convert from a subtype to the consolidated type and back to the same subtype without losing any information. Consequently, is important to select the representation with the highest degree of precision.

4. The CTH

The global schema represents a global view of information that is to be exchanged. It is a bridge format, which reduces the number of translations that must be defined. The elements of the global schema are consolidated types. The CTH does more than describe the structure of the global schema. It also contains the relationships between its elements and the elements of its constituent schemas. We introduce a separate term for the consolidated type hierarchy because neither the global schema nor its

elements capture both the structure of the consolidate types and their relationships with the elements of the various system schemas.

Now that we have explained the theory of the different parts and their relationships, it's time to look at how we implemented and integrated these pieces.

B. IMPLEMENTATION & EXAMPLE

We have created a simple example to illustrate how the different parts of our system fit together to achieve the desired result. In our example we have two message formats that we want to reconcile. We invented the message formats for the purpose of this example, but they are adequate to show the relationships between the different parts of our system and how they are used.

Both formats carry information about tactical units in a battlespace. The Army message format is designed to contain information about ground forces. Originally constructed as a voice message, it is now a standard digital message as well. The Navy message format contains data about ships sent via tactical data links from a variety of sensors. Both messages contain information about objects the operators are observing.

1. Schemas

The schemas were simply implemented as XML schemas. For our purposes, the essential requirement was to be able

to capture the structure of the data. This could have been done in many different ways, including UML diagrams. However, since DTDs or XML schemas can also be used for validating the XML documents, they might already exist for some systems and they could serve a dual purpose. We prefer the use of schemas over DTDs for reasons given in chapter 3, and our example uses XML Schemas.

Before we go further, we'd like to acknowledge a valuable tool we discovered in our research called XML Spy. XML Spy is the product of Altova GmbH, of Austria. It is an easy to use integrated development environment for XML, with authoring tools for XML documents, DTDs, schema, and style sheets. The product is available for download at www.xmlspy.com and free thirty day trial downloads are available. We used XML Spy for all the XML and related coding for our examples. We have included partial screen shots of the program in Figures 4-1 through 4-3 below. We are using the program to show the schema, because it can display them in a graphical representation, rather than having to look at the code; however the code is included in the Appendices.

The Army message format we called a SALUTE message. Figure 4-1 depicts the schema for the message format. The root element in the SALUTE schema is the element named SOURCE. The Type element contains information about the message type, and the GroundUnit element contains the

information on the ground units. Note the symbology depicts that there can be a sequence of GroundUnit elements contained in a valid XML document.

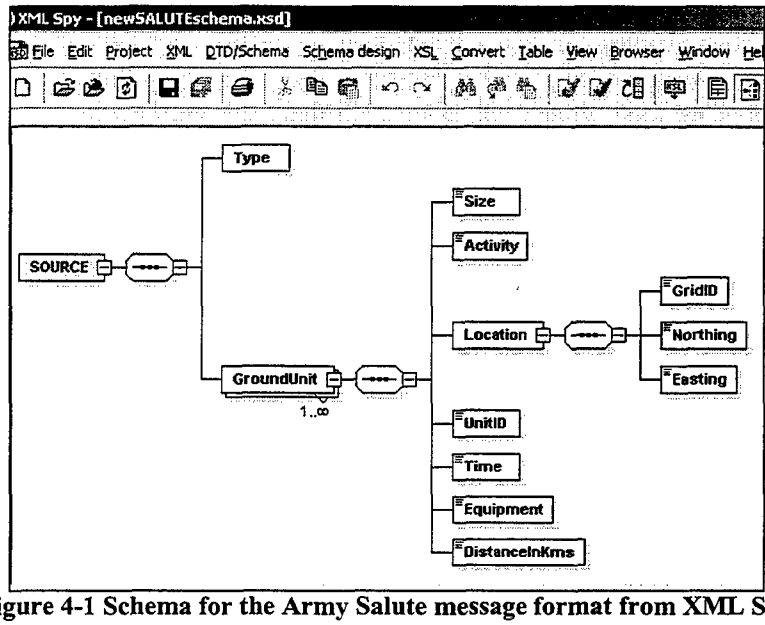

Figure 4-1 Schema for the Army Salute message format from XML Spy

Figure 4-2 depicts the Navy message format. It has some fields that will map to the Army message format, and some that do not.

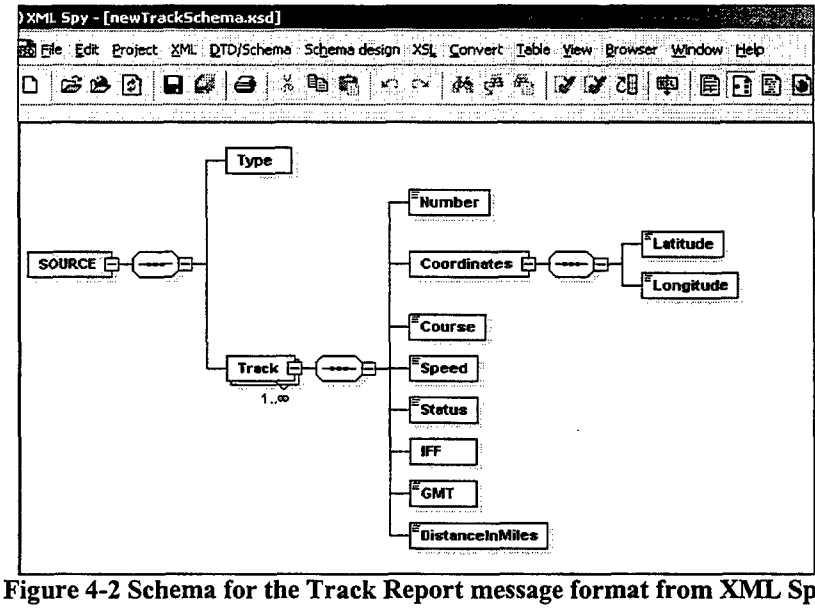

Figure 4-2 Schema for the Track Report message format from XML Spy

The global schema in Figure 4-3 depicts a composite view of the information provided by both message formats. Here you can see that Location is a consolidated type.

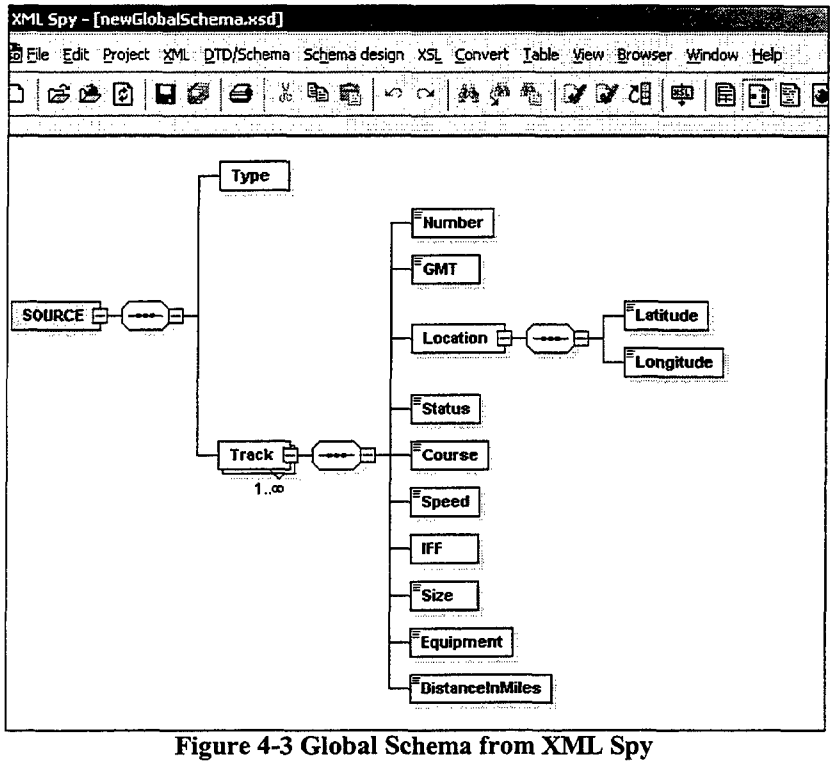

Figure 4-3 Global Schema from XML Spy

Also, notice that we included elements in the global schema, such as Course and Speed, which did not have a corresponding element in the Army schema. If a Navy system were to send a message to an Army system, the Army system has no use for such information. This begs the question, why include these elements in the global schema?

There are two reasons to include those elements in the global schema. The first reason relates to the comment we made earlier about choosing the representation with the greatest precision. If we convert a Navy message to conform

to the global schema without those elements, we would lose the Course and Speed information in the process. If we then convert it back to the Navy message format, we can't get that information back. We threw away that information. We would like to be able to convert from any system format to the global format and back without losing any information.

The second reason to include unique elements in the global schema is to make it easier to find compatible elements between schemas. Imagine that we decide to integrate a third message format into the global schema, and we left out Course, Speed and other elements unique to each of the preexisting Army and Navy schemas. If the new schema we want to introduce has elements that do correspond to the previously unique elements, we may never discover the correspondence, unless we also look for corresponding elements in the Army and the Navy message schemas. Instead, if we include all of the elements, then when we integrate a new schema, we will be able to discover the common information to be shared among systems, without having to analyze each system independently.

2. Consolidated Types

We captured the consolidated types in an XML document we named CT.XML. Pictorially, you can think of CT.XML as shown in Figure 4-4. Each root node represents a

consolidated type. Each child node depicts the corresponding element from a particular message format.

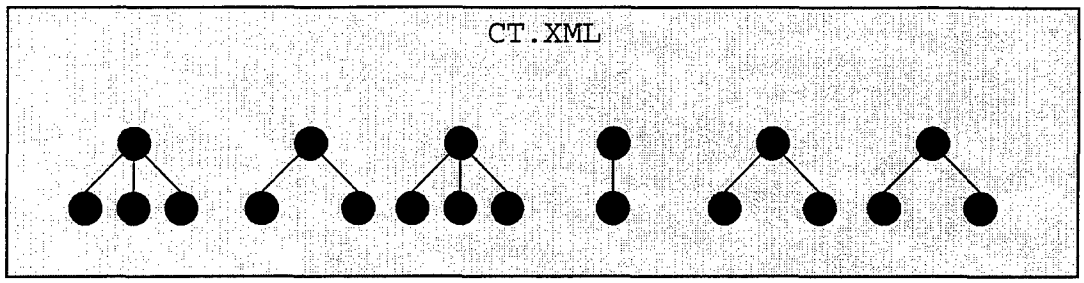

Figure 4-4 Symbolic view of CT.XML

Figure 4-5 is an excerpt from CT.XML, the XML representation of the consolidated type hierarchy. The full listing is included in Appendix F.

```
<Location>
   <TrackReport name="Coordinates"/>
   <Salute name="Location"
      upXlate="Grid2LatLong.xsl"
      dnXlate="LatLong2Grid.xsl"/>
</Location>
```

Figure 4-5 The consolidated type Location from CT.XML

Figure 4-5 shows how the consolidated type, Location, is entered. The outer-most element is the name of the consolidated type, which comes from the global schema. The nested elements name the message formats that have a kind-of Location. Since both track report messages and salute messages have attributes that are a kind-of location, they are both listed here. Each of the nested elements may have between one and three attributes. The name attribute specifies the name of the corresponding element in their respective message formats. The upXlate attribute contains the name of the style sheet that will translate from the

enclosing message format to the format of the consolidated type. The style sheet named in the dnXlate attribute will perform the reverse operation, taking an instance of a consolidated type, and transforming it to conform with a specific message format.

Like many other aspects of our implementation, there were alternate ways of implementing the mappings between message formats and the global schema. One disadvantage of the way we implemented it is that searching through CT.XML for the translations would be slow compared to other methods, such as a table lookup or database query. But, since CT.XML will be searched when the stylesheets are generated, which happens prior to run-time, the speed of the search will not affect run-time performance.

C. CTH USE

Figure 4-6 shows a conceptual view of the CTH. The Army schema is in the upper plane, and the global schema is in the lower plane. The dashed arrows represent the associations and the translations between elements in the global schema and the Army schema, information that is stored in CT.XML. We have only included the Army Salute schema in the figure in the interest of readability, but we could have presented another plane for the Navy Track Report schema as well.

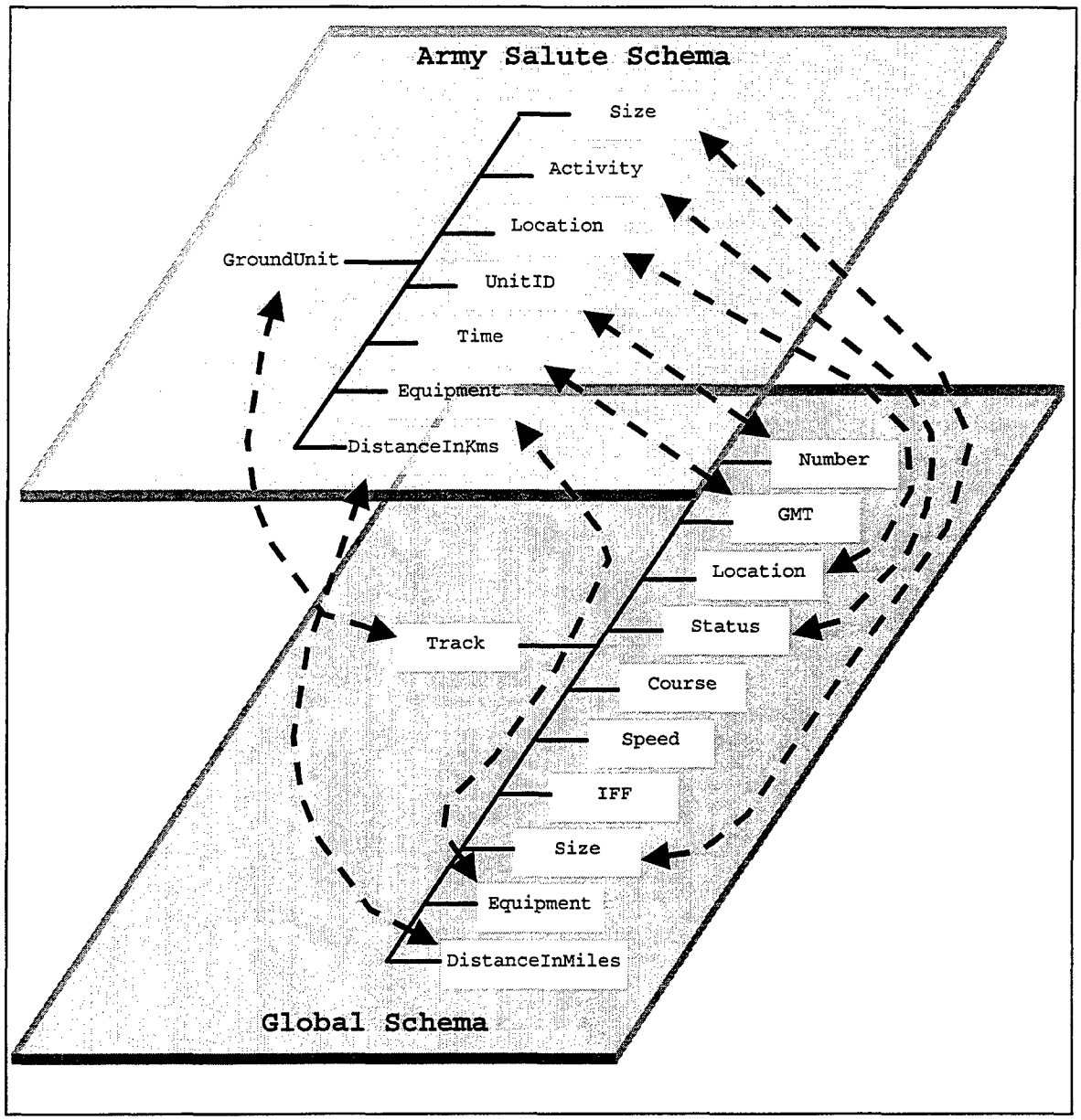

Figure 4-6 Conceptual View of the CTH. The Army schema is in the upper plane, and the global schema in the lower plane.

This is all we need to have a translator. When a translator receives a message it could determine the format, then recursively apply translations defined in the CTH by the arrows. Currently we create stylesheets before run-time based on the information contained in the CTH. At run-time we let the XSL processor act as our translator using the stylesheets to give it processing instructions.

1. Before Run-Time

a) Mapping

The CTH is a framework for matching potential data sources and consumers. It enables the sharing of that data, despite representational differences. When a system is introduced into a network, a schema for the data it exports and/or imports must be available or must be produced so that its elements can be mapped to the global schema. In our work, we performed this by hand.

In our system we generated the initial global schema from the Navy schema. Then we integrated the Army schema into this initial global schema. We will walk through the steps we followed during this process.

We started with the root element in the Army schema and looked for a corresponding element in the global schema. We descended through the structure of the Army schema, establishing these correlations at every level

possible. When we mapped the Army Schema to the global schema, we established these relationships:

Army Schema Element Name	Global Schema Element Name
GroundUnit	Track
Size	
Activity	Status
Location	Location
GridID	
Northing	
Easting	
UnitID	Number
Time	GMT
Equipment	
DistanceInKms	DistanceInMiles

Table 4-1 Initial Mapping of Elements in the Army Schema to the Global Schema

As you can see, Size and Equipment in the Army schema did not have corresponding elements in the global schema, so we added them to the global schema and we add them to CT.XML as consolidated types. GridID, Northing, and Easting also did not have corresponding elements in the global schema; however, we did not add those elements to the global schema as we did with Size and Equipment. This is where an engineer will have to decide whether to incorporate the elements into global schema, or define a translation at a higher level that will perform the conversion. Table 4-1 shows the mappings between the two schemas at this stage.

Army Schema Element Name	Global Schema Element Name
GroundUnit	Track
Size	Size
Activity	Status
Location	Location Translations: Grid2LatLong.xsl LatLong2Grid.xsl
GridID	
Northing	
Easting	
UnitID	Number
Time	GMT
Equipment	Equipment
DistanceInKms	DistanceInMiles

Table 4-2 Initial Mapping of Elements in the Army Schema to the Global Schema

b) Translating

When the mapping is complete, the engineer needs to determine which of two types of translations are required. The two types of translations are those that consist of nothing more than an element name change; and those that require a change in the data. Since XSLT facilitates modularity, some of the latter types of translations might already be defined. In our example, we defined translations that converted from grid to lat-long and back, and made the appropriate entries in CT.XML. Figure 4-5 shows the CT.XML entry for Location. Although our stylesheets do not actually convert a grid position to a latitude and longitude position, the intent here is to outline the process of reconciling a schema with the global schema.

Once each element's translation is defined, a pair of stylesheets can be generated that will translate from the particular message format to the global format, and back down, as in Figure 4-7.

Let's look at one of the stylesheets to see how the translations are defined in XSLT and how the process of

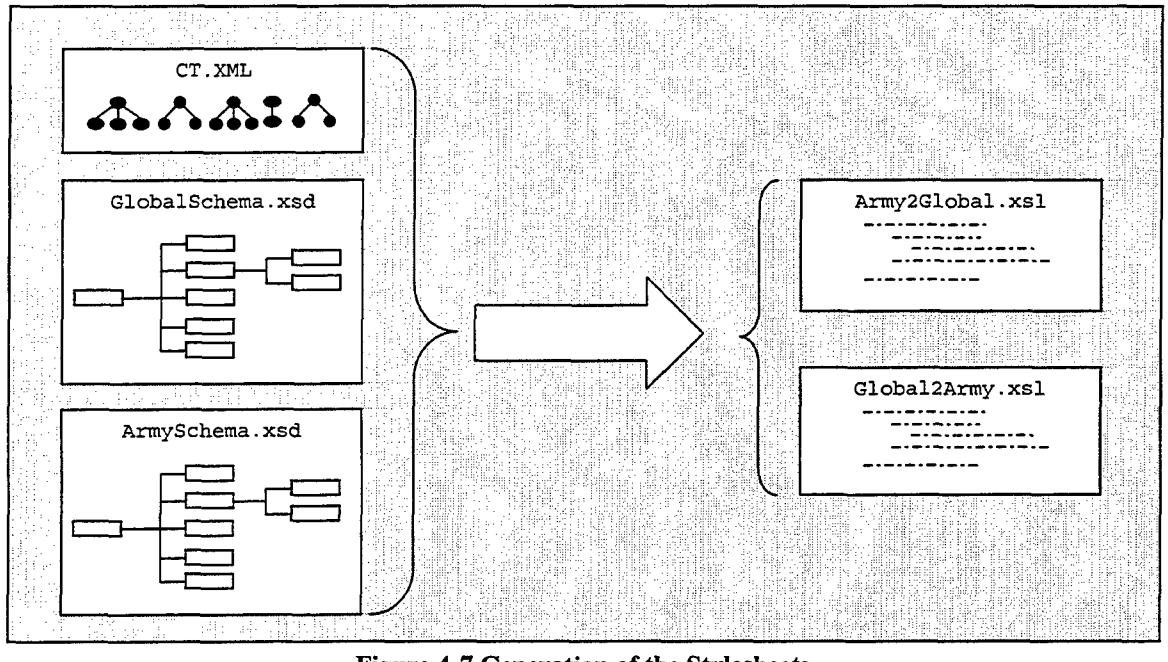

Figure 4-7 Generation of the Stylesheets

generating the stylesheet could be automated once the mapping has been completed. *(This explanation assumes the reader is somewhat familiar with the way that stylesheets work.)*

Our example comes from Appendix G, which is a stylesheet that transforms an Army SALUTE message (Appendix A) into the global CTH format. The first significant

instruction is on line 9. Line 9 tells the processor to look for an element named SOURCE in the XML document to be translated. We used SOURCE as a root node that would be common to all schemas, or message types. Nested in the SOURCE element is the element named Type, which we also used as an element common to all message formats. They serve as an identifier for the source and message type. Lines 10 through 13 are what the processor will output when a SOURCE element is found by the processor. Line 11 is significant because it specifies the schema that the output XML document must conform to, GlobalSchema.xsd. Given that the SOURCE and Type elements are standard elements in all messages, and given the schema for the output message, an automated stylesheet generator could produce this code in a stylesheet.

Lines 18 through 30 tell the processor how to translate a GroundUnit element. They tell the processor that the equivalent name in the global schema is a track, and they specify the order in which to process the children of the GroundUnit element. It is important for the sub-elements to appear in the output document in the correct order so that the document conforms to the global schema. Notice that the order of the output elements is specified in terms of the source schema element names, except lines 24 through 26. Those lines correspond to elements in the

global schema that have no equivalent element in the Salute schema.

A program could automatically generate this XSL code as well. The name correspondences between the schemas' elements are contained in CT.XML. The order in which the sub-elements should be processed is specified in the output schema, in this case the global schema.

Recall that earlier we said there are two basic types of translations. One type of translation merely involves a name change, and the other translation involves a change in the data. Most of the translations contained in Army2Global.xsl are of the former type. However, the translation from MGRS coordinates to latitude/longitude coordinates does require a change in the data. Line 5 is an import instruction to the processor. When the processor sees line 5, it effectively reads the stylesheet Grid2LatLong.xsl and pastes it in place of the import statement. Again, the information required for this line is contained in CT.XML. Incidentally, we chose to use the import statement to demonstrate modularity of stylesheets; however, we could have just done the copy-paste operation ourselves, or a program that generates the Army2Global stylesheet could do it.

We used JavaScript to perform the conversion from miles to kilometers, but we were unable to use the import functionality of XSL because of it. We'll discuss those

efforts later in this chapter. For the present discussion our aim has been to show the content of Army2Global.xsl, and that it could be generated automatically.

2. During Run-Time

Sending a message from System A to System B involves two translations. The first translation will transform the message from System A's format to the global format, the upward translation. The second translation will convert from global to System B's format, the downward translation. Both translations could be performed on either side of the transmission, as long as they're done in the proper order. That is, both could be done by the sender's translator, both by the receiver's translator, or one on each side.

There are two basic problems with doing both the upward and downward translations at the source. First, the source translator would have to know who all the recipients are, along with the appropriate translation for each. It would perform the upward translation and then it would have to perform downward translations for every different type of recipient, and send out multiple versions of the same data. The second potential problem is that changes in a consumer's schema might require the use of a new stylesheet that performs the new downward translations. Now we have to

worry about how to disseminate the new stylesheet to every source that produces information for the modified consumer.

The problem with performing both upward and downward translations at the consumer is essentially the same as the second issue, above. We must have a method of disseminating changes in a producer's upward translations to each of its consumers. Furthermore, both methods would involve some kind of lookup table that would be used at run-time in order to identify the appropriate stylesheet to apply to an outgoing or incoming document.

It is much simpler however, to perform the upward translation at the source and the downward translation at the receiver. This implementation eliminates the complications posed by the other two. Only one version of the document has to be sent. No lookup tables are required because producers always apply the same upward translations to their outgoing messages, and consumers always apply the same downward translations to incoming messages. Also, changes to producer and consumer schemas are localized. Figure 4-8 is a collaboration diagram showing how the system would work.

The CTH will not solve every problem by itself. Translations will still have to be written for many conversions between consolidated types and data contained in specific message formats. What the CTH will do for us is vastly reduce the number of translations that must be

defined, and in some cases enables reuse of those translations. It may also provide a framework for semi-automated generation of the translations.

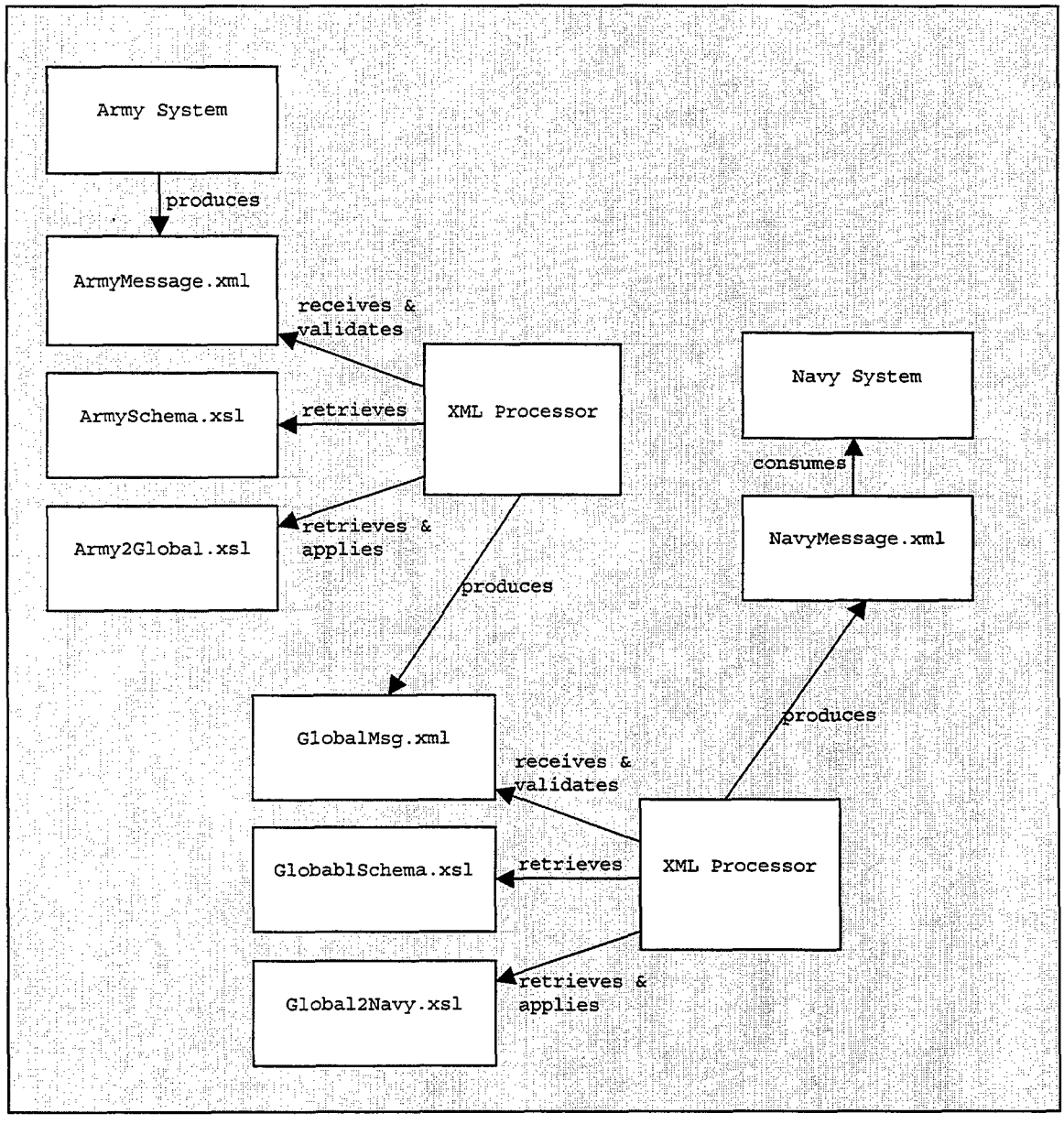

Figure 4-8 Collaboration Diagram of Proposed Implementation

D. RESULTS

We tested our system using a series of steps, incrementally checking what's been advertised about XML against what we were able to achieve. We started by creating two XML documents, one to represent a fictitious Army message format Appendix A, and the other, Navy, Appendix B. We created schemas for them, Appendices C and D, respectively. Next, we created a global schema that incorporated elements from both message formats, Figure 4-3 and Appendix E. Then we created CT.XML, Appendix F, to show the relationships between the elements of the global schema and its constituent schemas.

After entering correspondences between the message formats within CT.XML, we created the stylesheets to translate from the Army SALUTE message directly into the Navy Track Message. The main goal at this step was to verify the performance of an XSL processor. To execute the translations, we used a freeware program named Xalan constructed by IBM Apache Group (http://xml.apache.org/). Xalan is an XSL processor written in a variety of languages for different operating systems. The program takes command line parameters to specify the input and output XML documents, and which stylesheet to apply. The program and the stylesheet worked, and we also found that the resulting

message conformed to TrackSchema, which is the schema defined for the Navy Track Message.

Our next step was to create four stylesheets that performed the upward and downward translations for both Army and Navy message formats. We wanted to test the ability to translate from Army to Navy via the global schema, and perform the reverse. We also wanted to test the modularity of the stylesheets; so, we created two more just to handle the translation of positions, going from MGRS format to latitude-longitude format.

However, translating from MGRS to latitude-longitude requires the use of capabilities the W3C implementation does not support. Functional code is required in order to perform calculations on the data contained by an XML document. The Microsoft implementation of XSL supports JavaScript and Visual Basic Script (VBScript) functions that provide this capability. It uses the xsl:eval statement to invoke script functions from those two languages, but it does not support the import or include instructions as outlined in the W3C XSL namespace. [MSDN2] We implemented some of the final stylesheets (Appendices I and Q) using the xsl:eval processing instruction to demonstrate that XSL is capable of invoking a functional transformation for a user's specific needs, such as converting miles to kilometers.We converted the miles element into kilometers using JavaScript's math library. The stylesheet invokes the

commands using xsl:eval, which then searches for the language, specified in the second line of the stylesheet, as in Appendix I. Since this is an ability that Microsoft implemented for their own XSL processor, MSXSL [MS], the Xalan processor does not process the xsl:eval command. Table 4-3 is a listing of all the files we used, and their purpose.

Appendix	File Name	Description
A	ArmyMessage.xml	Message generated by an Army system. Valid in accordance with SALUTEschema.xsd
B	NavyMessage.xml	Message generated by a Navy system. Valid in accordance with TrackSchema.xsd
C	SALUTEschema.xsd	XML schema for validating messages generated by an Army system.
D	TrackSchema.xsd	XML schema for validating messages generated by a Navy system.
E	GlobalSchema.xsd	Contains the global view of data to be shared. Puts consolidate types in context. Also used for validating messages translated into the global schema.
F	CT.xml	Contains the relationships between the elements of the global schema and the elements of the Army & Navy schemas. (Not used at run-time).
G	Army2Global.xsl	Translates an Army message into a global message.
H	Navy2Global.xsl	Translates a Navy message into a global message.
I	Global2Army.xsl	Translates a global message into an Army message.
J	Global2Navy.xsl	Translates a global message into a Navy message.
K	Grid2LatLong.xsl	A stylesheet module.
L	LatLong2Grid.xsl	A stylesheet module.
M	NewGlobal.xml	An Army XML document that has been translated into a global XML document.
N	NewNavy.xml	An Army XML document that has been translated to a global, and then to a Navy XML document.
O	NewGlobal2.xml	A Navy XML document that has been translated into a global XML document.
P	NewArmy.xml	A Navy XML document that has been translated to a global, and then to a Navy XML document.
Q	Army2Global.xsl	Translates an Army Message into a Global message using Javascript commands

Table 4-3 Listing of files used in example

V. CONCLUSIONS

The purpose of our research was to find a means of communication between legacy systems, preferably using XML. While we were successful in the very limited demonstration of our consolidated type hierarchy, more work must be done to prove its applicability in C4ISR systems. This research was a first step, and should be followed by incorporating more functional transformations into the stylesheets, and then the application of the CTH to a set of real message formats.

The biggest advantage offered by the CTH is the reduction in the number of translations that must be defined. This advantage is realized by using a global, or bridge format for the various message types. Another significant benefit from the CTH model is the opportunity to automate part of the process of defining the translations. Automation could play a role at different stages in the generation of the stylesheets.

First, it is possible to create tool support for identifying elements in the new schema that correlate to an element in the global schema. [SC99] proposes a method for reconciling databases through semantic and structural matching. Since XML is a meta-language and is extensible, descriptive element names can be used, which lends itself to some level of syntactic matching between schemas. Since XML

also captures the structure of the data, structures can also be compared between schemas in order to find potential matches. Such a tool would identify possible matches in a graphical display, allow the engineer to confirm, override, or manually identify matches; and then make the appropriate entries in the global schema and CT.XML.

Another tool that would make the CTH easier to use is automated generation of the stylesheets. Once a message format has been mapped to the global schema, and the translations for individual elements have been identified in CT.XML, then the program should be able to automatically generate the stylesheets that translate entire messages to and from the global schema. All of the necessary information would be contained in the three documents of CT.XML, the global schema, and the system schema.

Another potential area for future work is to create a tool that would search a library of stylesheets in order to facilitate reuse of those transformations.

The best method of implementing the CTH may be in a publish/subscribe architecture. As the different systems log into the networked battlefield, the system would request to receive messages of a certain type. As each individual legacy system sends data over the network, a wrapper would intercept the message. The wrapper would mark up the message into a CTH XML representation, then send it to a web server. The web server would check the list of valid

subscribers for that message format, and send the message to those destinations. The destination system's XML wrapper would translate from the CTH mark-up form into the correct legacy system format.

By reutilizing the legacy systems similar to the mega-programming concept, we hope to save DoD thousands of dollars from cost savings and cost avoidance. Growing a Consolidated Type Hierarchy from our model will enable a variety of systems to communicate information across the battlefield regardless of branch or nationality.

The CTH is a powerful model that will allow more than just message systems to exchange information. It could be used for object-oriented databases, as well as source code files and initially any other kind of data. An application of this nature would allow more reuse of previously developed code and reduce development time and costs. An issue that remains to be investigated is the degree of overhead relative to real-time constraints and optimization methods for mitigating time and space overhead.

LIST OF REFERENCES

[DTIC] Defense Technical Information Center, Department of Defense Dictionary. http://www.dtic.mil/doctrine/jel/doddict/data/i/03090.html

[GW92] Wiederhold, G., Wegner, P., and Ceri, S. Toward Megaprogramming. Communications of the ACM (Nov. 1992).

[HMS] Hammer, J.; McLeod, D; Si, A.; Object Discovery and Unification in Federated Database Systems.

[JWJO00] Wing, J., and Ockerbloom, J. Respectful Type Converters. IEEE Transactions on Software Engineering (July 2000).

[KM98] Kahng, Jonghyun; McLeod, D. Dynamic Classificational Ontologies for Discovery in Cooperative Federated Databases. Cooperative Information Systems, 1996. Proceedings, First IFCIS International Conference on , 1996. Pages: 26 -35.

[MS] http://msdn.microsoft.com/downloads/default.asp?URL=/code/sample.asp?url=/MSDN-FILES/027/001/485/msdncompositedoc.xml

[MSDN] http://msdn.microsoft.com/library/default.asp?URL=/library/psdk/xmlsdk/xmlp7k6d.htm.

[MSDN2] http://msdn.microsoft.com/xml/XSLGuide/conformance.asp

[PROXML] Anderson, Richard; Birbeck, Mark; Kay, Michael; Livingstone, Steven; Loesgen, Brian; Martin, Didier; Mohr, Stephen; Ozu, Nicola; Peat, Bruce; Pinnock, Jonathon; Stark, Peter; Williams, Kevin. Professional XML. Wrox Press, August, 2000. p. 242.

[RKM] Roantree, M.; Keane, J.; Murphy, J. A Three-layer Model for Schema Management in Federated Databases. System Sciences, 1997, Proceedings of the Thirtieth Hwaii International Conference on , Volume: 1 , 1997 Pages: 44 -53 vol.1.

[SC99] S. Castano and V. De Antonellis. "A Schema Analysis and Reconciliation Tool Environment for Heteregeneous Databases", IEEE Databases, Feb 1999, pp. 53-62.

[Sin98] Narinder Singh. Unifying heterogeneous information models. Communications of the ACM 41, 5 (May. 1998), Pages 37 - 44.

[W3C] http://www.w3.org/TR/#About ©1995-2000, W3C.

BIBLIOGRAPHY

[AD99] van Deursen, T. Kuipers. "Identifying Objects using Cluster and Concept Analysis", Proceedings, 21st International Converference on Software Engineering, ICSE-99, ACM, 1999.

[CY94] C. Yu and W. Meng. "Progress in Database Search Strategies", IEEE Software, Oct 1994, pp. 1-19.

[EL99] E. Lee. "Embedded Software-An Agenda for Research", University of California Berkeley, Dec 1999. Available as UCB/ERL No. M99/63. (http://ptolemy.eecs.berkeley.edu/publications/papers/99/embedded/)

[DF91] D. Fang, J. Hammer, and D. McLeod. "The Identification and Resolution of Semantic Heterogeneity in Multidatabase Systems", IEEE Transactions on Software Engineering, Mar 1991, pp. 136-143.

[DH00] D. Hina. "Evaluation of the Extensible Mark-up Language as a Means for Establishing Interoperability between Heterogeneous Department of Defense (DoD) Databases." Naval Postgraduate School, Sep. 2000.

[HQDA] Headquarters, Dept. of the Army. ***Information Operations,* Field Manual 100-6,** Washington, DC: USGPO, 27 August 1996, p. iv.

[JBC] Joint Battle Center. "Joint Battle Management Integration(JBMI) Use Cases", Joint Battle Management Integration Assessment-Phase 2, 21 Aug 2000. *(internal document)* [MH99] Hiemstra, Michael A., Colonel, U.S. Army. Center for Army Lessons Learned, *NEWSLETTER NO. 99-2: "IO in a Peace Enforcement Environment"*, Fort Leavenworth, KS: USGPO, 19 June 1999.

[JH94a] J. Hammer, D. McLeod, and A. Si. "An Intelligent System for Identifying and Integrating Non-Local Objects in Federated Database Systems", University of Southern California. Available as a technical report on the internet. (ftp://ftp.usc.edu/pub/csinfo/tech-reports/papers/94-575.ps.Z)

[JH94b] J. Hammer, D. McLeod, and A. Si. "Object Discovery and Unification in Federated Database Systems", University of Southern California. Available as a technical report on the internet. (ftp://ftp.usc.edu/pub/csinfo/tech-reports/papers/94-574.ps.Z)

[L88] Luqi, V. Berzins, and R. Yeh. "A Prototyping Language for Real-Time Software", Software Engineering, IEEE Transactions on , Volume: 14 Issue: 10 , Oct. 1988, pp. 1409 -1423.

[TA00] T. Tran and J. Allen. "Interoperability and Security Support for Heterogeneous COTS/GOTS/Legacy Component Based Architecture. Naval Postgraduate School, Sep. 2000.

[VL99] V. Berzins, Luqi, B. Schultes, J. Guo, J. Allen, N Cheng, K. Gee, T. Nguyen, E. Stierna. "Interoperability Technology Assessment for Joint C4ISR Systems". Naval Postgraduate School, Sep. 2000.

APPENDIX A-ARMYMESSAGE.XML

This is the source file for the Army SALUTE message in XML. This was an input to the translator along with stylesheet ``Army2Global.xsl'', and was transformed into a global message, "NewGlobal.xml".

```
<!-- edited with XML Spy v3.5 NT beta 2 build Dec  1 2000 (http://www.xmlspy.com) by Brian Lyttle (Home) -->
<!--This file captures the representation of an Army SALUTE Report.  It is used when soldiers find an enemy on the battlefield, and
report the enemy's activity.  The Army constructed the report before automation, but today it still contains the same information.
The information is structured like this:
    S:  Size of the enemy unit, ie people, vehicles.
    A:  Activity of the enemy, ie walking, emplacing, sleeping.
    L:  Location in Military Grid Reference Position, with Grid identifier, Northing, and Easting.
    U:  Unit identification, to include distinctive symbols, patches, vehicle numbers.
    T:  Time the activity was observed.
    E:  Equipment the enemy possessed during the activity, such as M60 Machine Guns, AK-47s, mortars-->
<SOURCE name="ArmySystem" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=".\newSALUTESchema.xsd">
    <Type MsgID="SALUTE"/>
    <GroundUnit>
        <Size>10</Size>
        <Activity>WalkingNE</Activity>
        <Location>
            <GridID>NK</GridID>
            <Northing>100</Northing>
            <Easting>400</Easting>
        </Location>
        <UnitID>150MRR</UnitID>
        <Time>2159Z</Time>
        <Equipment>AK_47sampAT</Equipment>
        <DistanceInKms>10</DistanceInKms>
    </GroundUnit>
    <GroundUnit>
        <Size>5</Size>
        <Activity>RunningNE</Activity>
        <Location>
            <GridID>NK</GridID>
            <Northing>50</Northing>
            <Easting>350</Easting>
        </Location>
        <UnitID>100MRR</UnitID>
        <Time>2159Z</Time>
        <Equipment>M16</Equipment>
        <DistanceInKms>25</DistanceInKms>
    </GroundUnit>
</SOURCE>
```

APPENDIX B-NAVYMESSAGE.XML

This is the source file for the Navy Track Report message in XML. It shows what a Track Report would look like in XML.

```
<!--The Navy TrackReport possesses a set of tracks that identify objects. The objects are identified by a variety of
sensors such as Airborne radars and shipboard sensors. They communicate
information to each other via Tactical Data Links (TADIL) in a near real time fashion. The computers on-board the
sea and air platforms receive the infomration via the TADIL link, and use them in the information system as part of
a display for the operator. The display contains a picture of all nearby objects detected by the sensors. Our
representation is a simplified version used for our puposes to demonstrate the abilities of the CTH.
The entries for track are:
    Number: the number given to the object by the TADIL system.
    Coordinates: the latitude/longitude position of the object.
    Course: the direction (in degrees) of the object
    Speed: how fast the object is traveling in miles per hour
    Status: tells if the object is friendly, enemy, or unknown.
    IFF: the Identification Friend or Foe code that is received from the beacon on the object.
    GMT: time of the last sighting of this object, in Greenwich Mean Time.-->
<SOURCE name="NavyMessage" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=".\newTrackSchema.xsd">
    <Type MsgID="TrackReport"/>
    <Track>
        <Number>1000</Number>
        <Coordinates>
            <Latitude>32-36N</Latitude>
            <Longitude>30-20W</Longitude>
        </Coordinates>
        <Course>0</Course>
        <Speed>14</Speed>
        <Status>Unknown</Status>
        <IFF/>
        <GMT>1502</GMT>
        <DistanceInMiles>100</DistanceInMiles>
    </Track>
    <Track>
        <Number>1111</Number>
        <Coordinates>
            <Latitude>32-35N</Latitude>
            <Longitude>30-21W</Longitude>
        </Coordinates>
        <Course>0</Course>
        <Speed>14</Speed>
        <Status>Unknown</Status>
        <IFF/>
        <GMT>1503</GMT>
        <DistanceInMiles>10</DistanceInMiles>
    </Track>
</SOURCE>
```

APPENDIX C-SALUTESCHEMA.XSD

This is the XML Schema for the Army SALUTE Report, "SaluteSchema.xsd". It defines the structure of the "ArmyMessage.xml" document. This is the code represented by Figure 4-1.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT beta 2 build Dec  1 2000 (http://www.xmlspy.com) by Brian Lyttle (Home) -->
<!--W3C Schema generated by XML Spy v3.5 NT beta 2 build Dec  1 2000 (http://www.xmlspy.com)-->
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema" elementFormDefault="qualified">
    <xsd:element name="SOURCE">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Type">
                    <xsd:complexType>
                        <xsd:attribute name="MsgID" type="xsd:string" use="required"/>
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="GroundUnit" maxOccurs="unbounded">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="Size" type="xsd:byte"/>
                            <xsd:element name="Activity" type="xsd:string"/>
                            <xsd:element name="Location">
                                <xsd:complexType>
                                    <xsd:sequence>
                                        <xsd:element name="GridID" type="xsd:string"/>
                                        <xsd:element name="Northing" type="xsd:string"/>
                                        <xsd:element name="Easting" type="xsd:string"/>
                                    </xsd:sequence>
                                </xsd:complexType>
                            </xsd:element>
                            <xsd:element name="UnitID" type="xsd:string"/>
                            <xsd:element name="Time" type="xsd:string"/>
                            <xsd:element name="Equipment" type="xsd:string"/>
                            <xsd:element name="DistanceInKms" type="xsd:float"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

APPENDIX D-TRACKSCHEMA.XSD

This is the XML Schema for the Navy Track Report, "TrackSchema.xsd". It defines the structure of "NavyMessage.xml". This is the code represented by Figure 4-1.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT beta 2 build Dec  1 2000 (http://www.xmlspy.com) by Brian Lyttle (Home) -->
<!--W3C Schema generated by XML Spy v3.5 NT beta 2 build Dec  1 2000 (http://www.xmlspy.com)-->
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema" elementFormDefault="qualified">
    <xsd:element name="SOURCE">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Type">
                    <xsd:complexType>
                        <xsd:attribute name="MsgID" type="xsd:string" use="required"/>
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="Track" maxOccurs="unbounded">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="Number" type="xsd:string"/>
                            <xsd:element name="Coordinates">
                                <xsd:complexType>
                                    <xsd:sequence>
                                        <xsd:element name="Latitude" type="xsd:string"/>
                                        <xsd:element name="Longitude" type="xsd:string"/>
                                    </xsd:sequence>
                                </xsd:complexType>
                            </xsd:element>
                            <xsd:element name="Course" type="xsd:string"/>
                            <xsd:element name="Speed" type="xsd:string"/>
                            <xsd:element name="Status" type="xsd:string"/>
                            <xsd:element name="IFF" type="xsd:string"/>
                            <xsd:element name="GMT" type="xsd:string"/>
                            <xsd:element name="DistanceInMiles" type="xsd:string"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

APPENDIX E-GLOBALSCHEMA.XSD

This is the code from "GlobalSchema.xsd". It is represented by Figure 4-3. The global schema defines the structure of a global message, as in "NewGlobal.xml" and "NewGlobal2.xml".

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT beta 2 build Dec  1 2000 (http://www.xmlspy.com) by Brian Lyttle (Home) -->
<!--W3C Schema generated by XML Spy v3.5 NT beta 2 build Dec  1 2000 (http://www.xmlspy.com)-->
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema" elementFormDefault="qualified">
    <xsd:element name="SOURCE">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Type">
                    <xsd:complexType>
                        <xsd:attribute name="MsgID" type="xsd:string" use="required"/>
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="Track" maxOccurs="unbounded">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="Number" type="xsd:string"/>
                            <xsd:element name="GMT" type="xsd:string"/>
                            <xsd:element name="Location">
                                <xsd:complexType>
                                    <xsd:sequence>
                                        <xsd:element name="Latitude" type="xsd:string"/>
                                        <xsd:element name="Longitude" type="xsd:string"/>
                                    </xsd:sequence>
                                </xsd:complexType>
                            </xsd:element>
                            <xsd:element name="Status" type="xsd:string"/>
                            <xsd:element name="Course" type="xsd:string"/>
                            <xsd:element name="Speed" type="xsd:string"/>
                            <xsd:element name="IFF">
                                <xsd:complexType/>
                            </xsd:element>
                            <xsd:element name="Size" type="xsd:string"/>
                            <xsd:element name="Equipment" type="xsd:string"/>
                            <xsd:element name="DistanceInMiles" type="xsd:string"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

APPENDIX F-CT.XML

This file contains the relationships between the consolidated types found in the global schema and the elements found in the Army and Navy schemas. This is a concrete example of Figure 4-4.

```
<?xml version="1.0" encoding="UTF-8"?>
<ConsolidatedTypes xmlns="www.nps.navy.mil/sw/CTH/Global">
    <Track>
        <TrackReport name="Track" upXlate="Navy2Global.xsl" dnXlate="Global2Navy.xsl"/>
        <Salute name="GroundUnit" upXlate="Army2Global.xsl" dnXlate="Global2Army.xsl"/>
    </Track>
    <Number>
        <TrackReport name="Number"/>
        <Salute name="UnitID" upXlate="UnitID2Track.xsl" dnXlate="Track2UnitID.xsl"/>
    </Number>
    <Location>
        <TrackReport name="Location"/>
        <Salute name="Location" upXlate="Grid2LatLong.xsl" dnXlate="LatLong2Grid.xsl"/>
    </Location>
    <Course>
        <TrackReport name="Course"/>
    </Course>
    <Speed>
        <TrackReport name="Speed"/>
    </Speed>
    <Status>
        <TrackReport name="Status"/>
        <Salute name="Activity"/>
    </Status>
    <IFF>
        <TrackReport name="IFF"/>
    </IFF>
    <GMT>
        <TrackReport name="GMT"/>
        <Salute name="Time"/>
    </GMT>
    <Size>
        <Salute name="Size"/>
    </Size>
    <Equipment>
        <Salute name="Equipment"/>
    </Equipment>
    <Latitude>
        <TrackReport name="Latitude"/>
    </Latitude>
    <Longitude>
        <TrackReport name="Longitude"/>
    </Longitude>
    <DistanceInMiles>
        <TrackReport name="DistanceInMiles"/>
        <SALUTE name="DistanceInKms"/>
    </DistanceInMiles>
</ConsolidatedTypes>
```

APPENDIX G-ARMY2GLOBAL.XSL

This XSLT stylesheet transforms an Army SALUTE report into a global message. When we applied this stylesheet to "ArmyMessage.xml" the message produced was "NewGlobal.xml". Line numbers have been added to facilitate referral in the text.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet version="1.0" xmlns:xsl=http://www.w3.org/1999/XSL/Transform
3  xmlns:fo="http://www.w3.org/1999/XSL/Format">
4  <!--Stylesheet to translate from Army SALUTE Report to a CTH message-->

5  <xsl:import href=".\Grid2LatLong.xsl"/>

6  <xsl:template match = "/">
7      <xsl:apply-templates/>
8  </xsl:template>

9  <xsl:template match="SOURCE">
10     <SOURCE name="GlobalMessage" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
11                 xsi:noNamespaceSchemaLocation=".\GlobalSchema.xsd" >
12         <xsl:apply-templates/>
13     </SOURCE>
14 </xsl:template>

15 <xsl:template match="Type">
16     <Type MsgID="TrackReport"/>
17 </xsl:template>

18 <xsl:template match="GroundUnit">
19     <Track>
20         <xsl:apply-templates select="UnitID"/>
21         <xsl:apply-templates select="Time"/>
22         <xsl:apply-templates select="Location"/>
23         <xsl:apply-templates select="Activity"/>
24         <Course/>
25         <Speed/>
26         <IFF/>
27         <xsl:apply-templates select="Size"/>
28         <xsl:apply-templates select="Equipment"/>
29     </Track>
30 </xsl:template>

31 <xsl:template match="UnitID">
32     <Number>
33         <xsl:value-of select="."/>
34     </Number>
35 </xsl:template>

36 <xsl:template match="Time">
37     <GMT>
38         <xsl:value-of select="."/>
39     </GMT>
40 </xsl:template>
```

```
41 <xsl:template match="Location">
42     <Location>
43     <xsl:apply-templates/>
44     </Location>
45 </xsl:template>

46 <xsl:template match="Activity">
47     <Status>
48         <xsl:value-of select="."/>
49     </Status>
50 </xsl:template>

51 <xsl:template match="Size">
52     <Size>
53         <xsl:value-of select="."/>
54     </Size>
55 </xsl:template>

56 <xsl:template match="Equipment">
57     <Equipment>
58         <xsl:value-of select="."/>
59     </Equipment>
60 </xsl:template>

61 </xsl:stylesheet>
```

APPENDIX H-NAVY2GLOBAL.XSL

This XSLT stylesheet transforms a Navy Track report into a global message. When we applied this stylesheet to "NavyMessage.xml" the message produced was "NewGlobal2.xml".

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <!--Stylesheet to translate from a Navy Track Report to a CTH message-->

    <xsl:template match = "/">
        <xsl:apply-templates/>
    </xsl:template>

    <xsl:template match="SOURCE">
        <SOURCE name="GlobalMessage" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation=".\newGlobalSchema.xsd" >
            <xsl:apply-templates/>
        </SOURCE>
    </xsl:template>

    <xsl:template match="Type">
        <Type MsgID="TrackReport"/>
    </xsl:template>

    <xsl:template match="Track">
        <Track>
            <xsl:apply-templates select="Number"/>
            <xsl:apply-templates select="GMT"/>
            <xsl:apply-templates select="Coordinates"/>
            <xsl:apply-templates select="Status"/>
            <xsl:apply-templates select="Course"/>
            <xsl:apply-templates select="Speed"/>
            <xsl:apply-templates select="IFF"/>
            <Size/>
            <Equipment/>
            <xsl:apply-templates select="DistanceInMiles"/>
        </Track>
    </xsl:template>

    <xsl:template match="Number">
        <Number>
            <xsl:value-of select="."/>
        </Number>
    </xsl:template>

    <xsl:template match="Coordinates">
        <Location>
            <xsl:apply-templates/>
        </Location>
    </xsl:template>

    <xsl:template match="Latitude">
        <Latitude>
            <xsl:apply-templates/>
```

```
        </Latitude>
    </xsl:template>

    <xsl:template match="Longitude">
        <Longitude>
            <xsl:apply-templates/>
        </Longitude>
    </xsl:template>

    <xsl:template match="Course">
        <Course>
            <xsl:value-of select="."/>
        </Course>
    </xsl:template>

    <xsl:template match="Speed">
        <Speed>
            <xsl:value-of select="."/>
        </Speed>
    </xsl:template>

    <xsl:template match="Status">
        <Status>
            <xsl:value-of select="."/>
        </Status>
    </xsl:template>

    <xsl:template match="IFF">
        <IFF>
            <xsl:value-of select="."/>
        </IFF>
    </xsl:template>

    <xsl:template match="GMT">
        <GMT>
            <xsl:value-of select="."/>
        </GMT>
    </xsl:template>

    <xsl:template match="DistanceInMiles">
        <DistanceInMiles>
            <xsl:value-of select="."/>
        </DistanceInMiles>
    </xsl:template>


</xsl:stylesheet>
```

APPENDIX I-GLOBAL2ARMY.XSL

This XSLT stylesheet transforms a global message into an Army SALUTE report. When we applied this stylesheet to "NewGlobal.xml" the message produced was "NewArmy.xml".

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/TR/WD-xsl" language="JavaScript">
    <!--Stylesheet to translate from a CTH message to an Army SALUTE Report-->

<!-- <xsl:import href=".\LatLong2Grid.xsl"/>-->

    <xsl:template match = "/">
        <xsl:apply-templates/>
    </xsl:template>

    <xsl:template match="SOURCE">
        <SOURCE name="ArmySystem" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
                    xsi:noNamespaceSchemaLocation=".\newSALUTEschema.xsd" >
            <xsl:apply-templates/>
        </SOURCE>
    </xsl:template>

    <xsl:template match="Type">
        <Type MsgID="SALUTE"/>
    </xsl:template>

    <xsl:template match="Track">
        <GroundUnit>
            <xsl:apply-templates select="Size"/>
            <xsl:apply-templates select="Status"/>
            <xsl:apply-templates select="Location"/>
            <xsl:apply-templates select="Number"/>
            <xsl:apply-templates select="GMT"/>
            <xsl:apply-templates select="Equipment"/>
            <xsl:apply-templates select="DistanceInMiles"/>
        </GroundUnit>
    </xsl:template>

    <xsl:template match="Number">
        <UnitID>
            <xsl:value-of select="."/>
        </UnitID>
    </xsl:template>

    <xsl:template match="GMT">
        <Time>
            <xsl:value-of select="."/>
        </Time>
    </xsl:template>

    <xsl:template match="Location">
        <Location>
            <xsl:apply-templates/>
        </Location>
    </xsl:template>
```

```
    <xsl:template match="Latitude">
        <GridID>
        </GridID>
        <Northing>
            <xsl:value-of select="."/>
        </Northing>
    </xsl:template>

    <xsl:template match="Longitude">
        <Easting>
            <xsl:value-of select="."/>
        </Easting>
    </xsl:template>

    <xsl:template match="Status">
        <Activity>
            <xsl:value-of select="."/>
        </Activity>
    </xsl:template>

    <xsl:template match="Size">
        <Size>
            <xsl:value-of select="."/>
        </Size>
    </xsl:template>

    <xsl:template match="Equipment">
        <Equipment>
            <xsl:value-of select="."/>
        </Equipment>
    </xsl:template>

    <xsl:template match="DistanceInMiles">
        <DistanceInKms><xsl:eval>this.nodeTypedValue*(2.21)</xsl:eval></DistanceInKms>
    </xsl:template>

</xsl:stylesheet>
```

APPENDIX J-GLOBAL2NAVY.XSL

This XSLT stylesheet transforms a global message into a Navy Track report. When we applied this stylesheet to "NewGlobal2.xml" the message produced was "NewNavy.xml".

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <!--Stylesheet to translate from a CTH message to a Navy Track Report-->

    <xsl:template match = "/">
        <xsl:apply-templates/>
    </xsl:template>

    <xsl:template match="SOURCE">
        <SOURCE name="NavyMessage"
            xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation=".\newTrackSchema.xsd" >
            <xsl:apply-templates/>
        </SOURCE>
    </xsl:template>

    <xsl:template match="Type">
        <Type MsgID="TrackReport"/>
    </xsl:template>

    <xsl:template match="Track">
        <Track>
            <xsl:apply-templates select="Number"/>
            <xsl:apply-templates select="Location"/>
            <xsl:apply-templates select="Course"/>
            <xsl:apply-templates select="Speed"/>
            <xsl:apply-templates select="Status"/>
            <xsl:apply-templates select="IFF"/>
            <xsl:apply-templates select="GMT"/>
            <xsl:apply-templates select="DistanceInMiles"/>
        </Track>
    </xsl:template>

    <xsl:template match="Number">
        <Number>
            <xsl:value-of select="."/>
        </Number>
    </xsl:template>

    <xsl:template match="Location">
        <Coordinates>
            <xsl:apply-templates/>
        </Coordinates>
    </xsl:template>

    <xsl:template match="Latitude">
        <Latitude>
            <xsl:apply-templates/>
        </Latitude>
```

```
    </xsl:template>

    <xsl:template match="Longitude">
        <Longitude>
            <xsl:apply-templates/>
        </Longitude>
    </xsl:template>

    <xsl:template match="Course">
        <Course>
            <xsl:value-of select="."/>
        </Course>
    </xsl:template>

    <xsl:template match="Speed">
        <Speed>
            <xsl:value-of select="."/>
        </Speed>
    </xsl:template>

    <xsl:template match="Status">
        <Status>
            <xsl:value-of select="."/>
        </Status>
    </xsl:template>

    <xsl:template match="IFF">
        <IFF>
            <xsl:value-of select="."/>
        </IFF>
    </xsl:template>

    <xsl:template match="GMT">
        <GMT>
            <xsl:value-of select="."/>
        </GMT>
    </xsl:template>

    <xsl:template match="DistanceInMiles">
        <DistanceInMiles>
            <xsl:value-of select="."/>
        </DistanceInMiles>
    </xsl:template>


</xsl:stylesheet>
```

APPENDIX K-GRID2LATLONG.XSL

This XSLT stylesheet is imported by "Army2Global.xsl". This stylesheet does not actually convert a grid position into a latitude-longitude position. We used this stylesheet to test and demonstrate the modularity of XSLT stylesheets.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">


        <xsl:template match="GridID">
        </xsl:template>

        <xsl:template match="Northing">
            <Latitude>
                <xsl:value-of select="."/>
            </Latitude>
        </xsl:template>

        <xsl:template match="Easting">
            <Longitude>
                <xsl:value-of select="."/>
            </Longitude>
        </xsl:template>

</xsl:stylesheet>
```

APPENDIX L-LATLONG2GRID.XSL

This XSLT stylesheet is imported by "Global2Army.xsl". This stylesheet does not actually convert a latitude-longitude position into a grid position. We used this stylesheet to test and demonstrate the modularity of XSLT stylesheets.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">

    <xsl:template match="Latitude">
        <GridID>
        </GridID>
        <Northing>
            <xsl:value-of select="."/>
        </Northing>
    </xsl:template>

    <xsl:template match="Longitude">
        <Easting>
            <xsl:value-of select="."/>
        </Easting>
    </xsl:template>

</xsl:stylesheet>
```

APPENDIX M-NEWGLOBAL.XML

This is the output of the XSL processor when "Army2Global.xsl" is applied to "ArmyMessage.xml"

```
<SOURCE name="GlobalMessage" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=".\newGlobalSchema.xsd">
    <Type MsgID="TrackReport"/>
    <Track>
        <Number>
            150MRR
        </Number>
        <GMT>
            2159Z
        </GMT>
        <Location>
            <Latitude>
                100
            </Latitude>
            <Longitude>
                400
            </Longitude>
        </Location>
        <Status>
            WalkingNE
        </Status>
        <Course/>
        <Speed/>
        <IFF/>
        <Size>
            10
        </Size>
        <Equipment>
            AK_47sampAT
        </Equipment>
        <DistanceInMiles>
            4.52488687782805
        </DistanceInMiles>
    </Track>
    <Track>
        <Number>
            100MRR
        </Number>
        <GMT>
            2159Z
        </GMT>
        <Location>
            <Latitude>
                50
            </Latitude>
            <Longitude>
                350
            </Longitude>
        </Location>
        <Status>
            RunningNE
        </Status>
        <Course/>
```

```
        <Speed/>
        <IFF/>
        <Size>
            5
        </Size>
        <Equipment>
            M16
        </Equipment>
        <DistanceInMiles>
            11.3122171945701
        </DistanceInMiles>
    </Track>
</SOURCE>
```

APPENDIX N-NEWNAVY.XML

This is the output of the XSL processor when "Global2Navy.xsl" is applied to "NewGlobal.xml"

```
<SOURCE name="NavyMessage" xsi:noNamespaceSchemaLocation=".\newTrackSchema.xsd"
xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance">
    <Type MsgID="TrackReport"/>
    <Track>
        <Number>150MRR </Number>
        <Coordinates>
            <Latitude>100</Latitude>
            <Longitude>400</Longitude>
        </Coordinates>
        <Course/>
        <Speed/>
        <Status>WalkingNE</Status>
        <IFF/>
        <GMT>2159Z</GMT>
        <DistanceInMiles>4.52488687782805</DistanceInMiles>
    </Track>
    <Track>
        <Number>100MRR </Number>
        <Coordinates>
            <Latitude>50</Latitude>
            <Longitude>350</Longitude>
        </Coordinates>
        <Course/>
        <Speed/>
        <Status>RunningNE</Status>
        <IFF/>
        <GMT> 2159Z</GMT>
        <DistanceInMiles>11.3122171945701</DistanceInMiles>
    </Track>
</SOURCE>
```

APPENDIX O-NEWGLOBAL2.XML

This is the output of the XSL processor when "Navy2Global.xsl" is applied to "NavyMessage.xml"

```
<?xml version="1.0" encoding="UTF-16"?>
<SOURCE name="GlobalMessage" xsi:noNamespaceSchemaLocation=".\newGlobalSchema.xsd"
xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance">
    <Type MsgID="TrackReport"/>
    <Track>
        <Number>1000</Number>
        <GMT>1502</GMT>
        <Location>
            <Latitude>32-36N</Latitude>
            <Longitude>30-20W</Longitude>
        </Location>
        <Status>Unknown</Status>
        <Course>0</Course>
        <Speed>14</Speed>
        <IFF/>
        <Size/>
        <Equipment/>
        <DistanceInMiles>100</DistanceInMiles>
    </Track>
    <Track>
        <Number>1111</Number>
        <GMT>1503</GMT>
        <Location>
            <Latitude>32-35N</Latitude>
            <Longitude>30-21W</Longitude>
        </Location>
        <Status>Unknown</Status>
        <Course>0</Course>
        <Speed>14</Speed>
        <IFF/>
        <Size/>
        <Equipment/>
        <DistanceInMiles>10</DistanceInMiles>
    </Track>
</SOURCE>
```

APPENDIX P-NEWARMY.XML

This is the output of the XSL processor when "Global2Navy.xsl" is applied to "NewGlobal.xml"

```
<SOURCE name="ArmySystem" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=".\newSALUTEschema.xsd">
    <Type MsgID="SALUTE"/>
    <GroundUnit>
        <Size/>
        <Activity>
            Unknown
        </Activity>
        <Location>
            <GridID/>
            <Northing>
            32-36N
        </Northing>
            <Easting>
            30-20W
        </Easting>
        </Location>
        <UnitID>
            1000
        </UnitID>
        <Time>
            1502
        </Time>
        <Equipment/>
        <DistanceInKms>221</DistanceInKms>
    </GroundUnit>
    <GroundUnit>
        <Size/>
        <Activity>
            Unknown
        </Activity>
        <Location>
            <GridID/>
            <Northing>
            32-35N
        </Northing>
            <Easting>
            30-21W
        </Easting>
        </Location>
        <UnitID>
            1111
        </UnitID>
        <Time>
            1503
        </Time>
        <Equipment/>
        <DistanceInKms>22.1</DistanceInKms>
    </GroundUnit>
</SOURCE>
```

APPENDIX Q-ARMY2GLOBAL.XSL USING ''XSL:EVAL''

This is file differs from Appendix G because it uses the "xsl:eval" command and does not use the import ability implemented in the w3c version of XSL. However, it does convert from kilometers to miles and still transforms MGRS to lat/long coordinates.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/TR/WD-xsl" language="JavaScript">
    <!--Stylesheet to translate from Army SALUTE Report to a CTH message-->
    <!-- <xsl:include href=".\Grid2LatLong.xsl"/>-->
    <!--The include statement is an accepted statement in a different XSL namespace called
xmlns:xsl="http://www.w3.org/1999/XSL/Transform".
However, in the namespace used by this stylesheet, "include" and "import" are not accepted commands. Since we
wanted to demonstrate the
ability of XML to functionally transform objects, we selected the above namespace. The "XSL/Transform"
namespace is used to transform the
trees formed by the two documents, while the "TR/WD-xsl" namespace is used to format objects for a destination
system.

The w3c is reviewing different recommendations, and we hope the two namespaces are combined :).-->
    <xsl:template match="/">
        <xsl:apply-templates/>
    </xsl:template>
    <xsl:template match="SOURCE">
        <SOURCE name="GlobalMessage"
            xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
                    xsi:noNamespaceSchemaLocation=".\newGlobalSchema.xsd" >
            <xsl:apply-templates/>
        </SOURCE>
    </xsl:template>
    <xsl:template match="Type">
        <Type MsgID="TrackReport"/>
    </xsl:template>
    <xsl:template match="GroundUnit">
        <Track>
            <xsl:apply-templates select="UnitID"/>
            <xsl:apply-templates select="Time"/>
            <xsl:apply-templates select="Location"/>
            <xsl:apply-templates select="Activity"/>
            <Course/>
            <Speed/>
            <IFF/>
            <xsl:apply-templates select="Size"/>
            <xsl:apply-templates select="Equipment"/>
            <xsl:apply-templates select="DistanceInKms"/>
        </Track>
    </xsl:template>
    <xsl:template match="UnitID">
        <Number>
            <xsl:value-of select="."/>
        </Number>
    </xsl:template>
    <xsl:template match="Time">
        <GMT>
            <xsl:value-of select="."/>
        </GMT>
    </xsl:template>
```

```
    <xsl:template match="Location">
        <Location>
            <xsl:apply-templates/>
        </Location>
    </xsl:template>
    <xsl:template match="Activity">
        <Status>
            <xsl:value-of select="."/>
        </Status>
    </xsl:template>
    <xsl:template match="Size">
        <Size>
            <xsl:value-of select="."/>
        </Size>
    </xsl:template>
    <xsl:template match="Equipment">
        <Equipment>
            <xsl:value-of select="."/>
        </Equipment>
    </xsl:template>
    <xsl:template match="GridID"/>
    <xsl:template match="Northing">
        <Latitude>
            <xsl:value-of select="."/>
        </Latitude>
    </xsl:template>
    <xsl:template match="Easting">
        <Longitude>
            <xsl:value-of select="."/>
        </Longitude>
    </xsl:template>
    <xsl:template match="DistanceInKms">
        <DistanceInMiles>
            <xsl:eval>this.nodeTypedValue/(2.21)</xsl:eval>
        </DistanceInMiles>
    </xsl:template>
</xsl:stylesheet>
```

ABSTRACT

There is a need for Commercial-off-the-shelf (COTS), Government-off-the-shelf (GOTS) and legacy components to interoperate in a secure distributed computing environment in order to facilitate the development of evolving applications.

This thesis researches existing open standards solutions to the distributed component integration problem and proposes an application framework that supports application wrappers and a uniform security policy external to the components. This application framework adopts an Object Request Broker (ORB) standard based on Microsoft Distributed Component Object Model (DCOM). Application wrapper architectures are used to make components conform to the ORB standard. The application framework is shown to operate in a common network architecture.

A portion of the Naval Integrated Tactical Environmental System I (NITES I) is used as a case study to demonstrate the utility of this distributed component integration methodology (DCIM).

I. INTRODUCTION

There is a need for Commercial-off-the-shelf (COTS), Government-off-the-shelf (GOTS) and legacy components to inter-operate in a secure distributed computing environment in order to facilitate the development of evolving applications.

This thesis researches existing open standards solutions to the distributed component integration problem and proposes an application framework that supports application wrappers and a uniform security policy external to the components. This application framework adopts an Object Request Broker (ORB) standard based on Microsoft Distributed Component Object Model (DCOM). Application wrapper architectures are used to make components conform to the ORB standard. The application framework is shown to operate in a common network architecture.

A portion of the Naval Integrated Tactical Environmental System I (NITES I) is used as a case study to demonstrate the utility of this distributed component integration methodology (DCIM). The System Requirement Specification (SRS), System Design Specification (SDS) and Visual Basic Implementation, found in the appendices, are the results of a collaborative effort with graduate students Karen Gee and Thomas Nguyen.

Unified Modeling Language (UML) methodology is used in the formal specification of the system.

The Joint C4ISR Battle Center (JBC) Study considered several approaches to solving the interoperability problem, including wrappers,

messaging, data mediators, data replicators, data translators, and ORBs, and evaluated each approach using the following criteria: performance, reliability, speed to field, cost, extendibility, COTS support, security and standards. The empirical scores for each criterion of each approach are plotted on a Kiviat graph. The JBC Study, published at the Naval Post Graduate School in 1999, recommends a solution in the context of ORBs, but with caveats. Re-evaluation is needed, as new products are available. Background and training of personnel is an important consideration in selecting a solution. [Ref. 1] This thesis also recommends the ORB approach and focuses on Microsoft Distributed Component Object Model (DCOM) with emphasis on setting security policy external to the component. Legacy applications are made DCOM compliant by wrapping the application within a DCOM component. Custom applications wrappers need to be designed, which is consistent with the findings of the JBC study.

This thesis is organized into the following chapters:

- Chapter II researches existing solutions to the distributed component integration problem.
- Chapter III proposes a methodology that can be used to transform desktop legacy applications into distributed web based applications.
- Chapter IV presents a design pattern application framework encompassing security and wrappers that is applied to the case study.

- Chapter V discusses the portion of the NITES system used as case study to validate the usefulness of the proposed methodology.
- Chapter VI presents the lessons learned and conclusions from the case study.

II. EXISTING SOLUTIONS TO THE INTEROPERABILITY PROBLEM

A. GENERIC SECURITY SERVICE APPLICATION PROGRAM INTERFACE (GSS-API)

GSS-API is emerging as an Internet standard for securing applications. GSS-API is embedded in Common Object Request Broker Architecture (CORBA), Kerberos, Distributed Computing Environment/Remote Procedure Call (DCE/RPC), Sequence Packet Exchange (SPX), KryptoKnight, and SOCKS [Ref. 2]. GSS-API is popular because it is an interface specification that is independent of implementation mechanism, independent of placement, and independent of communication protocol. The interface specification is a product of the IETF Common Authentication Technology Working Group. Version 2 of GSS-API has 37 function calls broken down into 4 categories: Credential Management, context-level, per-message and support.

GSS-API assumes the application establishes a connection to a service, messages are transferred to and from the service, and the service will not request another external service on behalf of the user.[Ref. 2]

B. KERBEROS

Kerberos was developed in the 1980's at MIT to provide additional security for the Athena system. The primary goals were to provide single logon to a network of application servers and protect authentication from masquerading attacks. Kerberos is an implementation mechanism for GSS-API. Kerberos assumes the client, network and server cannot be trusted and that a third party key distribution center (KDC)

is needed to store secret keys. The KDC is composed of two logical entities, the authentication server (AS) and the ticket-granting server (TGS). The AS is responsible for authenticating the user and providing the user a ticket to access the TGS. The user sends its identity, server and nonce. A nonce is a randomly generated one-time value that is used to counter a replay attack. The AS responds with a session key, server and nonce encrypted using the user's secret key and a ticket encrypted with the server's secret key. The TGS is responsible for granting the user a ticket to access the requested server for a limited period of time. The user sends to the server an authenticator encrypted with the session key and the ticket obtained from TGS. The server decrypts the ticket to obtain the session key which in turn is used to decrypt the authenticator. Typically the authenticator has a timestamp that must be within 5 minutes of the current time. To provide mutual authentication the server returns the authenticator encrypted with the session key. Strong authentication is achieved because secret keys were never passed in the clear. [Ref. 3]

Kerberos has several weaknesses. The user's secret key is stored in the host's memory during AS exchange. Kerberos is vulnerable to password guessing attacks. Registering each service with the KDC does not scale. Applications must be modified to take advantage of Kerberos.

C. A SECURE EUROPEAN SYSTEM FOR APPLICATIONS IN A MULTI-VENDOR ENVIRONMENT (SESAME)

Sesame is the European substitute for Kerberos. Sesame implements all the specified security services. Sesame architecture can be divided into 4 major entities: client, security server, application server and support components. GSS-API calls need to be added to the client and application server entities in places where messages are being sent and received. The C source code for Sesame V4 for Redhat Linux V5 is available at www.cosic.east.kuleuven.ac.be/sesame. There is a project underway to convert Sesame to Java in order to improve portability. [Ref. 2]

D. DISTRIBUTED COMPUTING ENVIRONMENT (DCE)

The Open Systems Foundation (OSF) specification for DCE includes facilities for security, directory services, time services, threads and remote procedure calls.

DCE 1.2 is compatible with Kerberos V5 so single logon and mutual authentication services are available. DCE uses Access Control Lists (ACLs) for authorization. Role based authorization is not available. Like Kerberos, DCE/RPC uses a session key to provide secure communication services between the client and server. A rich set of APIs, including GSS-API is available to the programmer. These APIs provide data confidentiality and integrity services. [Ref. 2]

The DCE web site is www.camb.opengroup.org/tech/dce.

E. KRYPTOKNIGHT

KryptoKnight has been under development at IBM since 1992. Kerberos influenced the design of this system. Similar security services include single logon per user, mutual authentication, key distribution and data integrity and confidentiality. Role based authorization is not provided. The 2-party, 3-party and inter-domain protocols are designed to minimize network usage and computer processing.[Ref. 2]

The KryptoKnight web page is www.zurich.ibm.com/~sti/g-kk/extern/kryptoknight

F. WINDOWS NT SECURITY MODEL

The goal of any multitasking and networked operating system security is to ensure that system resources such as memory, files, devices and CPUs cannot be accessed without authorization.

The NT security model has three major components: the logon process, the security reference monitor, and other security subsystems.

1. Local User Logon Process

Each user has an account on a local machine that is managed by administrators using the Security Accounts Manager (SAM). In a NT server environment, each user may also have a domain account. The Primary Domain Controller (PDC) and the Backup Domain Controller (BDC) are responsible for authenticating the user. Once authenticated, the user has access to any machine on the network that allows access to domain users. The trusted domain relationship is one-way and not transitive.

Each user may be assigned to one or more groups. If the number of users exceeds the number of groups, assigning users to groups and privileges and permissions to groups reduces the administrator's task of managing security policy.

2. Security Reference Monitor

The reference monitor is responsible for authorizing access to any NT object and audit generation. The reference monitor accesses all NT objects consistently and uniformly. User mode processes pass an object handle to system services operating in kernel mode.

There are 23 NT object types: adapter, controller, desktop, device, directory, driver, event, eventPair, file, IOCompletion, key, mutant, port, process, profile, section, semaphore, symbolicLink, thread, timer, token, type, and windowStation. Each object type has a set of attributes that are common to all object types and a set of attributes specific to the object type. The object manager uses the common attributes to provide the following services: close, duplicate, query object, query security, set security, wait for single object, wait for multiple objects.

Each NT object has a security descriptor attribute which defines the permissions, auditing and ownership of an object. The corresponding structures are named Discretionary Access Control List (DACL), System Access Control List (SACL), and Owner Security Ids (OwnerSID). Each entry in the list is named an

Access Control Entry (ACE). The owner controls a DACL ACE. The security administrator controls a SACL ACE. An ACE can contain a collection of access rights that may be generic, standard or specific. Generic access rights are read, write, execute and all (read, write, execute). Generic access rights can be mapped to standard access rights that are delete access, read access to security descriptor, read, write, execute, synchronize, write DAC, write Owner, required, and all.

In summary a user access token includes a Security ID (SID), a list of privileges and a list of group SIDs. An object security descriptor includes an owner SID, DACL, and SACL. [Ref. 4]

3. Audit Security Subsystem

The following table describes the types of events that can be audited in Windows NT. [Ref. 5]

Type of event	Description
Logon and Logoff	A user logged on or off or made a network connection.
File and Object Access	A user opened a directory or a file that is set for auditing in File Manager, or a user sent a print job to a printer that is set for auditing in Print Manager.
Use of User Rights	A user used a user right (except those rights related to logon and logoff).
User and Group Management	A user account or group was created, changed, or deleted. A user account was renamed, disabled, or enabled; or a password was set or changed.
Security Policy Changes	A change was made to the User Rights, Audit, or Trust Relationships policies.

Restart, Shutdown, and System	A user restarted or shut down the computer, or an event has occurred that affects system security or the security log.
Process Tracking	These events provided detailed tracking information for things like program activation, some forms of handle duplication, indirect object accesses, and process exit.

Table 1.1 Windows NT Event Types for Audit

The Event Viewer utility formats and displays audit event records.

Audit event records include header information that is present in all event records. The following list describes this common information.

- The time the event was generated.
- The SID of the subject that caused the event to be generated. If possible, Event Viewer translates this SID to an account name for display. The SID is the impersonation ID if the subject is impersonating a client, or the primary ID if the subject is not impersonating.
- The name of the system component or module that submitted the event. For security audits this is always Security.
- The module-specific ID of the specific event.
- The event type, either Success Audit or Failure Audit.
- The event category, used to group related events such as logon audits, object access audits, and policy change audits.[Ref. 5]

G. DCOM

Figure 1.1 shows the overall DCOM architecture. The client uses an interface, represented by a lollipop, to access a service provided by a remote component. Using DCE RPC and common security providers makes DCOM available on other platforms including Apple Macintosh, Sun Solaris, Linux, AIX, and MVS.

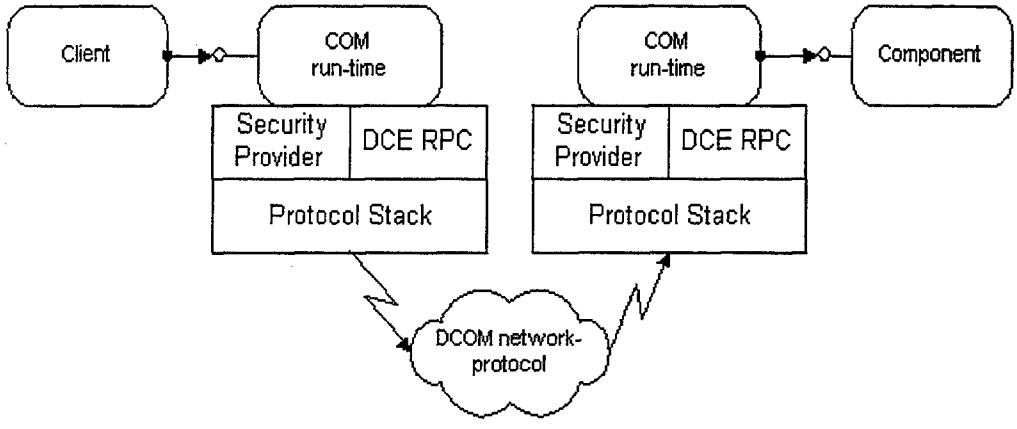

Figure 1.1. Overall DCOM Architecture [Ref. 5]

DCOM can provide security services for COTS components externally by using the DCOM configuration tool or by embedding security API calls within components. The primary DCOM security services fall into three categories: access, launch and call. Access security checks for privilege to connect to a running object. Launch security checks for privilege to create an object. Call security checks for privilege to access a component interface.

Each client has a security context that encapsulates security services. Security features, such as mutual authentication, can be selected just by setting a property value.

DCOM can impersonate the client on a server machine to allow nested client-server architecture. Impersonation can also be used to control access to individual properties and methods of components.

DCOM is layered on Object Remote Procedure Call (ORPC) which is an extension of DCE RPC. These services are accessible through the WIN32 Security Support Provider Interface (SSPI). DCOM can also accommodate multiple third party security providers.

DCOM uses Windows NT NTLM, Kerberos V5 or Distributed Password Authentication (DPA) authentication protocols.

DCOM uses SSL/PCT protocols to provide integrity and confidentiality services for communication connections.

DCOM uses the Windows Registry and the ACL facilities of the Windows NT operating system. DCOM is also available on Macintosh and UNIX platforms.[Ref. 4]

H. JAVA

Java 1.1 applets run in a virtual machine on a host machine. The assumption is that all applets are un-trusted unless accompanied by a digital signature. The virtual machine protects the host from un-trusted applets utilizing the "sandbox" approach. This means the capabilities of Java applications that are potentially harmful to the host are restricted in applets. For example, an applet may not access the host file system.

The java.lang.SecurityManager class implements the applet security restrictions. A security policy is created by instantiating and registering a security manager object. A potentially harmful operation causes an exception that is handled by a security manager method.

I. CORBA

The Common Object Services specification (CORBASec) describes security related tasks and requirements needed for CORBA.

A CORBA ORB, ORBacus, from Object Oriented Concept Inc. has been used to implement some specified security services. ORBacus currently provides the Security Level 1 functionality of CORBASec. Security Level 1 provides security services for applications that are unaware of security including mutual authentication, confidentiality and integrity.

The messages exchanged are encapsulated in the Secure Inter-ORB Protocol (SECIOP) message format. SECIOP provides a standard for maintaining security and interoperability between ORBs. Each end maintains its state following the rules of the SECIOP Context Management finite state machine.

The security functionality underneath is that of Kerberos V5 and is accessed through a Java binding of the GSS-API.

J. SECURE SOCKETS LAYER (SSL)

SSL is positioned between the TCP/IP application and connections layers enabling multiple services such as Telnet, HTTP and FTP to

establish secure connections without modification to the services. SSL utilizes RSA Public/Private key architecture. The server identity is validated to the client by x.509 digital certificates. Optionally the client identity can also be validated to the server. The server has access to an LDAP compliant key directory server.[Ref. 6]

K. SECURE HYPERTEXT TRANSFER PROTOCOL (S-HTTP)

S-HTTP permits parties to negotiate symmetric or asymmetric keys, key management technique, message formats, and cryptographic strength. S-HTTP allows for multiple trust models to be negotiated between client and server. Security features are specific to the HTTP protocol.[Ref. 3]

L. IP SECURITY (IPSEC)

IPSec provides for secure transfer of IP packets across an untrusted network. IPSec resides at the network layer of the OSI model. IPSec is transparent to protocols at higher layers in the OSI model. IPSec is an open standard for encryption on an IP network.

Two one-way security associations (SA) between hosts or gateways store security parameters (Source IP, cryptographic algorithm, cryptographic keys, user or gateway name, data sensitivity level, transport layer protocol, source and destination ports). Unique SA key includes security parameter index (SPI), IP destination, and security protocol, either Association Header (AH) or Encapsulated Security Payload (ESP). With ESP, the enclosed packet(tunneling) is encrypted,

so original source and destination addresses could be unregistered. [Ref. 7]

III. GENERIC WRAPPER FOR SYSTEM COMPONENTS

A. REQUIREMENTS OF THE GENERIC WRAPPER FOR SYSTEM COMPONENTS

1. General Description

The security services designed for commercial applications often focus on data integrity while military applications focus on data confidentiality. In order for COTS components to operate in a military environment, the commercial security services must be carefully selected to achieve military security requirements. The next section contains a list of security services applicable to the military environment that are also available in various combinations within commercial products. A methodology shall be developed to transform classes of legacy modules into reusable components using the wrapper architecture.

Components shall pass messages transparently across language, operating systems and network boundaries.

A common set of security services across operating systems will simplify implementation of a security policy.

The following security services shall be available to the customer:

- Single logon for users
- Mutual authentication
- Auditing
- Key distribution
- Role based Access Control

- Data confidentiality
- Data integrity
- Data availability
- Non-repudiation

The single logon for users means the user needs to identify him once per session. It is the responsibility of the security services to protect and distributed the authentication information of a user.

Mutual authentication ensures proper identification of the user to the system and the system to the user.

Auditing means significant security events are recorded for later analysis. Significant security events shall include login, logout, password change, and access validation.

Key distribution provides a secure transport mechanism for encryption keys.

Role based access control assigns roles to users and privileges to roles, thereby simplifying access control if the number of roles is less than the number of users.

Data confidentiality means data is disclosed according to a policy.

Data integrity means the recipient gets the intended data.

Data availability means the user has access to the data when needed.

Non-repudiation means the sender of a message cannot later deny he sent the message.

2. Environment

The classes of projects targeted by this thesis typically operate in an environment with the following conditions:

- Components pass messages synchronously or asynchronously.
- Components may have real-time constraints.
- A hierarchy of interacting COTS, GOTS and custom components may be assembled to form an application.
- Implementation will be dependent on the security services of the host operating systems.
- Security policies need to evolve and policy implementations need to be manageable in a distributed computing environment.
- Some components may be in binary executable form where compile or link is not possible. Other components may be re-linked but not recompiled. Other components may not be re-linked but substitution of dynamic load libraries (DLL) is possible. Other components may be modified at the source code level and recompiled.
- The security services will not be exported outside of the United States.
- Attacks can come from inside or outside an organization.
- This security system must be adaptable to counter new kinds of security attacks.

- The target systems will operate at a single level of security at no higher than the discretionary access control level (C2).

B. SPECIFICATION OF THE GENERIC WRAPPER FOR SYSTEM COMPONENTS

Wrappers that need to exchange self-describing content over a network can use XML. Utilization of XML within wrappers makes data transport mechanism independent of language or operating system. Following is a description of the XML standard.

1. XML Standard

XML is an emerging standard for transferring data among distributed components in web applications. Industry has been quick to agree on XML vocabularies. NITES has developed a nationally recognized vocabulary for meteorological data. See Appendix E for XML meteorological vocabulary and sources for other vocabularies.

XML offers the following desirable features:

- XML describes data that can be specified in a lexical tree structure. Unlike directed graphs, trees can be efficiently traversed.
- XML and HTML share the same level in the WEB architecture. Both can use the secure HTML mechanism and the digital signature mechanism.
- XML specification is the product of the World Wide Web Consortium (W3C) and is recognized as a standard for distribution of data over the Internet.

- All content is encoded in the specified Unicode character set. There is no need to wrap vendor specific data formats.
- Industry specific XML vocabularies make content available to any compliant application.
- XML vocabularies are extensible without affecting earlier versions.

Any DoD joint application should consider evolving to XML. Some common steps to gradually incorporate XML into an existing project include:

- Categorize the types on information the system handles. Examples are personnel, weather, tactical, and logistics.
- Search for existing XML standards in categories.
- If there are no XML standards within a category, organize a standards committee, and produce an industry wide standard.
- Develop components to transform existing messages, records, etc. into XML entities. A one-time transformation is usually preferable to repeated run-time transformations.
- Use existing tools to provide additional transformations such as record set to XML.
- Use security zones of the browser to implement security policy. Use XML parser imbedded in browser to extract information for presentation.

a) Security

The security zone features have been extended in Internet Explorer 5 (IE5) to provide security services for the embedded XML parser. The zones include local, Internet, local intranet, trusted site, and restricted site in order of trustworthiness. The originating zone may access a zone that is equal or less trustworthy. [Ref. 5]

b) Namespaces

XML namespace specification developed by World Wide Web Consortium (W3C) is implemented on IE5. This allows developers to define unique element names using a registered qualifier.

c) Document Type Definitions (DTDs)

DTDs utilize XML to describe rules to validate an XML document. DTDs are an optional section of the XML document.

d) Document Object Model (DOM)

The DOM provides a standard way to programmatically construct and traverse any XML document. The XML document is composed of objects with attributes and methods. DOM can be applied to the task of transforming an ActiveX Data Object (ADO) record set into an XML document. Interfaces are defined for the DOM and all XML objects.

e) XML Specification

The XML specification is on the Web at URL www.w3.org/xml. Production rules are in the Extended Backus-Naur Format (EBNF). An annotated version is at Web site www.xml.com/xml/pub/axml/axmlintro.html.

The design goals for XML are:

- XML shall be straightforwardly usable over the Internet.
- XML shall support a wide variety of applications.
- XML shall be compatible with SGML.
- It shall be easy to write programs which process XML documents.
- The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
- XML documents should be human-legible and reasonably clear.
- The XML design should be prepared quickly.
- The design of XML shall be formal and concise.
- XML documents shall be easy to create.
- Terseness in XML markup is of minimal importance. [Ref. 8]

2. COTS Application exposes API

DCOM and CORBA use an Interface Definition Language (IDL) to name and describe an interface containing public attributes, methods and events. There is a many-to-many relationship between interfaces and components. A component may implement one or more interfaces. The interface serves as a contract between the component developer and user.

How do you ensure each interface has a unique name when many independent activities are creating interfaces? One solution is to use a routine that will always generate a different name each

time it is called. DCOM uses this solution to generate unique class and interface names. Once an interface has been assigned a name it will never change. There is no way to modify an interface and use its original name. This guarantees that all legacy code will never need to be changed because an interface has been modified.

DCOM interfaces are language and platform independent. For example, a component written in Visual Basic and running on a Windows NT platform can use a component written in C++ and running on a Unix platform.

DCOM and CORBA require each component to implement the Unknown interface. From this interface, all interfaces implemented by the component can be dynamically discovered.

Dynamic discovery and use of an interface is known as late binding. Use of a priori knowledge of implemented interfaces is known as early binding. DCOM and CORBA both support early and late binding. There is a performance penalty for using late binding.

Microsoft Visual Basic hides many interface details. The development environment generates the IDL from the class implementation. The unique IDL name is automatically generated. The clause "with events" will enable receipt of events. The Unknown interface is automatically generated.

Microsoft Word, Excel and Powerpoint are examples of COTS components that expose an API. In the case study the Powerpoint API is used by the application wrapper.

3. Standard file naming and directory conventions for component determination

On Windows NT there is a many-to-one relationship between a file type and an application. For example, the file type PPT is associated with the PowerPoint application.

NITES imagery applications generate TIF, GIF, and MIF file types. PowerPoint is capable of processing the above file types.

Middleware wrappers can take advantage of standard file naming conventions and directory conventions to integrate components. For example, if a COTS application periodically generates an imagery file to a known directory, middleware can poll the directory for new files with a file type of interest and pass the file to a consumer of the file type.

4. Command line input support for COTS COMPONENTS Invocation

UNIX and DOS have popularized starting an application and passing switches and parameters on a command line. This same mechanism can be used from within a program to start another program. A wrapper can use this mechanism to integrate independent COTS applications.

A chaining model is used when the calling program terminates after execution. An asynchronous model is used when the calling and called programs operate in parallel. A synchronous model is

used when the calling program waits for completion of the called program.

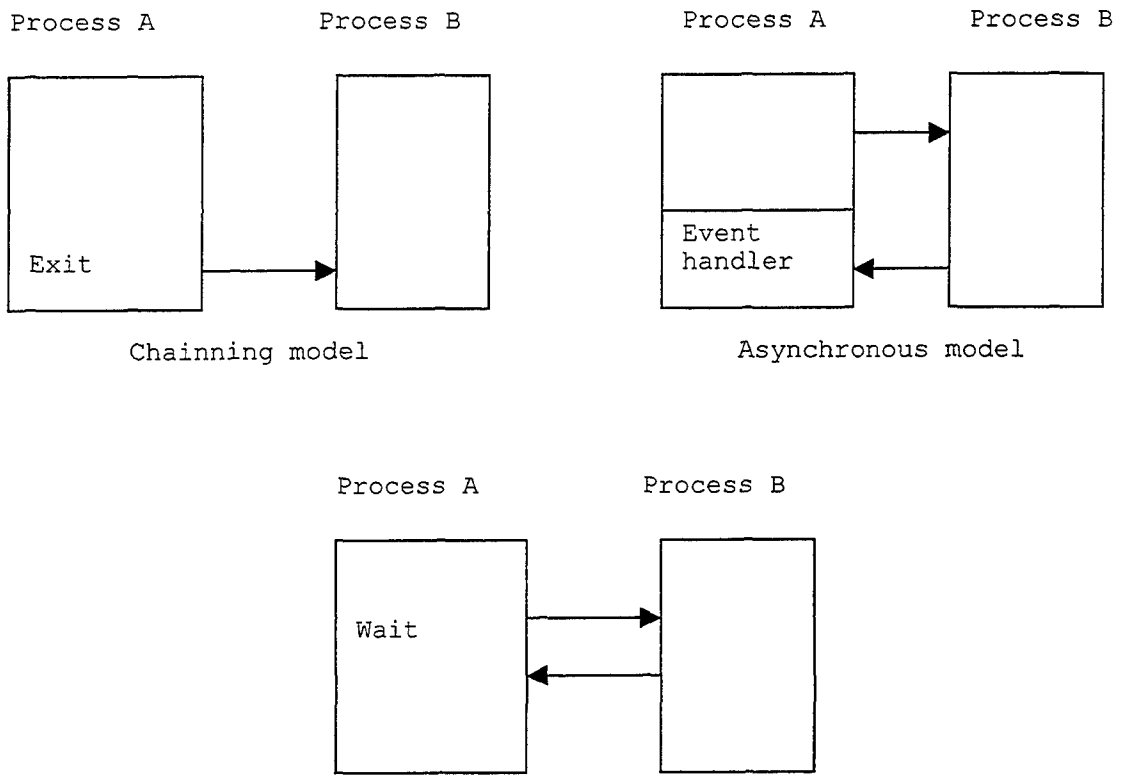

Figure 3.1. Wrapper calling models

IV. ARCHITECTURAL DESIGN PATTERN

A. ARCHITECTURAL DESIGN

The architectural design pattern represented in Figure 4.1 is common to many IT systems including NITES and USCG National Distress Response System Modernization Program (NDRSMP).

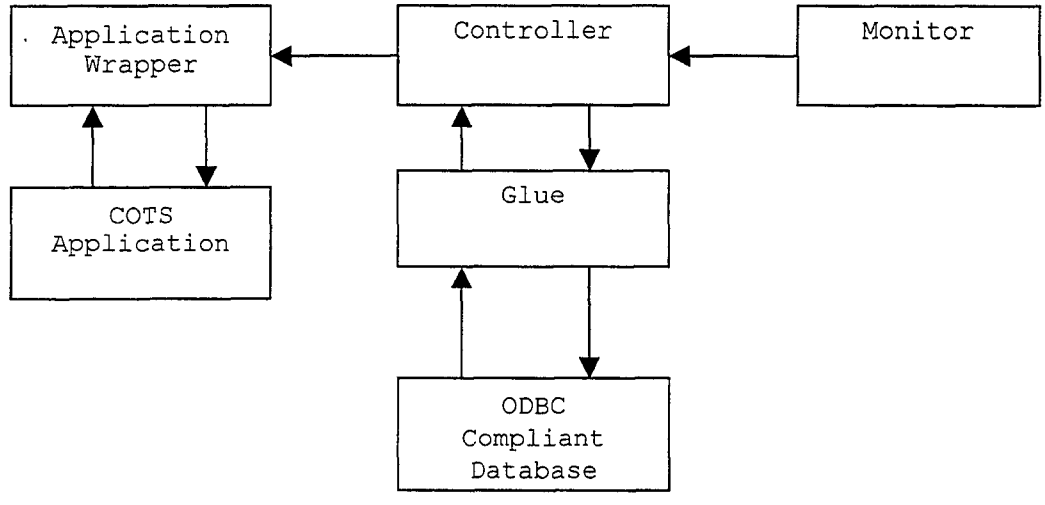

Figure 4.1. Architectural Design Pattern

The realization of this architecture on a network of Windows NT machines running DCOM, IIS, Internet Explorer and optionally a UNIX relational database server machine, satisfies the requirements of the previous section.

In NITES, the object is a TIF file containing a satellite image. In NDRSMP, the object is a WAV file containing a voice segment. The Monitor component is responsible for detecting the presence of a new

object. The controller component is responsible for coordinating multiple concurrent asynchronous activities. The glue component is responsible for storing and retrieving objects from a ODBC compliant relational database. The Application Wrapper is responsible for making the object available to a COTS viewer application.

B. NITES IMPLEMENTATION

1. Using Architectural Design Pattern

A Windows NT DCOM solution in Visual Basic (VB) was used in NITES to implement the architectural design pattern. See Appendix D for the skeleton VB code. The launch, access and permission security features were set external to each component using DCOMCNFG utility. The DCOMCNFG utility was also used to set the location of each component and user account assigned to the component. The automation data types were used to make marshaling and un-marshaling of data transparent to each component. Migration from a desktop application to an Internet Explorer 5 (IE) was performed to reduce maintenance. Client components can be maintained on the server and automatically downloaded to the client. Migration is accomplished by converting the project type from standard executable to an ActiveX control using Microsoft Visual Studio.

The key to generic wrapper design is to use standard objects. Standard objects include widely used file extensions such as Tagged Image File Format (TIFF) and WAV, XML meta data, and

record sets. There are COTS plug-in viewers for each of the above standard object types.

2. Thin Client Technology

The web based application wrapper is implemented using modern thin client technology. When a user opens a HTTP page from a browser, the wrapper is then automatically downloaded and installed on the client machine. Once the wrapper is up and running, all images needed for creating the brief are dynamically downloaded from the server using the OpenURL method. OpenURL uses the current open HTTP connection to transfer image files. The continuous brief is created on the client machine using the PowerPoint APIs. The PowerPoint is used to display the brief.

3. Push Technology

The advantage of using push technology is that the client does not need to poll the server periodically for new data. The server notifies its clients (wrapper) when new data (images) arrive. The wrapper receives the notification and compares the image type with the type being showed. If the image types match, the wrapper downloads a new set of images from the server and updates the brief.

C. NETWORK ARCHITECTURE

Figure 4.2 depicts network architecture similar to many systems including NITES. The network is composed of an intranet divided into four sub-nets, a router connecting the four sub-nets and providing a connection to the internet service provider, and a dial-in access server. Two sub-nets separate the traffic of two user groups. Security and packet wrapper options within this network architecture are characterized. The components in the architectural design pattern are typically deployed on the web server and user computers.

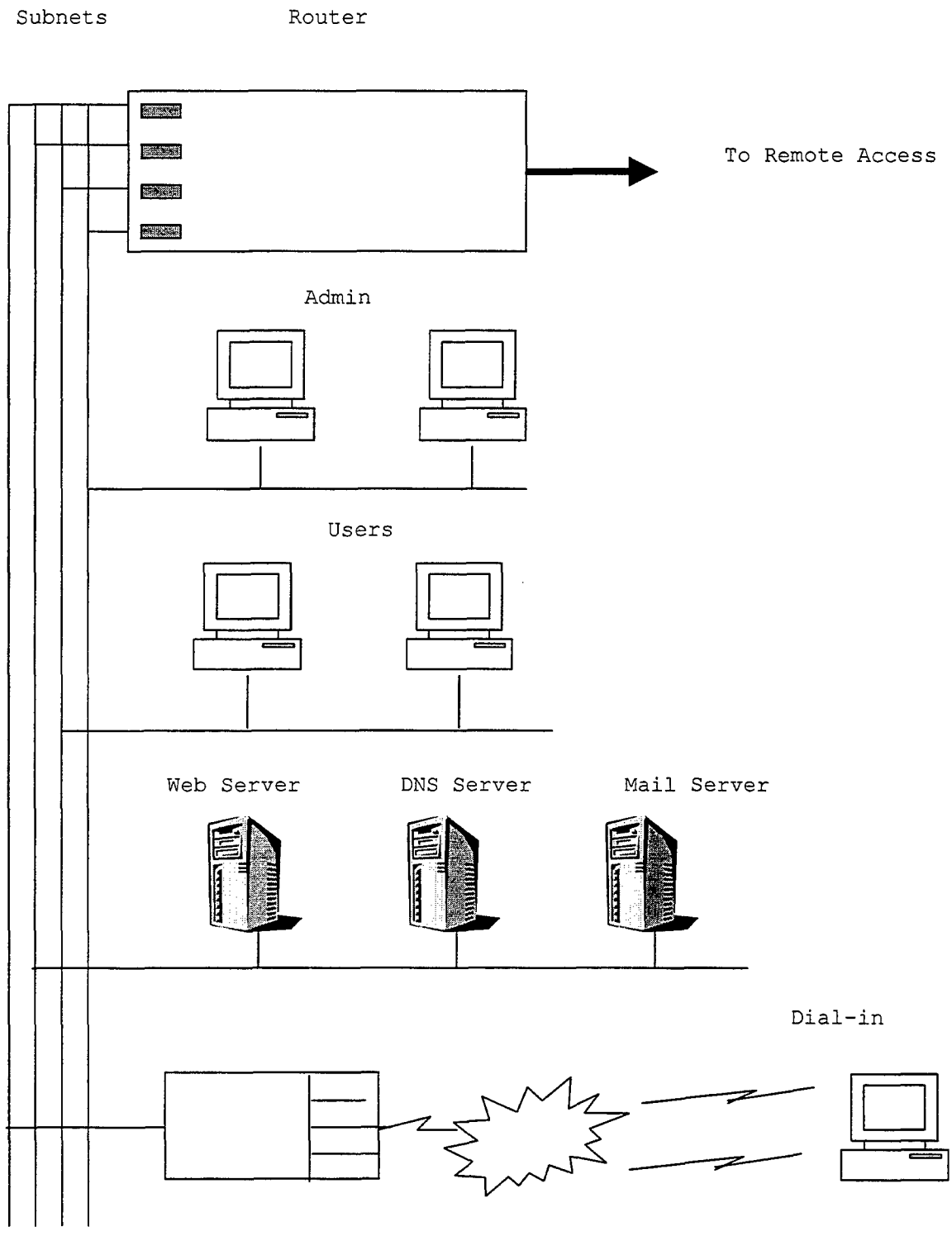

Figure 4.2. Network Architecture

1. Intranet Security

A hierarchical network architecture formed with routers offers traffic isolation and additional security. Using ACLs and IP filters on the router Ethernet interfaces can control traffic flow across subnets. Some routers, including the popular Cisco router, are capable of protecting against IP spoofing.

2. Internet Security

Standard security mechanisms are available at different layers of the OSI Network Model. Point-to-point tunneling protocol (PPTP), Layer 2 tunneling protocol (L2TP), Frame Relay, and Asynchronous transfer mode (ATM) are available at the Data link layer. IP security (IPSec) and Generic routing encapsulation (GRE) are available at the Network layer. SOCKSv5, SSL and TLS are available at the session layer.

3. Dial-in Security

Some authentication schemes, such as password authentication protocol (PAP), transfer passwords in the clear and are vulnerable to snooping. Stronger authentication schemes are available.

The dial-in access server is a convenient place to host authentication schemes for mobile users. Remote Authentication Dial-in User Service (RADIUS) is a draft standard that covers protocols for a centralized access server. RADIUS allows for one-time token authentication schemes.

Windows NT provides Challenge Handshake Authentication Protocol (CHAP). Client and server share a common secret key. A unique session key is negotiated without transferring the secret key in the clear. A unique session key limits the usefulness of replay attacks to the current session.

V. CASE STUDY

A. CASE STUDY OVERVIEW

A subset of the operational NITES system was chosen for the case study. This subset is representative of the issues involved in the integration of COTS software components where only the executables are available.

The case study covers the wrapper and security aspects of component integration.

The wrapper transforms COTS applications into a COM/DCOM component enabling interfaces with infrastructure components as shown in Figure 5.2.

1. App

The App is the COTS application that provides the APIs used by the App Wrapper to integrate with other components.

2. App Wrapper

The App Wrapper is the software code developed to add, modify, and hide functionality from COTS, GOTS or legacy software components to align them with the overall system requirements and architecture. In the design, wrapper and glue code technology is being implemented to enable the COTS applications to adhere to the existing NITES architecture.

3. System Monitor

The Monitor component is responsible for detecting the presence of a new object.

4. System Controller

The controller component is responsible for coordinating multiple concurrent asynchronous activities. The controller runs on the application server. It serves two functions within the system, handling notifications from the monitor and the glue component.

5. Storage Directory

The Storage Directory is a target directory that is accessed by the IMGEDT application and the Glue component. This is the location for the data temporarily stored before being updated to, or retrieved from the database.

6. Application (IMGEDT)

IMGEDT is a COTS application that generates the satellite images.

7. Glue Component

The glue component is responsible for storing and retrieving objects from an ODBC compliant relational database.

8. Database

The Database is an OBDC compliant relational database that is available for storing and retrieving data.

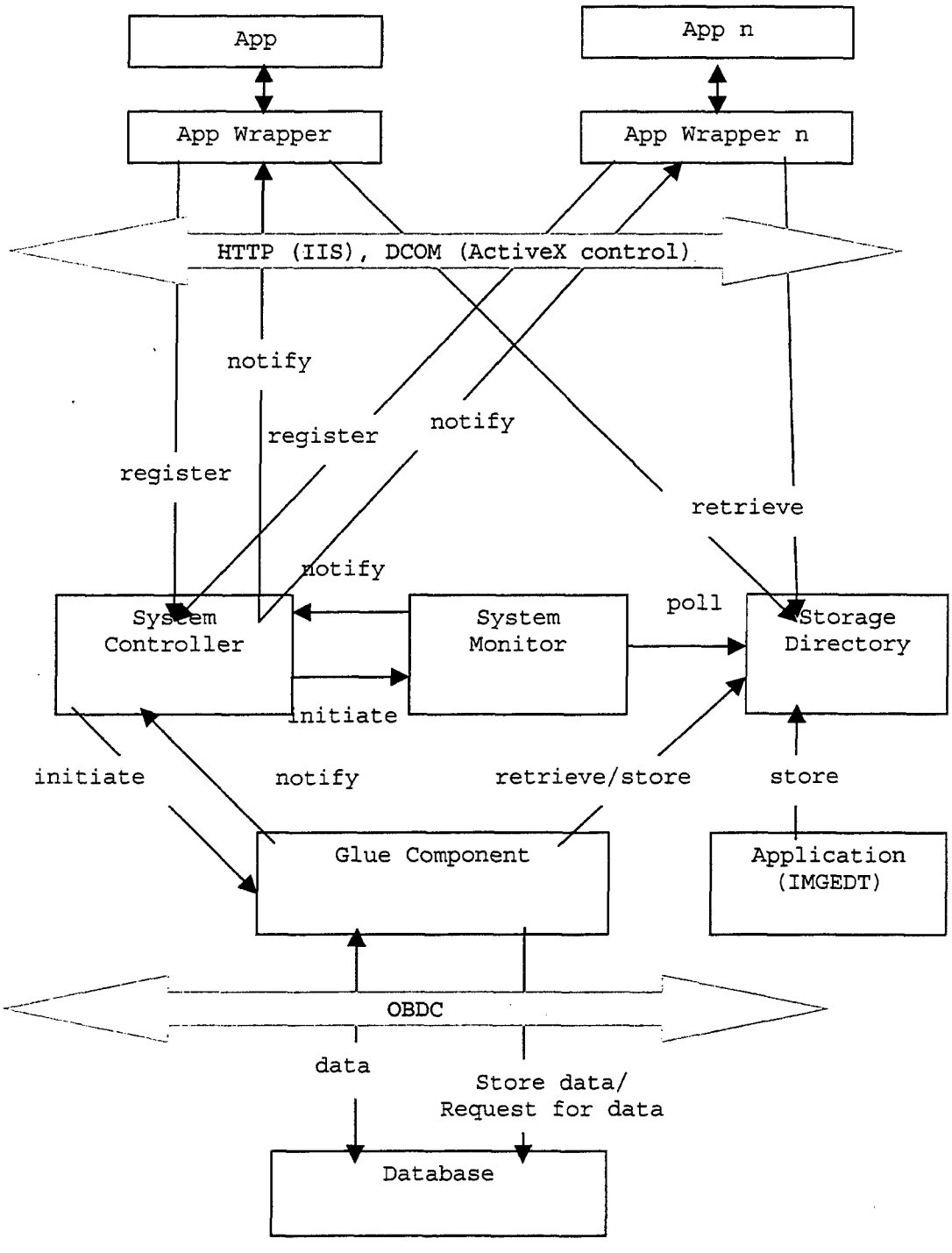

Figure 5.2 Component Integration DCOM Wrappers

Component security is based on external DCOM security features. External DCOM security provides the following advantages over internal DCOM security:

- Source code, object code or DLLs are not required. External security can be used when only executables are available.
- Since security policy is not embedded within components, components may be reused in security environments.
- Security policy can be implemented without writing any code or understanding component internals.

The case study focuses on two COTS applications within the operational NITES system. The first application, called image editor, produces a product. The second application, called continuous brief, presents a product. The image editor creates a file in a known directory. The file extension identifies the file type. The file is saved in a central relational database. This conforms to a design philosophy of NITES that each application interfaces with the database and not with each other.

The continuous brief loops through a set of the latest weather satellite images. The satellite images are extracted from the database. Continuous brief parameters include the number of images, viewing duration of each image, and image viewing dimensions.

Each application fits the three-tiered architecture of presentation, logic, and database. The presentation and logic tiers run on a PC with Windows NT. The database tier runs on Sun Solaris.

COM/DCOM is used to interface logic components on the PC. ADO/ODBC is used to interface to the relational database.

The Extensible Markup Language (XML) is used to wrap the data products in the relational database.

B. PRODUCE PRODUCTS TO DIRECTORY: IMAGE EDITOR (IMGEDT)

IMGEDT is a legacy NITES application that will be used to demonstrate the effectiveness of the design pattern produce products to directory. It is assumed only the executable is available, dynamic link library (DLL) substitution is not an option, and driver chaining will not be used.

IMGEDT is a Windows NT desktop application with no network or database connectivity. IMGEDT is capable of opening an image file, editing an image file and saving an image file to the local directory system.

The user signs on locally using id and password. The user has system privileges and object permissions to execute IMGEDT, read an image file and store an image file to a directory. Windows NT provides authentication and access control services.

Figure 5.3 shows the product producer sequence diagram. It is the responsibility of the System Monitor to poll the IMGEDT target directory for new or updated image files. It is assumed the IMGEDT target directory is located on a shared drive within an intranet and that the shared drive is accessible to the System Monitor. When a file is detected, the System Monitor initiates the sequence to store the image on a remote relational database.

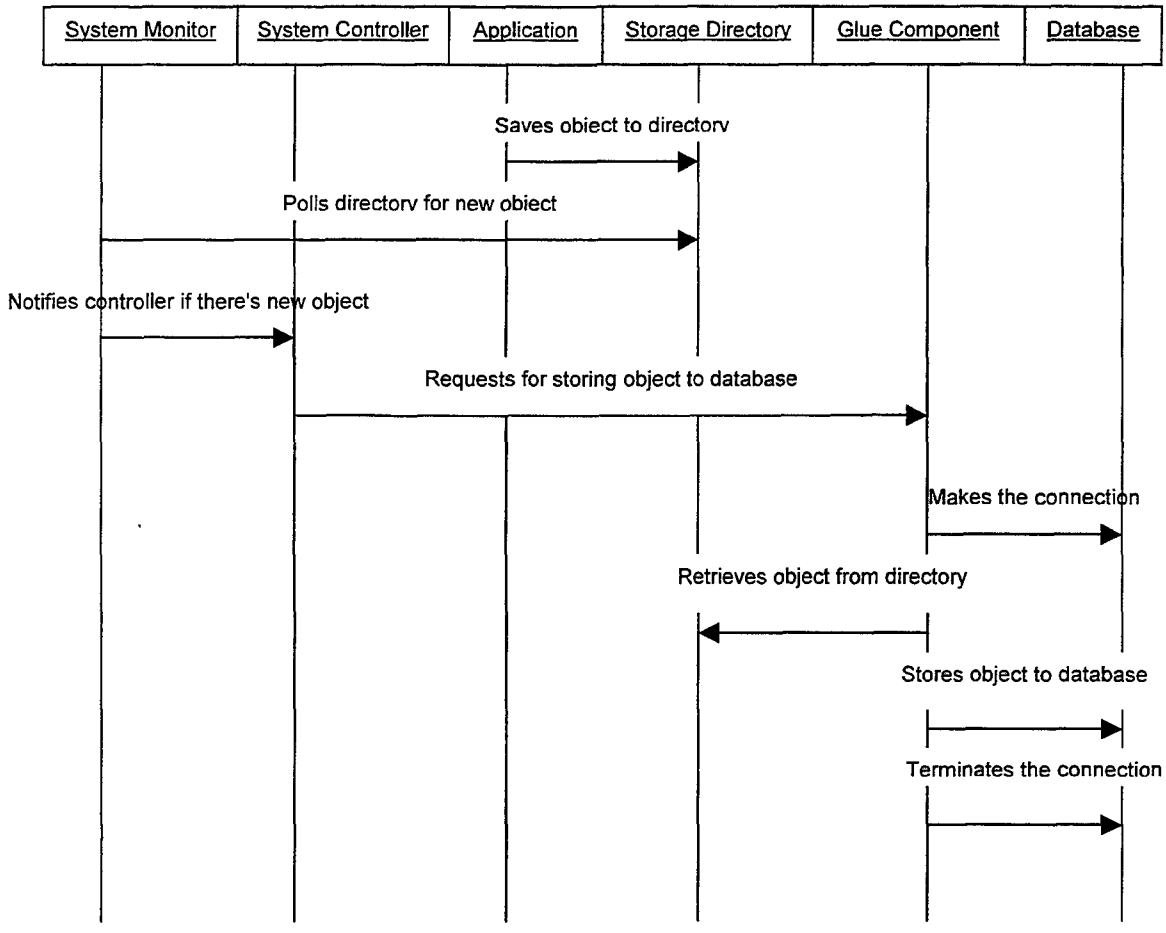

Figure 5.3 Store object into Database

Following is a detailed explanation of each step in the sequence diagram.

1. The application saves an object to the storage directory.
2. Concurrent to step 1, the system monitor periodically polls the storage directory for a new or updated object.
3. Access to the object is allowed only if the system monitor has read permission.

4. The system monitor notifies the system controller if there is new object.
5. The glue component establishes a remote connection to the relational database.
6. The glue component updates the database.
7. The relational database commits the object to the database after the command is successfully processed.
8. The glue component terminates the remote connection to the relational database.

C. DISPLAY PRODUCTS: CONTINUOUS BRIEF

The goals of the continuous brief case study are:

1. Prove that the presented wrapper and security architecture is feasible in the context of an existing system.
2. Measure performance impact due to security and wrappers.
3. Formalize the case study into a pattern for future projects.

The continuous brief is composed of the following objects:

1. Web Browser
2. PowerPoint as an ActiveX Document embedded within a browser.
3. PowerPoint Application wrapper that utilizes PowerPoint API.
4. Control that coordinates activities within the system

5. Communications that provide inter-component messaging facilities.
6. Database that provides storage and retrieval of row sets using SQL.
7. IMGNT application that interfaces with the database for storing and retrieving images.

1. Continuous Brief Initialization

Figure 5.4 shows the sequence of actions performed by cooperating objects to initialize the continuous brief.

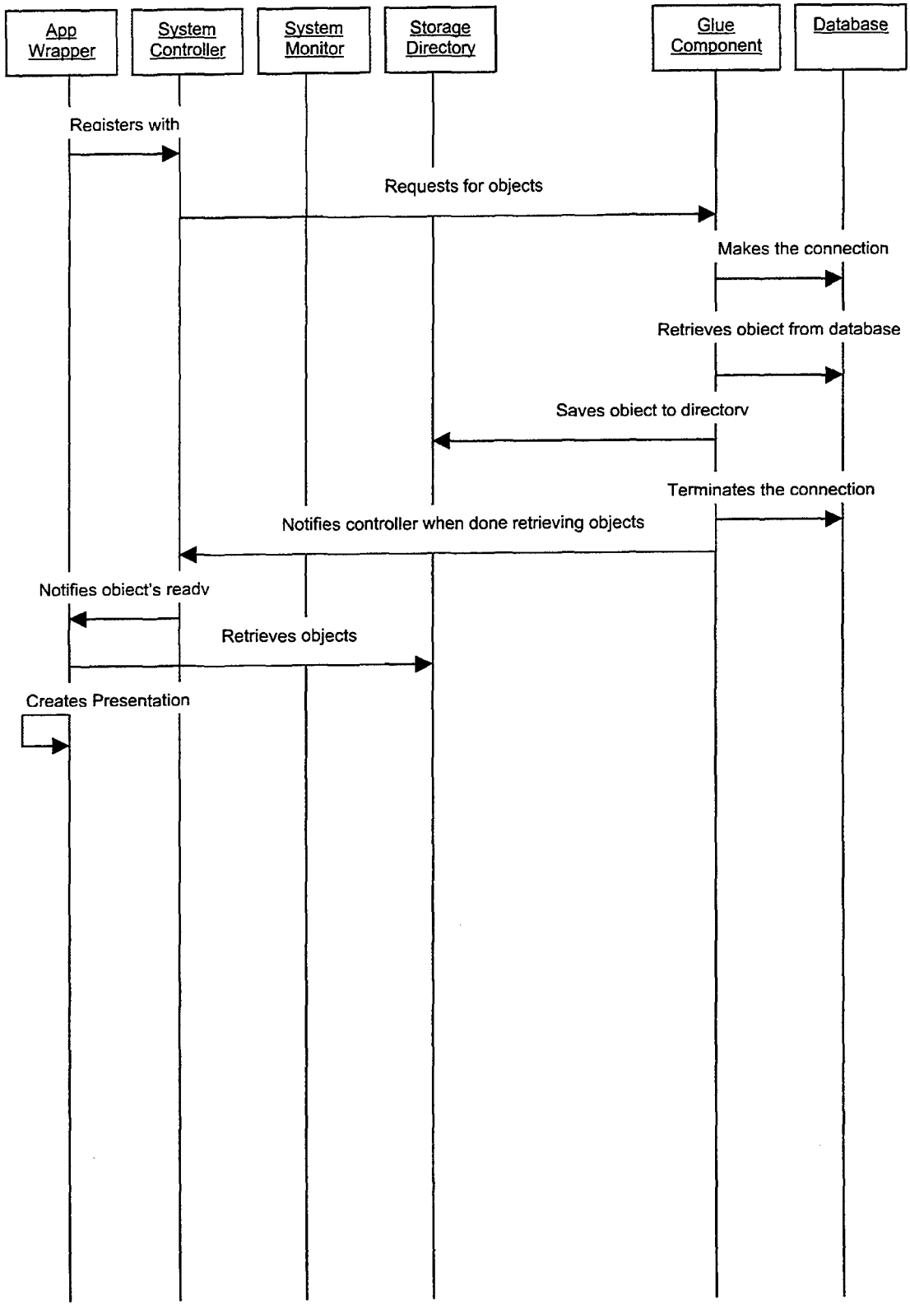

Figure 5.4. Continuous Brief Initialization Sequence Diagram

Following is a description of the diagram:

1. User registers to the web server. User authentication scheme will depend on user role and user location.
2. If user is authenticated, the web server sends the Initialization GUI home page containing parameters to be filled in.
3. The user fills in the number of images starting from the most current, the display duration of each image in seconds and the height and width of the display area. Default values are 24 images, 0 second duration, and display area equal to the screen size.
4. The web Server initiates the application wrapper and passes input parameters.
5. The application wrapper registers interest in new satellite images with the controller. The controller will notify all registered application wrappers when a new satellite image has been stored into the database.
6. The application wrapper requests the latest requested number of images from the database.
7. The glue component transforms the request into an asynchronous database query.
8. The database returns the requested images in a tif, jpeg or mif file format. The time the satellite image was photographed is part of the file name.
9. The glue component saves the requested images to the storage directory.

10. The application wrapper downloads the images via the current HTTP connection.

11. The application wrapper uses the PPT API to generate and show a continuous brief.

2. Continuous Brief Update

Figure 5.5 shows the sequence of actions performed by cooperating objects to update the continuous brief.

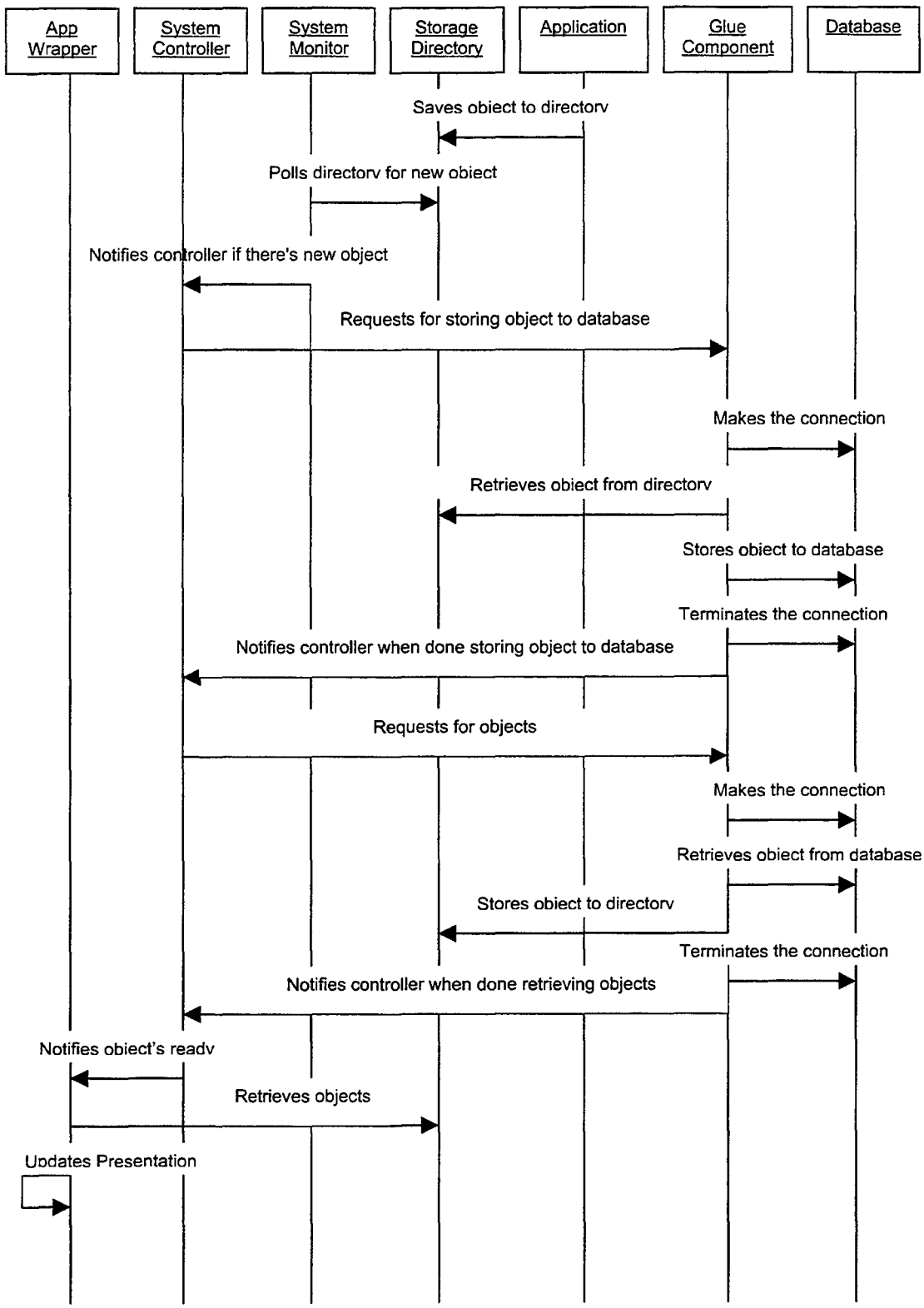

Figure 5.5 Continuous Brief Update Sequence Diagram

It is assumed that the App wrapper is embedded in the browser on the client machine. Following is a description of the diagram:

1. The Application saves new object to the storage directory.
2. The system monitor notifies system controller there is new object.
3. Controller forwards request to Glue component.
4. Glue component marshals request for database query and sends request using ODBC protocol.
5. Database processes request and stores the new object.
6. Glue component notifies controller that a new object has been inserted into the database.
7. System controller requests Glue component for objects.
8. Glue component initiates retrieval of objects from database.
9. Glue component notifies system controller when retrieval is completed.
10. Controller notifies registered App wrappers that new objects are available.
11. App wrapper updates presentation with new objects.

The Observer Pattern, as described in *Design Patterns*, also classifies this type of application. The subject is the satellite image section of the database and the observer is the application wrapper. The loose coupling between the database and the wrapper allows multiple wrappers to receive notification of a new satellite image.

3. User Interface

Before the brief is started, the user is prompted for the following parameters:

- The type of brief. Default is visual.
- Number of images in brief (1-99). Default 24
- Duration of each image (0-20 seconds). Default 0.
- Image display dimensions (height and width in twips). Default is window size.

These parameters initialize the brief via the brief interfaces. Buttons are used to start and stop the brief. A reset button restores input parameters to default values.

4. Brief Interfaces

a) Image Interface

The image interface is mapped to the PowerPoint shape object interface. Each image in the brief share the following properties:

SetWidth (twips width);

Sets the width of the display area in twips for the image.

SetHeight (twips height);

Sets the height of the display area in twips for the image.

Each image is sized to fit the display area.

b) Images Interface

The images interface is mapped to the PowerPoint slides object interface. The interface manages the images in the brief.

SetNumberOfImages (integer nImages);

Sets the number of images in the brief.

AddImage (picture image);

Adds the given image to the end of the brief. The images should be added in time sequence from the oldest to the newest.

c) Show Interface

The show interface is mapped to the PowerPoint show object interface. The interface manages the sequential display of each image in the brief.

SetImageDuration (integer seconds);

Sets the number of seconds that each slide is diplayed.

StartShow ();

Display images from first to last and repeat image sequence until show is stopped.

StopShow ();

Stop continuous brief.

D. DCOM DEPLOYMENT INSTRUCTIONS

The Visual Basic development environment provides tools to create a deployment package for ActiveX Exe remote servers. The remote server check box inside the project/properties/component section needs to be checked. Making the project using Files/Make creates an executable file (EXE), assigns a globally unique class ids and interfaces ids, and registers the component on the local machine. To avoid creation of new global identifiers each time the component is made, set the version compatibility to binary compatibility using the projects/properties/component pane. New global identifiers are only necessary when the interface definition changes. The package and deployment wizard steps you through the process of creating a deployment package. Since the target machine does not usually contain a development environment, the Visual Basic run time environment must be included in the deployment package. If the remote server component creates other components, the Visual Basic Reference file (VBR) and Type Library (TLB) must also be included in the deployment package.

Transfer the deployment package to the target machine and execute the setup application. Setup will register the component in the registry, copy dependent files to the appropriate system directory and update the programs folder.

Run DCOMCNFG on the server machine. The DCOM server check box needs to be checked in order for the DCOM server to run. Find the application name from the list of applications, and select properties. The location is local machine. The security setting controls user roles

that have privileges to launch, attach or change ownership of the remote server. The identification section is used to enter the user account and user password that will be used to launch the component. The protocol section is used to list the protocols to use in priority sequence.

Run DCONCNFG on the client machine. The DCOM server check box needs to be checked in order for the DCOM server to run. Find the server application name from the list of applications, and select properties. The location is the name of the remote server machine. The security setting controls user roles that have privileges to launch, attach or change ownership of the client component. The identification section is used to enter the user account and user password that will be used to launch the component. The protocol section is used to list the protocols to use in priority sequence.

The client is now ready to launch or attach to the remote server component. There is no need to manually start the server component. When the client creates a new the server component, the server component is launched on the remote machine.

Use the internet package option of the Package and Deployment Wizard to deploy an ActiveX control to the Web Server. This creates a CAB file containing the control and its dependencies. The CAB file is compressed to reduce download time. During the initial download, the ActiveX control is saved and registered on the client. Subsequent references to the control are resolved locally.

VI. CONCLUSIONS

The following conclusions are based on application of the distributed component integration methodology (DCIM) to the case study.

A. DCOM SOLUTION

DCOM is a natural choice for this implementation. The host machine is a PC running Windows NT and DCOM is bundled with the OS. There is familiarity with DCOM from prior projects. Visual Basic development environment hides low-level plumbing from the developer. Security policy can be defined external to the component implementation. The existing design pattern template fit the design of the continuous brief application.

DCOM proved to be a quick and efficient way to implement a robust continuous brief application. Components were tested in the VB debug environment. Then executables were tested on a single machine. Finally, the system was distributed to the Web server machine. No source code changes were made to execute in these three configurations.

B. ARCHITECTURAL DESIGN

The architectural design with accompanying VB application framework skeleton code proved to simplify implementation. The details of object creation, push technology, client registration for service, event processing, browser based components, asynchronous object execution, and polling were provided by the framework.

The framework was extended to poll a directory, make asynchronous database queries, add arguments to events, wrap PowerPoint and add a

user interface. The developer is able to focus on the application without being distracted by plumbing details.

C. WRAPPERS

Three types of wrappers were used in the implementation of the continuous brief: file type in directory, object, and COTS API. The monitor component of the architectural design was extended to periodically check for a new satellite image file in a directory specified by the configuration utility. The object wrapper used the file name structure to extract image time, type and location. The PowerPoint API was used show the continuous brief. Even though the show could have been easily implemented using a Java applet, PowerPoint could simplify future extensions such as image cropping and image titling.

To eliminate the need for PowerPoint on each client, the show could have been generated on the server and sent to the client for viewing. Microsoft provides a web based PowerPoint viewer free of charge.

D. SECURITY

The external security features of DCOM proved to simplify implementation of security policy; however Windows NT Service Pack 5 does not expose DCE encryption to external DCOM security. Single user logon, user privileges based on role and discretionary access control were available.

E. IMGNT

Administrative problems precluded the use of ImgNT to retrieve selected images from a database and store in a directory. The system had not been installed on an unclassified system, Visual Basic was not available, and ImgNT patches had not been made. It is assumed that ImgNT had already stored requested images to a directory.

F. FUTURE TRENDS

The value of the results of this thesis is time sensitive. Research on this thesis began in April 1999. Since that time Microsoft has released Windows 2000, SPAWAR has unveiled a public key infrastructure for e-mail, SPAWAR has a draft security policy, a network centric architecture has been deployed to the USS Coronado, CORBA has a wider selection of commercial ORBs, new standards for wireless communications have been developed, Linux is gaining support from many communities, security measures are receiving higher priority and many other innovations.

The distributed component integration methodology described in the thesis will remain in the mainstream for the foreseeable future. Independently designed components will need custom integration using some form of wrapper. Network administrators will require implementation of security policy using tools external to the application.

LIST OF REFERENCES

[1] Berzins V., Luqi, Schultes B. *JBC Report*, Naval Post Graduate School, 1999

[2] Ashley,P., Practical Intranet Security, Kluwer Academic Publishers, 1999

[3] Summers Rita C., *Secure Computing*, McGraw-Hill, 1997

[4] Grimes, R., Professional DCOM Programming, WROX, 1997

[5] Microsoft Corporation, *Entire Collection*, MSDN Library, 1996

[6] Krause M.,*Handbook of Information Security Management*, Auerbach, 1999

[7] Phaltankar K., Implementing Secure Intranets and Extranets, Artech House, 2000

[8] Moultis N., Kirk C., *XML Black Book*, Coriolis Technology Press, 1999

[9] Szyperski, Clemens, *Component Software*, Addison-Wesley, 1998

BIBLIOGRAPHY

Berzins and Luqi, *Software Engineering with Abstractions,* Addison-Wesley, 1991

Douglas B., Real-Time UML, Addison-Wesley, 1998

Gamma E., Helm R., Johnson R., Vlissides J., *Design Patterns CD,* Addison-Wesley, 1995

http://www.esat.kuleuven.ac.be/cosic/sesame3_2.html

APPENDIX A. GSS-API VERSION 2 FUNCTION CALLS

CREDENTIAL MANAGEMENT

GSS_Acquire_cred	acquire credentials for use.
GSS_Inquire_cred	display information about credentials.
GSS_Release_cred	release credentials after use.

CONTEXT-LEVEL CALLS

GSS_Init_sec_context	initiate outbound security context.
GSS_Accept_sec_context	accept inbound security context
GSS_Delete_sec_context	flush context.
GSS_Process_context_token	process received control token on context.
GSS_Context_time	indicate validity time remaining in context.

PER-MESSAGE CALLS

GSS_GetMIC	apply signature, receive as token separate from message.
GSS_VerifyMIC	validate signature token along

	with message.
GSS_Wrap	sign, optionally encrypt and encapsulate.
GSS_Unwrap	decapsulate, decrypt if needed, validate signature.

SUPPORT CALLS

GSS_Display_status	translate status codes to printable form.
GSS_Compare_name	compare two names for equality
GSS_Display_name	translate name to printable form.
GSS_Import_name	convert printable name to normalized form.
GSS_Release_name	free storage of normalized-form name.
GSS_Release_buffer	free storage of printable name
GSS_Release_oid_set	free storage of OID set object

APPENDIX B. SESAME CRYPTOGRAPHIC SUPPORT FACILITY (CSF) APIS

INITIALIZATION APIS

csf_get_qos()

Returns the list of allowed pairs of algorithms with associated key length, for a given quality of service, within a given CSF domain such as "quality of service". The first algorithm and key length pair represent the default.

A quality of service is

- A service (integrity or confidentiality),
- A strength (weak, medium or strong),
- A class of algorithms (symmetric or asymmetric)

csf_begin()

Starts CSF up for a given algorithm. This API is used to initialize internal data for a software algorithm, or to set-up a hardware device.

csf_end()

Turns off CSF for a given algorithm. This API is used to free internal data for a software algorithm, or to shut down a hardware device.

Key generation APIs

A key handle is generated by these APIs.

csf_gen_asym_key_pair()

Generates an asymmetric key pair with the key length, key data and the reversible cryptographic algorithm as parameters.

csf_gen_sym_key()

Generates a symmetric key with the key length, key data and the reversible cryptographic algorithm as parameters.

csf_derive_secret_key()

This API is used to derive a secret key of a given key length from a string or a basic key, using an irreversible encryption algorithm and a seed.

Key handling

csf_init_key()

Initializes the key to be used by the CSF module. An indication on the way the key is stored (hardware, software, smart card ...), on the way the key is used (encryption, decryption, signature key or a key to check a signature) and the key itself or a reference of that key is given in input. It returns an opaque key handle to be used by subsequent calls to CSF APIs.

csf_release_key()

Releases an opaque key handle.

csf_read_key_info()

Allows to retrieve a key or a key reference from a key handle.

csf_get_key_data()

Allows to retrieve key data (key usage and optionally key validity time, initial vector) from a key handle.

Crypto context APIs

csf_init_context()

Initializes a crypto context from a CSF key handle and a pair of algorithms (reversible or irreversible) and associated key length. This context contains elements (hardware or software) to be used in data protection operations. It returns an opaque context handle to be used by subsequent data protection CSF APIs.

If the crypto context already exists, it is modified according to the input parameters.

csf_create_owf_context()

Creates a CSF context, only usable for an irreversible encryption algorithm which does not use any key, such as MD4 or MD5. No key handle is needed to use this interface.

csf_release_context()

Releases an opaque CSF context handle.

csf_duplicate_context()

Duplicates an existing crypto context. A new context handle is generated. The new context can then be modified by a call to csf_init_context().

csf_retrieve_key_from_context()

Returns the key handle attached to a crypto context.

csf_query_context()

Returns the pair of algorithms (irreversible + reversible) with associated key length and the quality of service attached to a crypto context.

Data protection APIs

csf_encrypt()

Generates an encrypted text from a clear text and a crypto context (including a key, a reversible algorithm and optionaly initial vectors).

csf_decrypt()

Generates a clear text from an encrypted text using a crypto context (including a key and a reversible algorithm).

csf_generate_check_value()

Generates a signature from a clear text using a crypto context (including a key (private or secret), an irreversible algorithm and a reversible one).

csf_verify_check_value()

Checks the signature of a clear text using a crypto context (including a key (public or secret), an irreversible algorithm and a reversible one).

csf_owf()

Generates an irreversibly encrypted text from a clear text using a crypto context (including an irreversible algorithm).

Import/export APIs

csf_extract_key()

Packs the key and all data relative to the key (key usage, key validity) into an exportable format. This package has to be sent to the remote machine. csf_restore_key() has then to be called on this machine to restore the key information.

csf_restore_key()

Creates a key handle from a package obtained by an earlier call to csf_extract_key(), usually on another machine.

csf_extract_context()

Packs the key and all data relative to the crypto context (key usage, key validity, pair of algorithms) into an exportable format. This package has to be sent to the remote machine. csf_restore_context() has then to be called on this machine to restore the context information.

csf_restore_context()

Creates a key handle from a package obtained by an earlier call to csf_extract_key(), usually on another machine.

RANDOM NUMBER GENERATION API

csf_gen_rand_num()

Generates a random number of a given length.

Free routines

free_key_info()

Free a key (A key_info_t structure).

free_key_data()

Free key data (a key_data_t structure).

free_algo_id ()

Free an algorithm (an algo_identifier_t structure).

free_algo_id_pair()

Free a pair of algorithms (an algo_id_pair_t structure).

free_algo_id_pair_list ()

Free a list of algorithms (an algo_id_pair_list_t structure).

free_algo_list_except_one()

Free a list of algorithms, except one pair in the list.

SET-UP AND CONFIGURATION

Set-up and configuration of the CSF module is done by a control program called csfcp.

The CSF administrator is the only person authorized to run this program.

csfcp is be used to:

- Configure the quality of service, within the local domain. A list of allowed pairs of algorithm identifiers (irreversible or reversible) is to be associated to each qos.
- Configure the quality of service which is to be used to communicate between two CSF domains. A subset of the local qos configuration can be chosen and then sent to the second domain.
- Set-up all the algorithms available under CSF. For all available algorithms, the choice between hardware and software is made, for key storage and algorithm implementation.

APPENDIX C. SESAME ARCHITECTURE

A. PROTOCOL NOTATIONS

A	Authentication Server
P	Privilege Attribute Server
U	User Sponsor
R	User
X	Client Application
Y	Server Application
Z	Server Application accesses by delegate
V	PAC validation facility of application server Y
W	PAC validation facility of application server Z
K_{AB}	Long term key shared between A and B
k_{AB}	Session key shared between A and B
PK_A	Public key of A
PK_A^{-1}	Private key of A
$ReQPriv_R$	Requested privileges by user R sealed by k_{UP}
$Cert_i$	X.509 certificate for the public key Pk_i
RL_x	Requested lifetime for x
T_s, T_e	Start and end time
r_i	Nonce generated by i
n_i	Message sequence number
h()	Hash function

$KeyPK_{i-j-k} = ENC(PK_j)(k_{jk}, T_s, T_e, data)$

$KeyPK_{j-k}$ = ENC(PK_k) (k_{jk}, T_s, T_e, data)

AuthSK_{i-j} = ENC(k_{ij}) (j, t_i, data)

AuthPK_{i-j} = SIGN(Pk_i^{-1}) (j, t_i, *Key*PK_{i-j})

B. USER SPONSOR FUNCTIONS

- Sends an authenticator SIGN(PK_R^{-1}) (A, t_R, *Key*(PK_{R-A}) to the Authentication Server.
- Decrypts the incoming key package from AS using the user's private key.
- Sends a request for a PAC to the privilege attribute server. The request contains the requested lifetime of the PAC, TGT, session key authenticator ENC(k_{U-P}) (P, t_U, data).

C. AUTHENTICATION PRIVILEGE ATTRIBUTE CLIENT (APA)

The APA is developed by a programmer using the GSS-API. The User Sponsor uses this API to communicate with the authentication server and privilege attribute server to obtain authentication and credentials. See Appendix A for a description of GSS-API.

D. APPLICATION CLIENT

Every application client needs to be modified to include GSS-API.

1. Authentication Server (AS) Functions

Checks the X.509 certificate for the public key of user ($Cert_R$).

Verifies the authenticator portions of $Cert_R$.

Returns an authentication which includes the Primary Principal Identifier (PPID) as part of the ticket granting ticket (TGT),

and an authenticator containing the public key of the privilege attribute server (PAS)

$TGT_R = ENC(K_{AP})(R, U, T_s, T_e, k_{UP})$

$PAC_R = SIGN(PK_P^{-1})$(user role attributes, $PPID_R$, PV_R, DTQ_R, data)

E. PRIVILEGE ATTRIBUTE SERVER (PAS) FUNCTIONS

Supplies PAC as specified in ECMA 219 Security in Open Systems, 2nd edition, March 1996. European Computer Manufactures Association

F. KEY DISTRIBUTION SERVER (KDS)

- For the intra-domain case use Kerberos V5 model.
- For the inter-domain case use X.509 certificates.

G. PRIVILEGE ACCOUNT CERTIFICATE (PAC) VALIDATION FACILITY (PVF) FUNCTIONS

- Validate PAC
- Key Management

Support Components

- Audit
- Record security relevant events using appropriate identities.

H. PUBLIC KEY MANAGEMENT (PKM) FUNCTIONS

- Manage public and private keys using PGP solution
- Establish symmetric keys between parties *i* and *j* using public-key standard X.509.

i sends a session key to *j* encrypted with *j*'s public key. *i* sends an authenticator using its private key. *J* authenticates the message signature by applying *i*'s public key and comparing the message with the message signature. The session key is now available to both parties.

APPENDIX D. SKELETON VB CODE FOR DESIGN PATTERN

A. MONITOR COMPONENT

1. Modules

a. Module 1

```
Option Explicit
Public gMonitor As Monitor          ' Reference to monitor
Public glngUseCount As Long         ' Global reference count
```

2. Classes

a. Monitor

```
Option Explicit

Private mFormForTimer As FormForTimer
Private WithEvents mTimerForMonitor As Timer

Public Enum Enumeration
    enum1 = 1
    enum2 = 2
    enum3 = 3
End Enum

' Event that passes all automation data types supported by
' proxy and stub
Event MonitorActivity( _
                      bool As Boolean, _
                      chr As Byte, _
                      sfloat As Single, _
                      dfloat As Double, _
                      sint As Integer, _
                      lint As Long, _
                      enum123 As Enumeration, _
                      str As String, _
                      money As Currency, _
                      datetime As Date)


Private Sub Class_Initialize() ' Start Monitor Timer


   ' Create instance of form
   Set mFormForTimer = New FormForTimer
   Load mFormForTimer
```

```
    ' Connect timers' events to associated event procedures
    ' in Monitor
    Set mTimerForMonitor = mFormForTimer.TimerForMonitor

End Sub

Private Sub Class_Terminate()    ' Terminate Monitor
    Set mTimerForMonitor = Nothing
    Unload mFormForTimer
    Set mFormForTimer = Nothing
End Sub

Private Sub mTimerForMonitor_Timer()     ' Process Timer Event

    Dim bool As Boolean
    Dim chr As Byte
    Dim sfloat As Single
    Dim dfloat As Double
    Dim sint As Integer
    Dim lint As Long
    Dim enum123 As Enumeration
    Dim str As String
    Dim money As Currency
    Dim datetime As Date

    '<insert monitor task>

    ' Signal clients that monitor has detected activity
    RaiseEvent MonitorActivity(bool, _
                               chr, _
                               sfloat, _
                               dfloat, _
                               sint, _
                               lint, _
                               enum123, _
                               str, _
                               money, _
                               datetime)


End Sub
```

b. Monitor Connector

```
Option Explicit

Public Property Get Monitor() As Monitor ' Get reference to
                                         ' monitor
```

```
    Set Monitor = gMonitor
End Property

Private Sub Class_Initialize() ' Create Monitor and
                               ' reference count
   If gMonitor Is Nothing Then
      Set gMonitor = New Monitor
   End If
   glngUseCount = glngUseCount + 1
End Sub

Private Sub Class_Terminate()    ' Terminate Monitor when
                                 ' reference count = 0
   glngUseCount = glngUseCount - 1
   If glngUseCount = 0 Then
      Set gMonitor = Nothing
   End If
End Sub
```

B. CONTROLLER COMPONENT

1. Modules

a. Module 1

```
Option Explicit
Public gController As Controller    ' Reference to controller
Public glngUseCount As Long         ' Global reference count
```

2. Classes

a. Controller

```
Option Explicit

Event ControllerEvent()          ' Sent to AppWrapper(s)

Public WithEvents mglue As Glue ' WithEvents causes glue to
                                ' run asynchronously
Private WithEvents mMonitor As Monitor ' Get Monitor events

' Multiple connections to single monitor
Private mMonitorConnector As MonitorConnector

Private Sub Class_Initialize()  ' Connect to Monitor

  Set mMonitorConnector = New MonitorConnector
  Set mMonitor = mMonitorConnector.Monitor
```

```
End Sub
' Receive event from Monitor
Private Sub mMonitor_MonitorActivity( _
                      bool As Boolean, _
                      chr As Byte, _
                      sfloat As Single, _
                      dfloat As Double, _
                      sint As Integer, _
                      lint As Long, _
                      enum123 As Enumeration, _
                      str As String, _
                      money As Currency, _
                      datetime As Date)

    Set mglue = New Glue
    Call mglue.StartGlue              ' Glue runs asynchronously
End Sub

Private Sub mglue_glueDone()     ' Asynchronous glue component is done

Set mglue = Nothing
RaiseEvent ControllerEvent
End Sub
```

b. Controller Connector

```
Option Explicit

Public Property Get Controller() As Controller
    Set Controller = gController
End Property



Private Sub Class_Initialize()     ' Initialize Controller
                                   ' and reference count
   If gController Is Nothing Then
      Set gController = New Controller
   End If
   glngUseCount = glngUseCount + 1
End Sub

Private Sub Class_Terminate()    ' Terminate controller when reference
count = 0
   glngUseCount = glngUseCount - 1
   If glngUseCount = 0 Then
      Set gController = Nothing
   End If
End Sub
```

C. GLUE COMPONENT

1. Classes

a. Glue

```
Option Explicit

Event GlueDone()          ' Sent when glue task done

Public Sub StartGlue()   ' Start glue task
    ' <Insert glue task here>
    RaiseEvent GlueDone
End Sub
```

D. APPLICATION WRAPPER COMPONENT

1. Forms

```
Option Explicit

Private WithEvents mController As Controller
Private mControllerConnector As ControllerConnector

Private Sub Form_Load()      ' Connect to controller
    Set mControllerConnector = New ControllerConnector
    Set mController = mControllerConnector.Controller

End Sub

' Receive Controller event
Private Sub mController_ControllerEvent()

Text1.Text = "Received Controller Notification"
    ' <insert interface with COTS application>
End Sub
```

APPENDIX E. XML VOCABULARIES

The following list contains sources for some existing XML vocabularies:

Mathematical Markup Language (MathML) can be found at URL www.w3.org/Math

Web Interface Definition Language (WIDL) can be found at URL www.webmethods.com/technology/widl_description.html

The Nites I Meteorological Vocabulary Observation Markup Format(OMF):

```
<!-- <!DOCTYPE OMF SYSTEM "OMF.dtd" [ -->
<!-- Weather Observation Definition Format DTD -->
<!-- This is the OMF XML DTD. It can be referred to using the
formal public identifier
-//METNET//OMF 1.0//EN
For description, see OMF.html
$Id: OMF.dtd,v 3.8 1999/10/25 18:18:31 oleg Exp oleg $
-->
<!-- Weather Observation Definition Format -->
<!-- Basic attributes -->
<!ENTITY % TStamp-type "NMTOKEN">
<!ENTITY % TRange-type "CDATA">
<!ENTITY % TStamp "TStamp %TStamp-type; #REQUIRED">
<!ENTITY % TRange "TRange %TRange-type; #REQUIRED">
<!ENTITY % LatLon "LatLon CDATA #REQUIRED">
<!ENTITY % LatLons "LatLons CDATA #REQUIRED">
<!ENTITY % BBox-REQD "BBox CDATA #REQUIRED">
<!ENTITY % BBox-OPT "BBox CDATA #IMPLIED">
<!ENTITY % BId "BId NMTOKEN #REQUIRED">
<!ENTITY % SName "SName CDATA #REQUIRED">
<!ENTITY % Elev "Elev NMTOKEN #IMPLIED">
<!-- Basic elements -->
<!ELEMENT VALID (#PCDATA)>
<!ATTLIST VALID %TRange;>
<!-- A collection of weather observation reports -->
<!ELEMENT Reports ( METAR | SPECI | UAR | BTSC | SYN )*>
<!ATTLIST Reports %TStamp;>
<!-- Common report attributes -->
<!ENTITY % ReportAttrs
"%TStamp; %LatLon; %BId; %SName; %Elev;
```

```
Vis NMTOKEN #IMPLIED
Ceiling NMTOKEN #IMPLIED
">
<!-- METAR and SPECI reports -->
<!ELEMENT METAR (#PCDATA)>
<!ATTLIST METAR %ReportAttrs;>
<!ELEMENT SPECI (#PCDATA)>

<!ATTLIST SPECI %ReportAttrs;>
<!-- A collection of weather hazard advisories -->
<!ELEMENT Advisories ( SIGMET | AIRMET | WW )* >
<!ATTLIST Advisories %TStamp;>
<!-- A SIGMET advisory -->
<!ELEMENT SIGMET (VALID, AFFECTING?, EXTENT, BODY) >
<!ATTLIST SIGMET
class (CONVECTIVE| HOTEL| INDIA| UNIFORM| VICTOR| WHISKEY) #REQUIRED
id NMTOKEN #REQUIRED
%TStamp;
%BBox-OPT;
>
<!ELEMENT AFFECTING (#PCDATA)>
<!ELEMENT EXTENT (#PCDATA)>
<!ATTLIST EXTENT
Shape (AREA| LINE| POINT) #REQUIRED
%LatLons;
>
<!ELEMENT BODY (#PCDATA)>
<!-- A collection of weather forecasts -->
<!ELEMENT Forecasts ( TAF )* >
<!ATTLIST Forecasts %TStamp;>
<!-- A Terminal Aerodrome Forecast -->
<!ELEMENT TAF ( VALID, PERIOD+ ) >
<!ATTLIST TAF
%TStamp; %LatLon; %BId; %SName;
>
<!ELEMENT PERIOD ( PREVAILING, VAR* )>
<!ATTLIST PERIOD
%TRange;
Title NMTOKEN #IMPLIED
>
<!ELEMENT PREVAILING (#PCDATA)>
<!ELEMENT VAR (#PCDATA)>
<!ATTLIST VAR
%TRange;
Title CDATA #REQUIRED
>
<!-- Rawinsonde and Pibal Observation reports -->
<!ELEMENT UAR ( UAPART+, UAID*, UACODE*, UALEVELS ) >
<!ATTLIST UAR
%TStamp; %LatLon; %BId; %SName; %Elev;
```

```
>
<!ELEMENT UAPART (#PCDATA)>
<!ATTLIST UAPART
id NMTOKEN #REQUIRED
>
<!ENTITY % UARef "Ref NMTOKEN #REQUIRED">
<!ELEMENT UAID (#PCDATA)>
<!ATTLIST UAID %UARef; >
<!ELEMENT UACODE (#PCDATA)>
<!ATTLIST UACODE %UARef; >
<!ELEMENT UALEVELS (UALEVEL)*>
<!ELEMENT UALEVEL (#PCDATA)>
<!ATTLIST UALEVEL
%UARef;
P NMTOKEN #REQUIRED
H NMTOKEN #IMPLIED
T NMTOKEN #IMPLIED
DP NMTOKEN #IMPLIED
Wind CDATA #IMPLIED
>
<!-- Bathythermal, Salinity and Ocean Currents Observations -->
<!ELEMENT BTSC ( BTID, BTCODE?, BTLEVELS ) >
<!ATTLIST BTSC
%TStamp; %LatLon; %BId; %SName;
Title (JJYY | KKXX | NNXX) #REQUIRED
Depth NMTOKEN #IMPLIED
>
<!ELEMENT BTID (#PCDATA)>
<!ATTLIST BTID
DZ (7|8) #IMPLIED
Rec NMTOKEN #IMPLIED
WS (0|1|2|3) #IMPLIED
Curr-s (2|3|4) #IMPLIED
Curr-d NMTOKEN #IMPLIED
AV-T (0|1|2|3) #IMPLIED
AV-Sal (0|1|2|3) #IMPLIED
AV-Curr (0|1|2|3) #IMPLIED
Sal (1|2|3) #IMPLIED
>
<!ELEMENT BTCODE (#PCDATA)>
<!ELEMENT BTLEVELS (BTAIR?, (BTLEVEL)*)>
<!ELEMENT BTAIR (#PCDATA)>
<!ATTLIST BTAIR
T NMTOKEN #IMPLIED
Wind CDATA #IMPLIED
>
<!ELEMENT BTLEVEL (#PCDATA)>
<!ATTLIST BTLEVEL
D NMTOKEN #REQUIRED
T NMTOKEN #IMPLIED
```

```
S NMTOKEN #IMPLIED
Curr CDATA #IMPLIED
>
<!-- Surface Synoptic Reports from land and sea stations -->
<!ELEMENT SYN ( SYID, SYCODE?, SYG?, SYSEA? ) >
<!ATTLIST SYN
%TStamp; %LatLon; %BId; %SName; %Elev;
Title (AAXX | BBXX | ZZYY) #REQUIRED
SType (AUTO | MANN) "MANN"
>
<!ELEMENT SYID (#PCDATA)>
<!ATTLIST SYID
WS (0|1|3|4) #IMPLIED
>
<!ELEMENT SYCODE (#PCDATA)>
<!ELEMENT SYG (#PCDATA)>
<!ATTLIST SYG
T NMTOKEN #IMPLIED
TD NMTOKEN #IMPLIED
Hum NMTOKEN #IMPLIED
Tmm CDATA #IMPLIED
P NMTOKEN #IMPLIED
P0 NMTOKEN #IMPLIED
Pd NMTOKENS #IMPLIED
Vis NMTOKEN #IMPLIED
Ceiling NMTOKEN #IMPLIED
Wind CDATA #IMPLIED
WX CDATA #IMPLIED
Prec CDATA #IMPLIED
Clouds CDATA #IMPLIED
>
<!ELEMENT SYSEA (#PCDATA)>
<!ATTLIST SYSEA
T NMTOKEN #IMPLIED
Wave CDATA #IMPLIED
SDir CDATA #IMPLIED
>
<!-- Plain-text WMO Meteorological messages -->
<!ELEMENT Messages ( MSG )* >
<!ATTLIST Messages %TStamp;>
<!ELEMENT MSG ANY >
<!ATTLIST MSG
id NMTOKEN #REQUIRED
Type NMTOKEN #IMPLIED
%TStamp;
%SName;
%BBox-OPT;
BBB CDATA #IMPLIED
Descr CDATA #IMPLIED
><!-- ]> -->
```

APPENDIX F. SYSTEMS REQUIREMENTS SPECIFICATION

SOFTWARE REQUIREMENTS SPECIFICATION

FOR AN

ARCHITECTURAL FRAMEWORK

OF

DOD COTS/LEGACY SYSTEM

1. SCOPE

1.1 INTRODUCTION

The trend towards using Commercial Off-The-Shelf (COTS) software within Department of Defense (DoD) has become the accepted way to build systems. Twenty years ago, almost all DoD software-intensive systems were built by awarding large multimillion-dollar contracts to defense contractors to build these systems from scratch. In the 90's, with a constantly dwindling budget, the focus has shifted to building software-intensive systems by integrating COTS software components.

Building software systems from COTS components is quite different. The black box nature of the COTS software components along with the uncontrollable evolution process requires a different architectural approach in developing systems with COTS.

1.2 PURPOSE

The purpose of this requirements specification is to analyze and document the requirements in developing an architectural framework for COTS/Legacy systems within the DoD. To focus the requirements of the architectural framework, a DoD Meteorological and Oceanographic (METOC) system, the Naval Integrated Tactical Environmental System I (NITES I), which is very representative of today's DoD COTS/Legacy systems, will be used.

1.3 BACKGROUND

The NITES I project is a Space and Naval Warfare (SPAWAR) sponsored project within DoD. Like most other projects within DoD, the NITES I project is being developed in an environment that emphasizes the use of personal computers and COTS components.

NITES I acquires and assimilates various METOC data for use by US Navy and Marine Corps forecasters. The purpose of NITES I is to provide the METOC community (Users) with the tools necessary to support the warfighter (Customers).

The NITES I is the primary METOC data fusion platform and principal METOC analysis workstation, intended to be operated on both a classified and unclassified network environment by METOC personnel. This system receives, processes, stores and disseminates METOC data and provides analysis tools to render products for application to military and tactical operations. NITES I data and information/products are stored in a unified METOC database on the C4ISR network and available to local and remote planners and warfighters.

1.4 REFERENCES

Performance Specification (PS) for the Tactical Environmental Support System / Next Century TESS(NC) (AN/UMK-3) (NITES version I and II)

Security Guidelines for Space and Naval Warfare Systems Command (SPAWAR) Program Software Developers (DRAFT), October 1999.

Horizontal Integration: Windows NT Developer's Guidelines (DRAFT), Version 0.1.

2. GENERAL DESCRIPTION

2.1 ARCHITECTURE GOALS

Integration

COTS/GOTS/legacy components are usually created as standalone products. When these components are targeted for integration into a system, the architecture shall provide seamless integration of these COTS/GOTS/legacy components.

The architecture shall support middleware approaches to bind data, information and COTS/GOTS/legacy components.

Because evolution and upgrade of COTS/GOTS components are outside the control of the system integrators, the architecture of the COTS/GOTS/legacy system shall have an adaptable component configuration to reduce the effort of testing and reintegration when upgrades or new COTS/GOTS packages are introduced to the system.

INTEROPERABILITY

COTS/GOTS and legacy systems reside on multiple platforms. This architecture shall address distributed, heterogeneous systems consisting of both UNIX and PC-based platforms.

In order to achieve and maintain information superiority on the battlefield, the architectural framework for DoD COTS/GOTS/legacy systems shall have the capability to share, receive and transmit on heterogeneous networks and hardware devices.

The exchange of data between two systems shall be in such a way that interpretation of the data is precisely the same. The data displayed on two different systems shall remain consistent. The architectural framework shall include standard application program interfaces (APIs). APIs specify a complete interface between the application software and the platform across which all services are provided. A rigorous definition of the interface results in application portability provided the platform supports the API as specified, and the application uses the specified API. The API definitions shall include the syntax and semantics of the programmatic interface as well as the necessary protocol and data structure definitions.

ADOPTED FRAMEWORK TECHNOLOGY

Java/C++, web technologies, open systems, application program interfaces, common operating environment, object and component technology, commercial products and standards are all important to the COTS/GOTS/legacy system architecture.

The COTS/GOTS/legacy system shall adopt the Interface Definition Language (IDL) as the language for expressing the syntax of the framework services.

The COTS/GOTS/legacy system architecture shall be expressed as UML class and package diagrams, with detailed component descriptions using IDL with English narrative to provide semantics.

SECURITY

DoD tactical systems are normally classified to some security level. In building this architectural framework, the architecture shall address the DoD Trusted Computer System Evaluation Criteria (TCSEC) to at least the C2 security level.

The architecture shall include discretionary access control (DAC).

Only single level classification systems shall be supported in this architecture (i.e. no multi-level security (MLS).

Assembled components shall not require modification to add security services.

The security mechanisms shall be protected from unauthorized access.

The following security services shall be available to the component assembler:

1. Single login for users

   The single login for users means the user needs to identify himself once per session. It is the responsibility of the security services to protect and distribute the authentication information of a user.

2. Mutual authentication

   Mutual authentication ensures proper identification of the user to the system and the system to the user.

3. Auditing

   Auditing means significant security events are recorded for later analysis. Significant security events shall include logon and logoff, security policy changes, user and group management, and access to specified objects.

4. Secure key distribution

   Key distribution provides a secure transport mechanism for encryption keys.

5. Role based Access Control

   Role based access control assigns roles to users and privileges to roles, thereby simplifying access control if the number of roles is less than the number of users.

6. Data confidentiality

   Data confidentiality means data is disclosed according to a policy.

7. Data integrity

   Data integrity means the recipient gets the intended data.

8. Non-repudiation and authenticity

   Non-repudiation means the sender of a message can not later deny he sent the message.

NETWORK SECURITY

The trend in DoD is for networked systems vice standalone monolithic systems and because most systems have some level of classification, this architecture shall address network security.

The architectural framework shall support a secure network.

The architectural framework shall support the network security mechanisms specific to the target architecture, including firewalls, routers, encryption, and proxy services.

NETWORK COMMUNICATIONS

The architectural framework shall support different network protocols (i.e. TCP/IP) and topologies dependent on the target architecture.

The application layer shall be able to execute a variety of data management commands without having knowledge of the data location, database, file type, operating system, network protocol, or platform location.

DEVELOPMENT LANGUAGE

The architectural framework shall support any development language that is supported by the legacy system as well as any development language that supports platform independence for newly developed code in the target architecture.

2.2 ASSUMPTIONS AND DEPENDENCIES

Assumption 1: Legacy systems are monolithic and not modifiable.

Assumption 2: Legacy systems have some existing mechanism for interaction.

Assumption 3: There are varying degrees of COTS. To be considered COTS, the component cannot be modified.

Assumption 4: Reliability, performance, safety and security must be weighed in the target architecture.

Assumption 5: Multilevel security systems are beyond the scope of this effort.

3. TARGET ARCHITECTURE FUNCTIONS

DATABASE

COTS software applications which handle data tend to have their own mechanism and structure for the storage of the data internal to the COTS application. When the target architecture includes a master database to store its data, the architectural framework shall support the target architecture's central storage of data. The architecture shall support remote access to the database.

SECURITY

The target architecture shall support Discretionary Access Control (DAC).

Access to information controlled by an application shall be based on an access control list (ACL) of a parameter that can be used to distinguish between authorized and non-authorized entities. Entities include users, devices, and other applications.

The target architecture shall support non-repudiation.

a. The data recipient shall be assured of the originator's identify.

b. The data originator shall be provided with proof of delivery.

c. The algorithm used to digitally sign data entries and receipts shall be either the Digital Signature Standard (DSS) FIPS 186 or RSA (1024 bit).

d. The original transmitted data signed by the sender and the requested receipt signed by the recipient shall be time-stamped by a trusted third party.

GRAPHICAL USER INTERFACE (GUI)

The target architecture shall include a GUI style guide. If a GUI style guide does not exist for the target architecture, UNIX platforms shall adhere to the MOTIF standard and X-Windows standard, and PC platforms shall adhere to the Windows NT standard.

EXTERNAL SYSTEM INTERFACES

Because the target architecture exists in a network environment where it shares data with other external systems, the external system interfaces where information is exchanged shall be well defined to support interoperability.

MIDDLEWARE TECHNOLOGY

The COTS/GOTS/legacy architecture shall support new component integration technologies (i.e. COM/DCOM) to broker between components that by themselves normally do not communicate to form an integrated system.

The target architecture shall support wrappers to enable COTS/GOTS applications to interface with each other.
The wrappers shall support the METOC data (listed in Table 6 of reference 1) and its various formats within NITES. The architecture

shall ensure when an application updates a set of data, the update is consistently made throughout the rest of the database.

4. ARCHITECTURE ATTRIBUTES

4.1 PERFORMANCE REQUIREMENTS

The performance requirements for the target system are contained in Table 6B of the NITES Performance Specification. In addition to those performance requirements, the following requirements shall also be addressed in the target architecture.

The architecture shall optimize the database access over a network.

The architecture shall allow concurrent access of the database to multiple users.

The component technology shall not degrade the system performance by more than 10% of the target system's current performance requirements. Refer to Table 6B of the NITES Performance Specification.

4.2 RELIABILITY REQUIREMENTS

The target architecture shall use standard fault-tolerant technologies (i.e. Replication to maintain the reliability and availability requirements of DoD systems.)

While the data traverses throughout various applications, to different platforms, through the network and to/from database, it must remain consistent and not suffer any degradation.

4.3 DESIGN CONSTRAINTS

Because many existing legacy systems reside on UNIX platforms and the DoD has made a commitment to move towards a PC architecture, the architectural framework shall support both UNIX and PC platforms with the goal of moving towards a pure PC architecture. It is not required that all COTS/GOTS/legacy system components be executable on both platforms but the data must be able to be shared by components on different platforms.

Newly developed DoD systems must use COTS products to the greatest extent possible.

As most COTS/GOTS applications are designed to be standalone, these applications will usually have their own way of retrieving and storing data. When these applications are integrated into a system, the internals of the application of how it retrieves and stores data will not be modified.

There are varying degrees of COTS products. Depending on whether the COTS product is an opaque or a black box will drive the wrapper design and implementation.

APPENDIX G. SYSTEM DESIGN SPECIFICATION

1. SYSTEM ARCHITECTURE

1.1 SYSTEM ARCHITECTURE DIAGRAM

The Naval Integrated Tactical Environmental System (NITES) software runs in a distributed, heterogeneous environment on standard commercial-off-the-shelf (COTS) personal computers (PCs) and TAC-4 UNIX computers.

The NITES architecture consists of a central database residing on a UNIX computer, which is shared amongst the various NITES components (most of which reside on PCs with the exception of the tactical applications which reside on a TAC-4 UNIX computer) as depicted in figure 1. In this topology, there is no direct interaction between the components. All interactions are through the central database. This topology allows ease of integration of COTS components as it minimizes the integration effort since each component only has one interconnection.

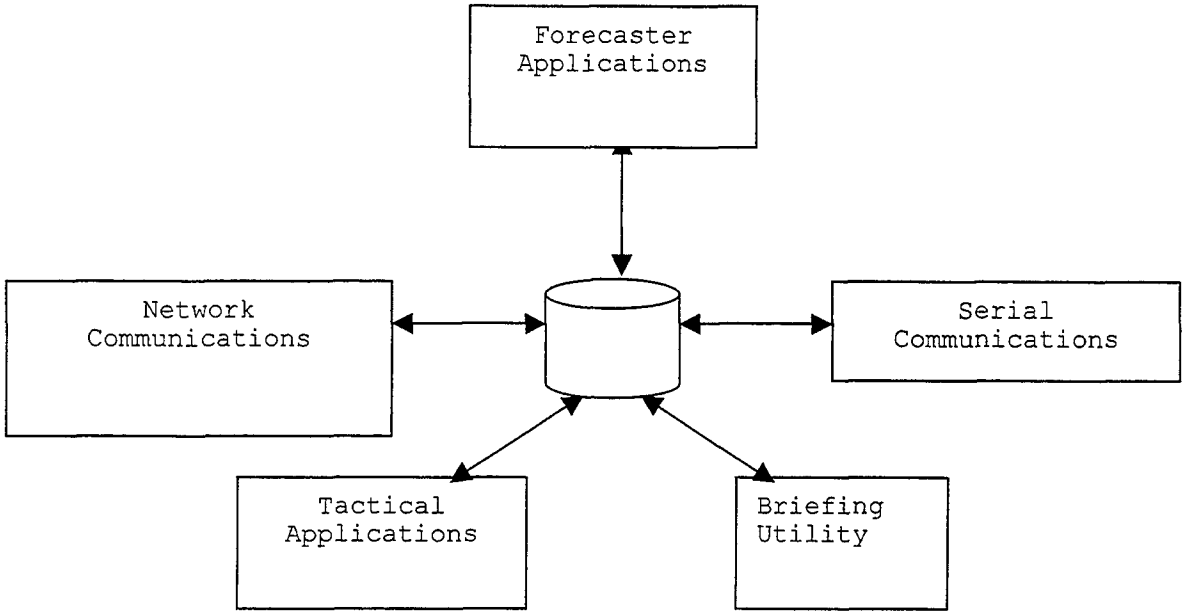

Figure 1 - NITES Architecture Diagram

Forecaster applications (COTS/GOTS) - Manipulate METOC data to easily plot, analyze, display on a common geographical reference.

Serial Communications (Legacy code) - Handles the ingest and dissemination of METOC data through existing legacy communication channels.

Briefing (COTS) - Briefing utility used to brief tactical commanders, flight operators the environmental conditions that they will be operating in.

Tactical applications (Legacy code and newly developed code) - Tactical applications take in METOC data to predict the affects of the environmental conditions on the environment, tactical equipment, etc.

Database (GOTS) - The database is the central repository for all METOC data.

Network communications (GOTS) - Handles the ingest and dissemination of METOC data through SIPRNET.

The deployment diagram, as depicted in figure 2, consists of a NITES Server, a NITES Database Server, and NITES workstations with a communications package, an applications package, a database package, a system controller package, a security package and a briefer package residing on multiple hardware platforms.

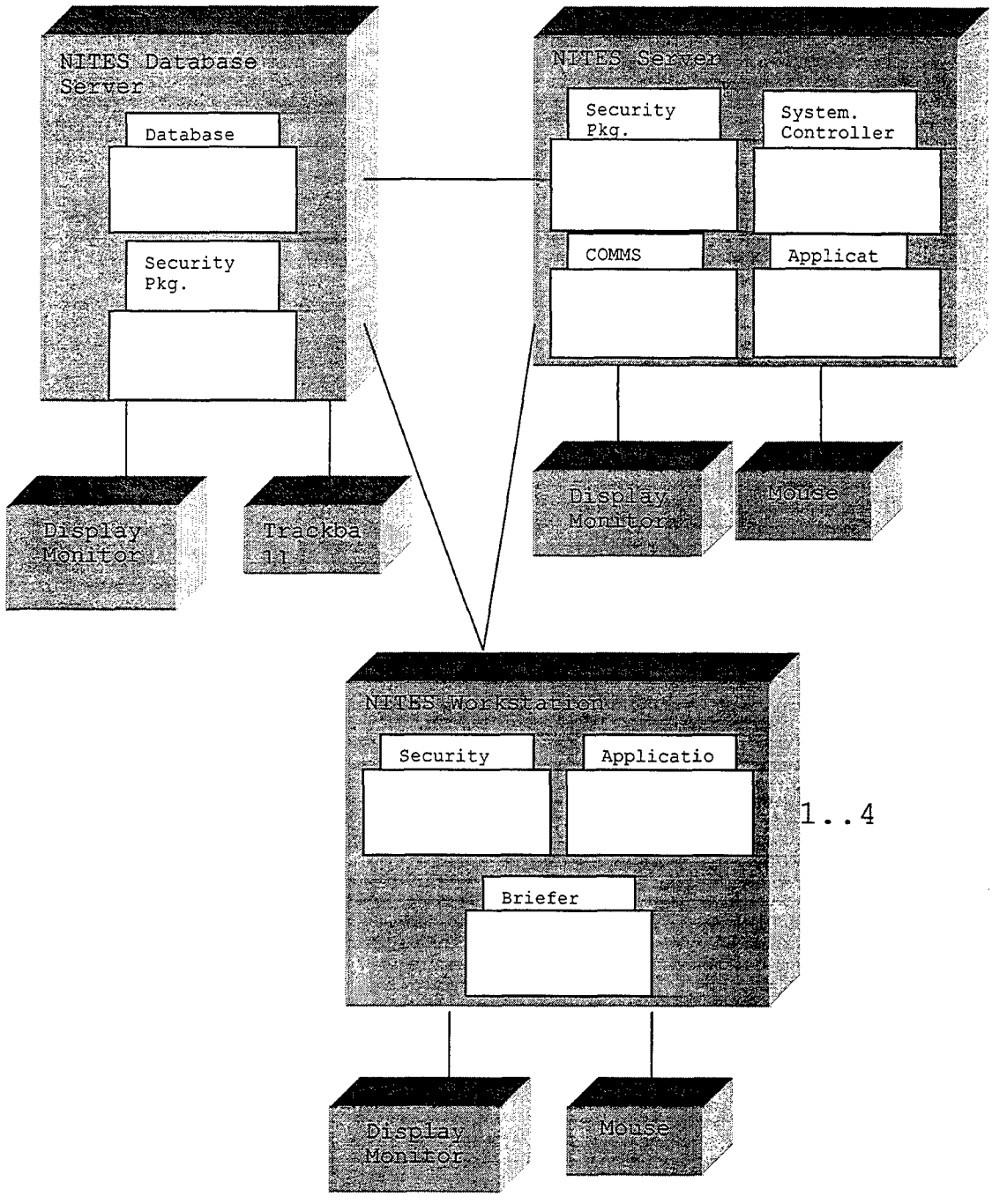

Figure 2 - Deployment Diagram

In the NITES architecture, all interactions are through the NITES database. However, in the initial delivery of the NITES software, this architecture was violated since none of the COTS applications were able to communicate with the NITES database to retrieve and/or store data and products.

A prototype of a portion of the NITES system will be developed to demonstrate the NITES architecture where a COTS application can communicate with the NITES database to retrieve and store data and products. A system controller package and the security package are newly developed for the NITES. The COTS applications packages and the briefer package will be modified to use wrapper and glue technology to enable it to communicate with the database package. These packages will be designed and developed to move the system in the direction of conforming to the existing architecture.

This prototype will use an object request broker (ORB) to marshal events/notifications in a distributed environment. Because this prototype is being developed under the Windows NT environment, and DCOM is freely available with Windows NT, we have chosen to use DCOM as our ORB.

DCOM components can communicate three ways: within the same process, out of process and between network nodes. The component internals do not need to be changed regardless of the deployment decision. The DCOMCNFG and dynamic link library (DLL) packaging are used to implement the deployment decision.

Deployment flexibility affords alternative performance solutions

in a distributed network environment. For example, the Monitor component could be deployed on a different network node than the Controller component to reduce CPU load. This solution assumes the sampling rate is higher than the notification rate.

1.2 INTER-TASK COMMUNICATION

The tasks on the NITES will be implemented to run asynchronously. Communications are broken down between the following tasks:

- Monitor/Controller
- Controller/Glue Component
- CBWrapper/Glue Component
- CBWrapper/Controller

The Application Wrapper is responsible for making the object available to a COTS viewer application.

MONITOR/CONTROLLER

Slides for the briefing package are generated by the operator using an external COTS/GOTS application. As each of these slides is generated, it is saved to a directory by the COTS/GOTS application. The system monitor polls the directory and when a file is found, notifies the controller.

CONTROLLER/GLUE COMPONENT

When the controller receives notification from the monitor that a new file exists, the controller will create an instance of the glue component.

CBWRAPPER/CONTROLLER

CBWrapper registers interest in new products with the controller.

When the controller is notified by the glue component that a file is successfully stored in the database, it will broadcast the information to all the wrappers running on client workstations. It is the responsibility of the CBWrapper to ignore image types not appropriate for the current brief. This assumes there is at least one wrapper running.

CBWRAPPER/GLUE COMPONENT

The CBWrapper requests an image product from the glue code, which will use the existing database APIs to connect to the database, retrieves the product and returns it to the CBWrapper. The request mechanism is used to initialize and update the brief.

2. SUBSYSTEM DESCRIPTION

The object diagram and sequence diagram depicts objects required to design the update of a briefing package and the scenario of updating a briefing package in figures 3 and 4 respectively.

MONITOR

The Monitor component is responsible for detecting the presence of a new object.

CONTROLLER

The controller component is responsible for coordinating multiple concurrent asynchronous activities. The controller runs on the application server. It serves two functions within the system, handling notifications from the monitor and the glue component.

GLUE COMPONENT

The glue component is responsible for storing and retrieving objects from an ODBC compliant relational database.

CBWRAPPER

Wrappers are software code developed to add, modify, and hide functionality from COTS, GOTS or legacy software components to align them with the overall system requirements and architecture. In the design, wrapper and glue code technology is being implemented to enable the COTS applications to adhere to the existing NITES architecture.

The briefing package consists of Microsoft PowerPoint, a COTS application package. The PowerPoint application contains APIs, which can be used by CBWrapper to create the added functionality of automatically creating and updating the briefing package in the background.

The PPT APIs used for the wrapper interface include:

- Presentations.Add
- Slides.Add
- SlideShowTransition
- SlideShowSetting
- Shapes.AddPicture
- Shapes.PictureFormat

INITIALIZATION GUI

The Initialization GUI is used to initialize each component with the number of images, starting from the most current; the image type; the display duration of each image in seconds; and the height and width of the display area. Default values are 24 images, 0 second duration, and display area equal to the workstation's screen size.

CONFIGURATION GUI

The Configuration GUI defines the set of image types available for the brief. Associated with each image type is the working directory containing the current set of brief images and a web server virtual directory corresponding to the working directory. The CBWrapper uses the configuration file to initialize the image type

options available to the briefer. The monitor uses the configuration file to build a list of directories to poll.

The Configuration GUI is not restricted to the image types settings. It can be used for defining various sets of key values. For instance, we can use this Configuration GUI to define the key set values for network configuration, or application's initial default settings. This provides the extensibility for future development of applications.

NAMING CONVENTION

The filename associated with each image type consists of the fields represented the created date and time, the file format (i.e., gif, jpeg, etc.), and other information for a particular image (i.e., the channel, the location, etc.)

The filename begins with the date and time, followed by other information. For instance, a file named "20000523.1331.gms5.IR.MODEL_OVERLAY.500HT.NOGAPS" indicates that the file was created on May 23, 2000, at 13:31. The CBWrapper uses the date and time embedded in the filename for updating the continuous brief.

The other information of the filename is used by the Glue component for storing and retrieving images to and from the database.

THIN CLIENT TECHNOLOGY

CBWrapper is implemented using modern thin client technology. When a user opens a HTTP page from a browser, the CBWrapper is then automatically downloaded and installed on the client machine. Once the CBWrapper is up and running, all images needed for creating the brief are dynamically downloaded from the server using the OpenURL method. OpenURL uses the current open HTTP connection to transfer image files. The continuous brief is created on the client machine using the PowerPoint APIs. The PowerPoint is used to display the brief.

PUSH TECHNOLOGY

The advantage of using this technique is that the client needs not to poll the server periodically for new data. The server notifies its clients (CBWrapper) when new data (images) arrive. The CBWrapper receives the notification and compares the image type with the type being showed. If the image types match, the CBWrapper downloads a new set of images from the server and updates the brief.

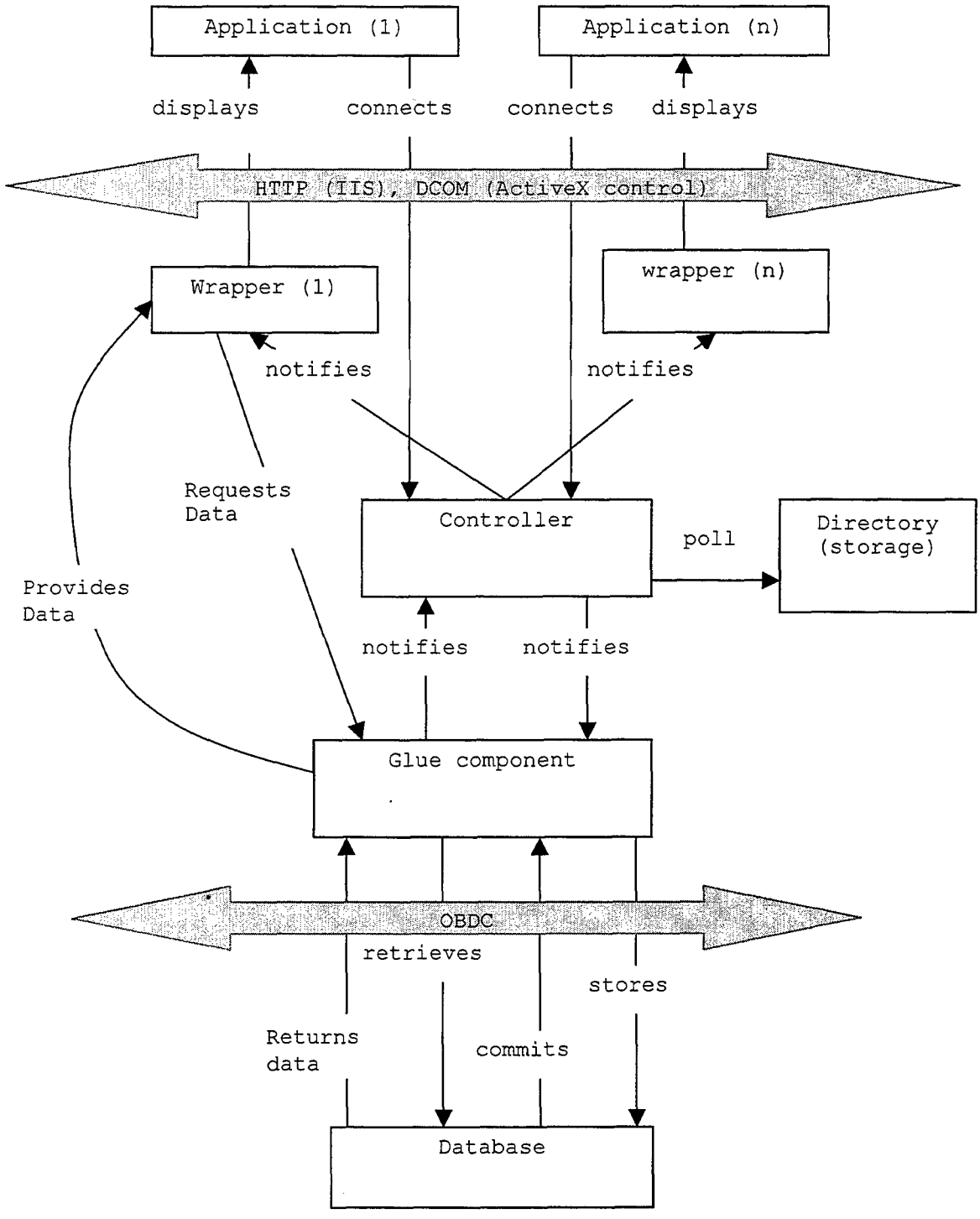

Figure 3 - Wrapper & Glue Code Object Diagram

OMF

Sharing different formatted data requires a common representation of data to interpret, send, and receive any data, any format, anywhere. Within NITES, meteorological and oceanographic observations, and certain types of bulletins (SIGMETS, JOTS warnings, and Tropical Cyclone Warnings, for example) are received and transmitted in an Extensible Markup Language (XML)-based format called Weather Observation Markup Format (OMF). OMF preserves the original text of each observation or bulletin, and also includes information decoded from the observation/bulletin and other metadata concerning the message.

OMF solves the data interoperability problem by providing self-describing tags along with the data so that the receiving applications can consistently interpret the data correctly. These self-describing tags are detailed in the Document Type Definition (DTD). When drafting the NITES data into OMF, three things must be agreed on: which tags will be allowed, how tagged elements may nest within one another and how they should be processed. The first two, the language's vocabulary and structure, are codified in the DTD.

OMF is an application of XML, and by its virtue, an application of SGML. SGML is used extensively within DoD for documenting of various types of information (military standards, procurement materials, service manuals). OMF brings weather observations into the same fold. Thus, the design goals of OMF are:

- Mark up (annotate) raw observation reports with additional description and derived, computed quantities.
- The raw report data must not be modified in any way, and should be conveniently extractable (by simply stripping all the tags away).
- OMF must be concise. While providing useful annotations to a client, OMF markup should not impose undue overhead on communication channels.
- It should be possible to extend the markup with additional annotations, without affecting applications that do not use this information.

The OMF contains the following elements:

- **Reports** - defines a group of weather observation reports
- **METAR** for a single METAR report
- **SPECI** for a single SPECI report
- **UAR** for a combined Rawinsonde and Pibal Observation report
- **BTSC** for ocean profile data (temperature, salinity, current)
- **SYN** for a surface synoptic report from a land or sea station
- **Advisories** - defines a collection of weather hazard warnings
- **SIGMET** - SIGnificant METeorological Information

- **Forecasts** - defines a set of weather forecasts
- **TAF** - Terminal Aerodrome Forecasts
- **Messages** - defines a set of plain-text bulletins.

The following sections define the major elements along with the minor elements that are relevant to them. In each section, XML DTD declarations are provided for precise definition of elements and attributes. The collection of XML DTD declarations found in this specification can be arbitrarily extended to add new elements and attributes for new enhancements. Some of the element attributes are common. For compactness, they are defined in the following table.

Table 1-1. Basic Attributes of an Observation in OMF

Attribute	Brief Description	Format	Description
TStamp	Time Stamp	unsigned integer	UTC time in seconds since the Epoch, 00:00:00 Jan 1, 1970 UTC. This is the value returned by a POSIX function time(2). Example: Tstamp='937507702'
TRange	Time Interval	a string of form "aaa, bbb", where aaa and bbb are unsigned integer numbers specifying the beginning and the end timestamps of the interval.	Timestamps are in seconds since the Epoch, 00:00:00 Jan 1, 1970 UTC. These are the values returned by a POSIX function time(2). Example: Trange='937832400, 937915200'
LatLon	Specification of a Point on the globe	A string of a form "aaa.bbb, ccc.ddd", where aaa.bbb and ccc.ddd are signed floating point numbers	The latitude and. longitude, respectively, of a point on the globe, in whole and fractional degrees. The numbers are positive for Northern latitudes and Eastern longitudes, and negative for Southern latitudes and Western longitudes. The range of the numbers is [-90.0, 90.0] for latitudes, (-180.0, 180.0] for longitudes. Example: LatLon='32.433, -99.850'

LatLons	Specification of a Sequence of Points on the Globe	a string of a form "lat1, lon1, lat2, lon2, latn, lonn" where each pair (lat1, lon1, etc.) are signed floating point numbers	A sequence of pairs of numbers, each pair giving the latitude and longitude of a single point in the sequence, in whole and fractional degrees. See the LatLon attribute above for more details. Example: LatLons='38.420, -111.125, 36.286, -111.492, 36.307, -112.630, 37.700, -113.223, 38.420, -111.125'

Table 1-1. Basic Attributes of an Observation in OMF

Attribute	Brief Description	Format	Description
BBox	Bounding box, which tells the latitudal and the longitudal spans of an area of the globe	A string of a form "lat-N, lon-W, lat-S, lon-E", where the lats and lons are signed floating-point numbers, in degrees	Specification of the bounding box for an area of interest. Here lat-N is the latitude of the Northern-most point of the area, lat-S is the latitude of the Southern-most point, lon-W is the longitude of the Western-most point of the area, and lon-E is the Eastern-most longitude. It is required that lat-N >= lat-S. The left-lon (lon-W) may however be greater than the right-lon (lon-E). For example, a range of longitudes [-170,170] specifies the entire world but Indonesia. On the other end, the range [170, -170] includes Indonesia only. By the same token, [-10,10] pertains to a 21-degree longitude strip along the Greenwich meridian, while [10,-10] specifies the whole globe except for that strip. Example: Bbox='60.0, -120.0, 20.0, -100.0'
BId	Station identificatio n group	Unsigned integer	WMO Block Station ID, or other identifier for buoy or ship
SName	Call sign and	A string of	The observing stations

	full name of an observing station	the form "ccccc, name", where ccccc are the call letters of the station (ICAO station id: 4 or 5 upper-case letters, may be omitted), name is an arbitrary string describing the station	ICAO, aircraft, or ship call sign, plus a plain-text station name (e.g. "KMRY, Monterey CA Airport" Example: Sname='KYNL, YUMA (MCAS)'
Elev	Elevation	A non-negative integer, or omitted if unknown.	Station elevation relative to sea level, in meters. This attribute may specify a surface elevation of an observation station, or an upper-air elevation for an upper-air report. Example: Elev='16'

Table 1-2. OMF Attributes for METAR and SPECI Reports

Attribute	Brief Description	Format	Description	Req'd?
TStamp	Time Stamp	<----------See Table 1-1-------->		Yes
LatLon	Station latitude and longitude	<----------See Table 1-1-------->		Yes
BId	Station Identification Group	Unsigned integer	WMO Block Station ID	Yes
SName	Call sign and full name of an observing station	<----------See Table 1-1-------->		Yes
Elev	Station elevation	<----------See Table 1-1-------->		No
Vis	Visibility	a number of meters, omitted, or a special token "INF"	Horizontal visibility in meters	No
Ceiling	Ceiling	a number of feet, omitted, or a special token "INF"	Ceiling in feet	No

Table 1-3. OMF Attributes for the SYN Element

Attribute	Brief Description	Format	Description	Req'd?
TStamp	Time Stamp	<----------See Table 1-1---------->		Yes
LatLon	Station latitude and longitude	<----------See Table 1-1---------->		Yes
BId	WMO Block Station Number	String	For a buoy or other observation platform, this id is a combination of a WMO region number, subarea number (per WMO Code Table 0161), and the buoy type and serial number. This information is reported in Section 0 of a synoptic report. If Section 0 contains a call sign rather than a numerical id (as typical with FM 13 SHIP reports), the BId attribute is computed as itoa(1000009 + hc) % 2^30, where hc is a numerical representation of the call letters considered as a number in radix 36 notation. For example, "0000" hashes to 0, and "ZZZZ" hashes to 1,679,615. Note this formula makes the Bid attribute a unique numeric identifier for the station.	Yes

SName	Call sign and full name of an observing station	<----------See Table 1-1---------- --->		Yes
Elev	Station elevation	<----------See Table 1-1---------- --->		No
Title	Report title	String	Title defining type of report: AAXX (FM-12), BBXX (FM-13), or ZZYY (FM-18)	Yes
Stype	Station type	String	Type of station: automated (AUTO) or manned (MANN); defaults to MANN	No

Table 1-4. OMF Attributes for the SYG Element

Attribute	Brief Description	Format	Description	Req'd?
T	Air Temperature	positive, zero, or negative number	Air temperature in degrees Celsius	No
TD	Dew point Temperature	positive, zero, or negative number	Dew point temperature in degrees Celsius	No
Hum	Relative humidity	non-negative number	Relative humidity in per cent	No
Tmm	Extreme temperatures over the last 24 hours	a string of a form "mmmm, MMMM" or omitted	Minimum and maximum temperatures (degrees Celsius) over the last 24 hours	No
P	Station pressure	positive number	Atmospheric pressure at station level, in hectoPascals	No
P0	Sea level pressure	positive number	Atmospheric pressure at station, reduced to sea level, in hPa	No
Pd	Pressure Tendency	String of form "dddd", or omitted	Pressure tendency during the 3 hours preceding the observation	No
Vis	Visibility Number of meters, omitted, or a special token "INF"	Horizontal visibility in meters	Horizontal visibility in meters	No
Ceiling	Ceiling	Number of feet, omitted, or a special token "INF"	Ceiling in feet	No
Wind	Wind speed and direction	String of form "nnn, mm" or omitted	nnn is a true direction from which the wind is blowing, in degrees, or VAR if " the wind is variable, or all directions or unknown or waves confused, direction indeterminate." This is an integer number within [0,360), with 0 meaning the wind is	No

			blowing from true North, 270 stands for the wind blowing from due West. Normally this number has a precision of 10 degrees. mm is the wind speed in meters per second.	

Table 1-4. OMF Attributes for the SYG Element (Cont.)

Attribute	Brief Description	Format	Description	Req'd?
Wx	Past and present Weather conditions and phenomena	String of four digits, "NOSIG", or omitted	See WMO-306, Code tables 4677 and 4561 for the meaning of the four digits. This attribute is coded as "NOSIG" if there is no significant phenomenon to report. The attribute is omitted if not observed or data is not available (see ix indicator, Code table 1860).	No
Prec	Precipitation amount	String of form "nnn, hh" or "" or omitted	nnn is the amount of precipitation which has fallen during the period preceding the time of observation. The precipitation amount is a non-negative decimal number, in mm. hh is the duration of the period in which the reported precipitation occurred, in whole hours. This attribute is encoded as "" if no precipitation was observed. The attribute	No

			is omitted if unknown or not available (see iR indicator, Code table 1819). Sea stations typically never report precipitation.	
Clouds	Amounts and types of cloud cover	String of five symbols "tplmh" or omitted	The first digit is the total cloud cover in octas (Code table 2700). The second digit is the cloud cover of the lowest clouds, in octas. The other three symbols are types of low, middle, and high clouds, resp. See WMO-306 Code tables for more details.	No
T	Sea surface temperature	Positive, zero, or negative number	Sea surface temperature in degrees Celsius	No
Wave	Sea wave period and height	String of form "pp, hh" or omitted	pp is the period of wind waves in seconds. hh is the height of wind waves, in meters. If a report carries both estimated and measured wind wave data, the instrumented information is preferred.	No

Table 1-4. OMF Attributes for the SYG Element (Cont.)

Attribute	Brief Description	Format	Description	Req'd?
SDir	Ship's course and speed	String of form "nnn, mm" or omitted.	nnn is a true direction of resultant displacement of the ship during the three	No

			hours preceding the time of observation. The number is in degrees, or VAR if "variable, or all directions or unknown or waves confused, direction indeterminate." This is an integer number within [0,360), with 0 meaning the ship has moved towards the true North; 270 means the ship has moved to the West. Normally this number has a precision of 45 degrees. mm is the average speed made good during the three hours preceding the time of observation, in meters per second.	

Table 1-6. OMF Attributes for the UALEVEL Element

Attribute	Brief Description	Format	Description	Req'd?
Ref	Reference to sounding Part	String - "TTAA", "TTBB", etc.	Reference to the part of the sounding from which the level data were derived	Yes
P	Pressure	positive number	Atmospheric pressure at sounding level, in hectoPascals	Yes
H	Geopotential height	Non-negative number of geopotential meters, or 'SURF' for surface, 'TROP' for tropopause, 'MAXW' for level of maximum winds, 'MAXWTOP' for maximum wind level at the top of the sounding, or omitted	Geopotential height of the reported level, or a special height indicator	No
T	Air Temperature	positive, zero, or negative number	Air temperature in degrees Celsius at the reported level	No
DP	Dew point temperature	positive, zero, or negative number	Dew point temperature in degrees Celsius at the reported level	No
Wind	Wind speed and direction	String of form "nnn, mm" or "nnn, mm bbb" or "nnn, mm ,aaa" or "nnn, mm bbb, aaa" or omitted	nnn is a true direction **from** which the wind is blowing, in degrees, or VAR if " the wind is variable, or all directions or unknown or waves confused, direction	No

			indeterminate." This is an integer number within [0,360), with 0 meaning the wind is blowing from true North, 270 stands for the wind blowing from due West. Normally this number has a precision of 10 degrees. mm is the wind speed in meters per second. If specified, bbb stands for the absolute value of the vector difference between the wind at a given level, and the wind 1 km below that level, in meters per second. The number aaa if given is the absolute value of the vector difference between the wind at a given level, and the wind 1 km above that level, in meters per second.	

Table 1-7. OMF Attributes for the BTSC Element

Attribute	Brief Description	Format	Description	Req'd?
TStamp	Time Stamp	<---------- See Table 1-1 --------->		Yes
LatLon	Latitude and Longitude of observation	<---------- See Table 1-1 --------->		Yes
BId	Station identifier group	positive integer	For a buoy or other observation platform, this ID is a combination of a WMO region number, subarea number (per WMO-306 Code Table 0161), and the buoy type and serial number. This information is reported in Section 4 of a BTSC report. If Section 4 contains a call sign rather than a numerical id, the BId attribute is computed as itoa(1000009 + hc), where hc is a numerical representation of the call letters considered as a number in radix 36 notation. For example, "0000" hashes to 0, and "ZZZZ" hashes to 1,679,615. Note this formula makes the Bid attribute a unique numeric identifier for the station.	Yes
SName	Call sign	string	Ship's call sign, if reported	Yes
Title	Report type	string	"JJYY" - FM 63 X Ext. BATHY report "KKXX" - FM 64 IX TESAC report "NNXX" - FM 62 TRACKOB report	Yes
Depth	Water depth	positive	Total water depth at point	No

		number	of observation	

Table 1-8. OMF Attributes for the BTID Element

Attribute	Brief Description	Format	Description	Req'd?
DZ	Indicator for digitization	"7" or "8" or omitted	Indicator for method of digitization used in the report (k_1 field). See WMO-306 Code Table 2262. Required for BATHY and TESAC reports	No
Rec	Instrument type code	5-digit code	Code for expendable bathythermograph (XBT) instrument type and fall rate (WMO-306 Code Table 1770)	No
WS	Wind speed units code	"0", "1", "2", "3", or omitted	Indicator for units of wind speed and type of instrumentation (i_u field). See WMO-306, Code Table 1853.	No
Curr-s	Method of current speed measurement	"2", "3", "4", or omitted	Indicator for the method of current measurement (k_5 field). See WMO-306 Code Table 2266.	No
Curr-d	Indicators for the method of subsurface Current measurement	3-digit numerical code	Indicators for the method of subsurface current measurement ($k_6k_4k_3$ codes). See WMO-306, Code Tables 2267, 2265, and 2264.	No
AV-T	Averaging period for sea temperature	"0", "1", "2", "3", or omitted (if no sea temperature data are reported)	Code for the averaging period for sea temperature (m_T code). See WMO-306, Code Table 2604	No
AV-SAL	Averaging period for salinity.	"0", "1", "2", "3", or omitted (if no salinity data are reported)	Code for the averaging period for sea salinity (m_S code). See WMO-306, Code Table 2604	No

AB-Curr	Averaging period for surface Current direction and speed	"0", "1", "2", "3", or omitted (if no current data are reported)	Code for the averaging period for surface current direction and speed ($_{mc}$ code). See WMO-306, Code Table 2604	No
Sal	Method of salinity/depth measurement	"1", "2", "3", or omitted (if no salinity data are reported)	Code for the method of salinity/depth measurement ($_{k2}$ code). See WMO- 306, Code Table 2263.	No

Table 1-9. OMF Attributes for the BTAIR Element

Attribute	Brief Description	Format	Description	Req'd?
T	Air temperature	Positive, zero, or negative number, or omitted	Air temperature just above the sea surface, in degrees Celsius.	No
Wind	Wind vector	String of form "nnn,mm", or omitted	Here nnn is a true direction from which the wind is blowing, in degrees, or VAR if " the wind is variable, or all directions or unknown or waves confused, direction indeterminate." This is an integer number within [0,360), with 0 meaning the wind is blowing from the true North;, 270 means the wind is blowing from the West. Normally this number has a precision of 10 degrees. mm is the wind speed in meters per second.	No

Table 1-10. OMF Attributes for the BTLEVEL Element

Attribute	Brief Description	Format	Description	Req'd?
D	Depth	Non-negative number	Depth of the level in meters.	Yes
T	Water temperature	Positive, zero, or negative number, or omitted	Water temperature at the reported level.	No
S	Salinity	Positive number, or omitted	Salinity at the reported level, in parts per thousand.	No
C	Current vector String of form	"nnn,mm", or omitted	nnn is the true direction toward which the sea current is moving, in degrees, or VAR if "the current is variable, or	No

			all directions or unknown, direction indeterminate." This is an integer number within [0,360), with 0 meaning the current flows toward true North; 270 means the current is flowing toward the West. Normally this number has a precision of 10 degrees. mm is the speed of current in meters per second.	

Table 1-11. OMF Attributes for the TAF Element

Attribute	Brief Description	Format	Description	Req'd?
TStamp	Time Stamp	<---------- See Table 1-1 --------->		Yes
LatLon	Latitude and Longitude of observation	<---------- See Table 1-1 --------->		Yes
BId	Block Station ID	positive integer	WMO Block Station ID of the reporting station	Yes
SName	Call sign	string	Ship's call sign, if reported	Yes

Table 1-12. OMF Attributes for the SIGMET Element

Attribute	Brief Description	Format	Description	Req'd?
class	SIGMET type	"CONVECTIVE", "HOTEL", "INDIA", "UNIFORM", "VICTOR", "WHISKEY"	Identifier for the type of SIGMET message	Yes
id	Identifier for a particular advisory	String	Identifier for the advisory; value depends on the advisory class.	Yes
TStamp	Time Stamp	<---------- See Table 1-1 --------->		Yes
BBox	Bounding box for advisory area	<---------- See Table 1-1 --------->		Yes

Table 1-13. OMF Attributes for the EXTENT Element

Attribute	Brief Description	Format	Description	Req'd ?
Shape	Type of area specification	"AREA", "LINE", "POINT"	Type of area shape specified	Yes
LatLons	List of latitudes and Longitudes defining the area	Positive, zero, or Negative numbers in lat/lon pairs	Control points (vertices) for a polygon/polyline representing the affected area	Yes

Table 1-14. OMF Attributes for the MSG Element

Attribute	Brief Description	Format	Description	Req'd?
id	Message identifier	A NMTOKEN, a four-to-six-character string of a form T1T2A1A2ii	Designator for the message type and subtype (T1T2), area (A1A2), and sequence code (ii) of the message, as described in WMO- 386.	Yes
Type	Message type	2-letter string (T1T2)	Designator for the message type and subtype (T1T2) as specified in WMO-386, Tables A and B1 through B6	Yes
TStamp	Time Stamp	<---------- See Table 1-1 --------->		Yes
SName	Originating station name	String	String containing the identification of the station that originated the message (normally its ICAO call sign)	Yes
BBB	Annotation group	3-character string	So-called "BBB groups" from the abbreviated message line. They indicate that the message has been delayed, corrected or amended. A BBB group can also be used for segmentation. See the WMO-386 for more detail.	No
Descr	Description	String	Keywords and other information describing the message.	No
BBox	Bounding box	<---------- See Table 1-1 --------->		No

Table 1-15 Layer Parameter Codes

layer	Description	Example
adiabatic-cond	Adiabatic condensation level (parcel lifted from surface)	(layer adiabatic-cond)
atm-top	Level of the top of the atmosphere	(layer atm-top)
cloud-base	Cloud base level	(layer cloud-base)
cloud-top	Cloud top level	(layer cloud-top)
conv-cld-base	Level of bases of convective clouds	(layer conv-cld-base)
conv-cld-top	Level of tops of convective clouds	(layer conv-cld-top)
entire-atm	Entire atmosphere	(layer entire-atm)
entire-ocean	Entire ocean	(layer entire-ocean)
height	Height above ground (meters)	(layer height 1500)
height-between	Layer between two heights above ground in hundreds meters (followed by top and bottom level values)	(layer height-between 50 30) for layer between 5000 and 3000 meters above ground
height-between-ft	Layer between two heights above ground, in feet (followed by top and bottom level values)	(layer height-between-ft 15000 10000)
height-ft	Height above ground (feet)	(layer height-ft 50)
high-cld-base	Level of high cloud bases	(layer high-cld-base)
high-cld-top	Level of high cloud tops	(layer high-cld-top)
hybrid	Hybrid level (followed by level number)	(layer hybrid 1)
hybrid-between	Layer between two hybrid levels (followed by top and bottom level numbers)	(layer hybrid 2 1)
isobar	Level of an isobaric surface (followed by the isobar value of the surface in hectoPascals (hPa) (1000, 975, 950, 925,900,850,800,750,700,65	(layer isobar 500)

	0,600,550,500,450,400,350,3 00,250,200, 150,100, 70, 50, 30, 20,10)	
isobar-between	Layer between two isobaric surfaces (followed by top and bottom isobar values in kPa, separated by a space)	(layer isobar-between 50 100) for layer between 500 and 1000 hPa
isobar-between-mp	Layer between two isobaric surfaces, mixed precision (followed by pressure of top in kPa and 1100 minus pressure of bottom in hPa)	(layer isobar-between-mp 50 100) for layer between 500 and 1000 hPa

Table 1-15 Layer Parameter Codes (Cont.)

Layer	Description	Example
isobar-between-xp	Layer between two isobaric surfaces, extra precision (followed by top and bottom isobar values expressed as 1100 hPa-isobar level, separated by a space)	(layer isobar-between 600 100) for layer between 500 and 1000 hPa
isotherm-0	Level of the zero-degree (Celsius) isotherm (or freezing level)	(layer isotherm-0)
land-depth	Depth below land surface in centimeters	(layer land-depth 5.0)
land-depth-between	Layer between two depths in ground (followed by the depth of the top of the layer and the depth of the bottom of the layer centimeters)	(layer land-depth-between 0 30) for layer from ground surface to 30 cm depth
land-height-cm	Height level above ground (high precision) (followed by height in centimeters)	(layer land-height-cm 50)
land-isobar	Pressure above ground level in hPa	(layer land-isobar 500)
land-isobar-between	Layer between two isobars abive levels (followed by top and bottom isobaric levels in hPa)	(layer land-isobar-between 500 1000)
low-cld-base	Level of low cloud bases	(layer low-cld-base)
low-cld-top	Level of low cloud tops	(layer low-cld-top)
max-wind	Level of maximum wind	(layer max-wind)
mid-cld-base	Level of middle cloud bases	(layer mid-cld-base)
mid-cld-top	Level of middle cloud tops	(layer mid-cld-top)
msl	Mean sea level	(layer msl)
msl-height	Height above mean sea level (in meters)	(layer msl-height 50)
msl-height-between	Layer between two heights above mean sea level in hundreds of meters (followed by top and bottom height values)	(layer msl-height-between 10 5) for layer between 1000 and 500 meters above ground
msl-height-ft	Height above mean sea level	(layer msl-height-ft 5000)

	(in feet)	
sea-bottom	Bottom of the ocean	(layer sea-bottom)
sea-depth	Depth below the sea surface (meters)	(layer sea-depth 50)
sigma	Sigma level in 1/10000	(layer sigma 9950) for sigma level .995
sigma-between	Layer between two sigma surfaces (followed by top and bottom sigma values expressed in 1/100, separated by a space)	(layer sigma-between 99.5 100.0) for layer between .995 and 1.0

Table 1-15 Layer Parameter Codes (Cont.)

Layer	Description	Example
sigma-between-xp	Layer between two sigma levels (followed by top and bottom sigma values expressed as 1.1-sigma)	(layer sigma-between-xp .105 .100) for layer between .995 and 1.0
surface	Earth's surface	(layer surface)
theta	Isentropic (theta) level (followed by potential temperature in degrees K)	(layer theta 300)
theta-between	Layer between two isentropic surfaces (followed by top and bottom values expressed as 475-theta in degrees K)	(layer theta-between 150 200)
tropopause	Level of tropopause (top of troposphere)	(layer tropopause)

PowerPoint API Function Description Table

Method	Description	Example
Application	Represents the entire Microsoft PowerPoint application.	MyPath = **Application**.Path
ActivePresentation	Returns a **Presentation** object that represents the presentation open in the active window. (Read-only)	Application .**ActivePresentation**.SaveAs MyPath
Presentations	Returns a **Presentation** object that represents the presentation in which the specified document window or slide show window was created. (Read-only)	firstPresSlides = Windows(1).Presentation.Slides.Count Windows(2).**Presentation**.PageSetup _ .FirstSlideNumber = firstPresSlides + 1
Presentations.Add	Creates a presentation. Returns a **Presentation** object that represents the new presentation.	This example creates a presentation, adds a slide to it, and then saves the presentation. With Presentations.**Add** .Slides.Add 1, ppLayoutTitle .SaveAs "Sample" End With
Slides	A collection of all the **Slide** objects in the specified presentation.	Use the **Slides** property to return a **Slides** collection: ActivePresentation.Slides.Add 2, ppLayoutBlank
Slides.Add	Creates a new slide and adds it to the collection of slides in the specified presentation. Returns a **Slide** object that represents the new slide.	This example adds a blank slide at the end of the active presentation. With ActivePresentation.Slides .**Add** .Count + 1, ppLayoutBlank End With

Shapes	A collection of all the **Shape** objects on the specified slide. Each **Shape** object represents an object in the drawing layer, such as an AutoShape, freeform, OLE object, or picture.	Use the **Shapes** property to return the **Shapes** collection. The following example selects all the shapes on myDocument. Set myDocument = ActivePresentation.Slides(1) myDocument.Shapes.SelectAll
Shapes.AddPicture	Creates a picture from an existing file. Returns a **Shape** object that represents the new picture.	Set myDocument = ActivePresentation.Slides(1) myDocument.Shapes.**AddPicture** "c:\microsoft office\" & _ "clipart\music.bmp", True, True, 100, 100, 70, 70
Shapes.PictureFormat	Contains properties and methods that apply to pictures and OLE objects. The **LinkFormat** object contains properties and methods that apply to linked OLE objects only. The **OLEFormat** object contains properties and methods that apply to OLE objects whether or not they're linked.	Set myDocument = ActivePresentation.Slides(1) With myDocument.Shapes(1).PictureFormat .Brightness = 0.3 .Contrast = 0.7 .ColorType = msoPictureGrayScale .CropBottom = 18 End With
SlideShowTransition	Contains information about how the specified slide advances during a slide show.	With ActivePresentation.Slides(1).SlideShowTransition .Speed = ppTransitionSpeedFast End With
SlideShowSetting	Represents the slide show setup for a presentation.	With ActivePresentation.SlideShowSettings .RangeType = ppShowSlideRange End With

APPENDIX H. VISUAL BASIC IMPLEMENTATION

1. Configuration GUI (CBcfg)

```
VERSION 5.00
Begin VB.Form CBform
   BackColor       =   &H80000004&
   Caption         =   "CBcfg"
   ClientHeight    =   9195
   ClientLeft      =   60
   ClientTop       =   345
   ClientWidth     =   8490
   LinkTopic       =   "Form1"
   ScaleHeight     =   9195
   ScaleWidth      =   8490
   StartUpPosition =   3  'Windows Default
   Begin VB.TextBox VirtualDirText
      Height          =   375
      Left            =   1080
      TabIndex        =   3
      Tag             =   "3"
      Top             =   7320
      Width           =   6375
   End
   Begin VB.TextBox TypeText
      Height          =   375
      Left            =   1080
      TabIndex        =   1
      Top             =   5160
      Width           =   6375
   End
   Begin VB.CommandButton Delete
      Caption         =   "Delete"
      Enabled         =   0   'False
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   9.75
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   375
      Left            =   4440
      TabIndex        =   6
      Top             =   8160
      Width           =   1335
```

```
End
Begin VB.CommandButton Add
   Caption         =   "Set"
   Enabled         =   0   'False
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   9.75
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   375
   Left            =   6120
   TabIndex        =   7
   Top             =   8160
   Width           =   1335
End
Begin VB.CommandButton Cancel
   Caption         =   "Cancel"
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   9.75
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   375
   Left            =   2760
   TabIndex        =   5
   Top             =   8160
   Width           =   1335
End
Begin VB.CommandButton OK
   Caption         =   "OK"
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   9.75
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   375
   Left            =   1080
```

```
      TabIndex        =   4
      Top             =   8160
      Width           =   1335
   End
   Begin VB.TextBox LocationText
      Height          =   375
      Left            =   1080
      TabIndex        =   2
      Tag             =   "3"
      Top             =   6240
      Width           =   6375
   End
   Begin VB.ListBox dataList
      Height          =   3570
      Left            =   1080
      TabIndex        =   0
      Top             =   720
      Width           =   6375
   End
   Begin VB.Label Label2
      Caption         =   "Virtual directory (optional):"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   9.75
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   255
      Left            =   1080
      TabIndex        =   11
      ToolTipText     =   "A virtual directory associated
with the key used by the Web server."
      Top             =   6840
      Width           =   2775
   End
   Begin VB.Label Label4
      Caption         =   "Key:"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   9.75
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
```

```
      Height          =   255
      Left            =   1080
      TabIndex        =   10
      ToolTipText     =   "An image type or any other
variable name."
      Top             =   4680
      Width           =   615
   End
   Begin VB.Label Label3
      Caption         =   "Directory:"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   9.75
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   255
      Left            =   1080
      TabIndex        =   9
      ToolTipText     =   "An actual directory associated
with the key."
      Top             =   5760
      Width           =   1095
   End
   Begin VB.Label Label1
      Caption         =   "Current configuration:"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   9.75
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   255
      Left            =   1080
      TabIndex        =   8
      ToolTipText     =   "The current setting for
Continuous Brief application."
      Top             =   240
      Width           =   2295
   End
End
Attribute VB_Name = "CBform"
Attribute VB_GlobalNameSpace = False
```

```
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
'#############################################################
'#
'#  File: CBform.frm
'#  Date                Author              Histor
'#  5/31/2000           Tam Tran            Created.
'#
'#  CBcfg is an utility application that provides a
'#  Graphical User Interface (GUI) for setting the image
'#  type and its location. This application supports the
'#  configuration of CBWrapper.
'#
'#############################################################

'***********************************************************
'
' String variables that hold the locations where to find
' the configuation file (cbdata.cfg), and the temporary
' directory for this application during run time.
'
'***********************************************************
Private cfgfile As String
Private cfgtmp As String
'***********************************************************
'
' Unload the CBcfg form when the Cancel button is clicked.
'
'***********************************************************
Private Sub Cancel_Click()
    Unload Me
End Sub
'***********************************************************
'
' Display information for each record selected from the
' current configuration list box.
'
'***********************************************************
Private Sub dataList_Click()
    Dim listStr As String
    Dim typeStr As String
    Dim locationStr As String
    Dim virtualStr As String

    listStr = dataList.Text
    Call lineInfo(listStr, typeStr, locationStr, virtualStr)
    ' Display the key name in the Key text box.
    TypeText.Text = typeStr
```

```
    ' Display the directory associated with the key in the
    ' Directory text box.
    LocationText.Text = locationStr
    ' Display the virtual directory associated with the key
    ' in the Virtual Directory text box
    VirtualDirText.Text = virtualStr
    Add.Enabled = False
    Delete.Enabled = True
End Sub
'*********************************************************
'
' Tasks done when deleting an item from the list.
' First, copy all lines from the cfgfile to the cfgtmp
' file except the line that's being deleted. Then copy
' back to the cfgfile from the cfgtmp.
'
'*********************************************************
Private Sub Delete_Click()
    Open cfgfile For Input As #1
    Open cfgtmp For Output As #2
    Do While Not EOF(1)
        Line Input #1, inputStr
        If Not (InStr(1, inputStr, TypeText.Text & "=",
vbTextCompare) > 0) Then
            Print #2, inputStr
        End If
    Loop
    Close #1
    Close #2
    ' Copy the cfgtmp to the cfgfile
    Open cfgtmp For Input As #1
    Open cfgfile For Output As #2
    Do While Not EOF(1)
        Line Input #1, inputStr
        Print #2, inputStr
    Loop
    Close #1
    Close #2
    Call updateList
End Sub
'*********************************************************
'
' Tasks done when the application is load.
' This requires two system environment variables set,
' which are CB_HOME, where the cbdata.cfg is located, and
' CB_TMP, where the temporary file is created.
'
'*********************************************************
Private Sub Form_Load()
```

```
    cfgfile = Environ("CB_HOME") & "\cbdata.cfg"
    cfgtmp = Environ("TEMP") & "\cbdata_.tmp"
    Call updateList
End Sub
'*********************************************************
'
' Activate the Add button if new value is enterred from
' the Image type box.
'
'*********************************************************
Private Sub KeyText_Change()
    Add.Enabled = True
End Sub

'*********************************************************
'
' Save the changes (if any), and close the CBcfg form
' when the OK button is clicked
'
'*********************************************************
Private Sub OK_Click()
    If (Add.Enabled) Then
        Call Add_Click
    End If
    Unload Me
End Sub
'*********************************************************
'
' The lineInfo subroutine parses a line input from the
' configuration file (cbdata.cfg). It separates information
' of the key, the directory, and the virtual directory
' from the line string input.
' Parameters:
'       in:
'           searchStr - the string is being parsed.
'       in/out:
'           K - a variable that holds the key string
'           D - a variable that holds the directory string
'           V - a variable that holds the virtual directory
string
'
'*********************************************************
Private Sub lineInfo(searchStr As String, K As String, D As
String, V As String)
    istart = 1
    istop = 0
    istop = InStr(istart, searchStr, "=", vbTextCompare)
    ' Get the key string
    K = Mid(searchStr, istart, istop - 1)
```

```
    istart = istop + 1
    istop = InStr(istart, searchStr, "|", vbTextCompare)
    ' Get the directory string
    If istop > istart Then
        D = Mid(searchStr, istart, istop - istart)
        istart = istop + 1
        'Get the location string
        V = Mid(searchStr, istart)
    Else
        D = Mid(searchStr, istart)
        V = ""
    End If
End Sub
'*********************************************************
'
' Tasks done when adding an item to the list. First, check
' if there is any line from cfgfile that has the same key
' value as the added item. Then update it with the new
' value. Otherwise, add a new line (item) to the cfgfile.
'
'*********************************************************
Private Sub Add_Click()
    Add.Enabled = False
    Open cfgfile For Input As #1
    Open cfgtmp For Output As #2
    ' Check for whether or not the image type exists.
    Do While Not EOF(1)
        Line Input #1, inputStr
        If Not (InStr(1, inputStr, TypeText.Text & "=",
vbTextCompare) > 0) Then
            ' Write to a temporary file
            Print #2, inputStr
        End If
    Loop
    If (StrComp("", VirtualDirText.Text, vbTextCompare) = 0)
Then
        Print #2, TypeText.Text & "=" & LocationText.Text
    Else
        Print #2, TypeText.Text & "=" & LocationText.Text &
"|" & VirtualDirText.Text
    End If
    Close #1
    Close #2
    ' Copy the cfgtmp to the cfgfile
    Open cfgtmp For Input As #1
    Open cfgfile For Output As #2
    Do While Not EOF(1)
        Line Input #1, inputStr
        Print #2, inputStr
```

```
    Loop
    Close #1
    Close #2
    Call updateList
End Sub
'*********************************************************
'
' Activate the Add button if new value is enterred from
' the Key text box.
'
'*********************************************************
Private Sub TypeText_Change()
    Add.Enabled = True
End Sub
'*********************************************************
'
' Activate the Add button if new value is enterred from
' the Directory text box.
'
'*********************************************************
Private Sub locationText_Change()
    Add.Enabled = True
End Sub
'*********************************************************
'
' Refresh the GUI after adding or deleting an item from
' the list.
'
'*********************************************************
Private Sub updateList()
    Dim intFile As Integer
    dataList.Clear

    intFile = FreeFile()
    Open cfgfile For Input As #intFile
    Do While Not EOF(intFile)    ' Check for end of file.
       Line Input #intFile, inputStr    ' Read line of data.
       dataList.AddItem inputStr
    Loop
    Close #intFile
    TypeText.Text = ""
    LocationText.Text = ""
    VirtualDirText.Text = ""
    Add.Enabled = False
    Delete.Enabled = False
End Sub
'*********************************************************
'
' Activate the Add button if new value is enterred from
```

```
    ' the Virtual Directory text box.
    '
    '*********************************************************
    Private Sub VirtualDirText_Change()
        Add.Enabled = True
    End Sub
```

2. Application Wrapper (CBWrapper)

```
    VERSION 5.00
    Object = "{48E59290-9880-11CF-9754-00AA00C00908}#1.0#0";
"MSINET.OCX"
    Begin VB.UserControl WebInterface
       BackColor       =   &H80000001&
       ClientHeight    =   5475
       ClientLeft      =   0
       ClientTop       =   0
       ClientWidth     =   8430
       ScaleHeight     =   5475
       ScaleWidth      =   8430
       Begin InetCtlsObjects.Inet Inet1
          Left            =   120
          Top             =   120
          _ExtentX        =   1005
          _ExtentY        =   1005
          _Version        =   393216
       End
       Begin VB.TextBox ImagesText
          BeginProperty Font
             Name            =   "Arial"
             Size            =   9.75
             Charset         =   0
             Weight          =   700
             Underline       =   0   'False
             Italic          =   0   'False
             Strikethrough   =   0   'False
          EndProperty
          Height          =   375
          Left            =   5880
          TabIndex        =   7
          Text            =   "24"
          Top             =   1680
          Width           =   735
       End
       Begin VB.TextBox HeightText
          BeginProperty Font
             Name            =   "Arial"
             Size            =   9.75
             Charset         =   0
```

```
      Weight           =   700
      Underline        =   0   'False
      Italic           =   0   'False
      Strikethrough    =   0   'False
   EndProperty
   Height           =   375
   Left             =   5880
   TabIndex         =   6
   Text             =   "540"
   Top              =   2520
   Width            =   735
End
Begin VB.TextBox WidthText
   BeginProperty Font
      Name             =   "Arial"
      Size             =   9.75
      Charset          =   0
      Weight           =   700
      Underline        =   0   'False
      Italic           =   0   'False
      Strikethrough    =   0   'False
   EndProperty
   Height           =   375
   Left             =   5880
   TabIndex         =   5
   Text             =   "720"
   Top              =   3360
   Width            =   735
End
Begin VB.TextBox DurationText
   BeginProperty Font
      Name             =   "Arial"
      Size             =   9.75
      Charset          =   0
      Weight           =   700
      Underline        =   0   'False
      Italic           =   0   'False
      Strikethrough    =   0   'False
   EndProperty
   Height           =   375
   Left             =   5880
   TabIndex         =   4
   Text             =   "0"
   Top              =   4200
   Width            =   735
End
Begin VB.CommandButton Start
   Caption          =   "Start"
   BeginProperty Font
```

```
      Name              =   "Arial"
      Size              =   9.75
      Charset           =   0
      Weight            =   700
      Underline         =   0   'False
      Italic            =   0   'False
      Strikethrough     =   0   'False
   EndProperty
   Height            =   495
   Left              =   720
   TabIndex          =   3
   Top               =   2400
   Width             =   1215
End
Begin VB.CommandButton Default
   Caption           =   "Default"
   BeginProperty Font
      Name              =   "Arial"
      Size              =   9.75
      Charset           =   0
      Weight            =   700
      Underline         =   0   'False
      Italic            =   0   'False
      Strikethrough     =   0   'False
   EndProperty
   Height            =   495
   Left              =   720
   TabIndex          =   2
   Top               =   4080
   Width             =   1215
End
Begin VB.ComboBox ImageType
   BeginProperty Font
      Name              =   "Arial"
      Size              =   9.75
      Charset           =   0
      Weight            =   700
      Underline         =   0   'False
      Italic            =   0   'False
      Strikethrough     =   0   'False
   EndProperty
   Height            =   360
   Left              =   720
   TabIndex          =   1
   Text              =   "Select an image type"
   Top               =   1680
   Width             =   2895
End
Begin VB.CommandButton Stop
```

```
    BackColor       =   &H00C0C0C0&
    Caption         =   "Stop"
    BeginProperty Font
       Name             =   "Arial"
       Size             =   9.75
       Charset          =   0
       Weight           =   700
       Underline        =   0   'False
       Italic           =   0   'False
       Strikethrough    =   0   'False
    EndProperty
    Height          =   495
    Left            =   720
    MaskColor       =   &H80000004&
    TabIndex        =   0
    Top             =   3240
    Width           =   1215
End
Begin VB.Label images
    BackColor       =   &H80000001&
    Caption         =   "Images:"
    BeginProperty Font
       Name             =   "Arial"
       Size             =   9.75
       Charset          =   0
       Weight           =   700
       Underline        =   0   'False
       Italic           =   0   'False
       Strikethrough    =   0   'False
    EndProperty
    ForeColor       =   &H8000000E&
    Height          =   255
    Left            =   4800
    TabIndex        =   14
    Top             =   1680
    Width           =   855
End
Begin VB.Label Label1
    BackColor       =   &H80000001&
    Caption         =   "Height:"
    BeginProperty Font
       Name             =   "Arial"
       Size             =   9.75
       Charset          =   0
       Weight           =   700
       Underline        =   0   'False
       Italic           =   0   'False
       Strikethrough    =   0   'False
    EndProperty
```

```
      ForeColor       =   &H8000000E&
      Height          =   255
      Left            =   4800
      TabIndex        =   13
      Top             =   2520
      Width           =   735
   End
   Begin VB.Label Label2
      BackColor       =   &H80000001&
      Caption         =   "Width:"
      BeginProperty Font
         Name            =   "Arial"
         Size            =   9.75
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H8000000E&
      Height          =   255
      Left            =   4800
      TabIndex        =   12
      Top             =   3360
      Width           =   735
   End
   Begin VB.Label Label3
      BackColor       =   &H80000001&
      Caption         =   "Duration:"
      BeginProperty Font
         Name            =   "Arial"
         Size            =   9.75
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H8000000E&
      Height          =   255
      Left            =   4800
      TabIndex        =   11
      Top             =   4200
      Width           =   855
   End
   Begin VB.Label Label4
      BackColor       =   &H80000001&
      Caption         =   "Second(s)"
      BeginProperty Font
```

```
      Name            =   "Arial"
      Size            =   9.75
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   ForeColor       =   &H8000000E&
   Height          =   255
   Left            =   6840
   TabIndex        =   10
   Top             =   4200
   Width           =   975
End
Begin VB.Label Label5
   Alignment       =   2  'Center
   BackColor       =   &H80000001&
   Caption         =   "CONTINUOUS BRIEF"
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   18
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   ForeColor       =   &H8000000E&
   Height          =   495
   Left            =   2280
   TabIndex        =   9
   Top             =   360
   Width           =   3975
End
Begin VB.Label type
   BackColor       =   &H80000001&
   Caption         =   "Image type:"
   BeginProperty Font
      Name            =   "Arial"
      Size            =   9.75
      Charset         =   0
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   ForeColor       =   &H8000000E&
   Height          =   255
```

```
      Left            =   720
      TabIndex        =   8
      Top             =   1200
      Width           =   1215
   End
End
Attribute VB_Name = "WebInterface"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
'#########################################################
'#  File: WebInterface.ctl
'#  Date                Author              History
'#  5/31/2000           Tam Tran            Created.
'#########################################################
Option Explicit
'*********************************************************
'
' The Continuous Brief wrapper (CBWrapper) is an ActiveX
' Control that represents the Graphical User Interface
' (GUI) via the Web browser (Internet Explorer). It allows
' an user to select the type of images that he/she wants
' to view. Also, it allows the user to set the number of
' images, the size, and the duration for the display.
'
'*********************************************************
Private mControllerConnector As ControllerConnector
Private mMonitor As Monitor
Private mMonitorConnector As MonitorConnector
Private WithEvents mController As Controller
Attribute mController.VB_VarHelpID = -1
' Get reference to Application object from the PowerPoint
API.
Public myPPT As PowerPoint.Application
Public AppRunning As Boolean
Private BriefStarted As Boolean
Private downloadFolder As String
Private cfgFolder As String
Private ServerURL As String

'*********************************************************
'
' Reset the Continuous Brief GUI to its default values.
' Set slide show to fullscreen size.
' Set number of images to 24
' Set duration of the slide show to 0.
'
'*********************************************************
```

```
Private Sub Default_Click()
    ImageType.Text = "Select an image type"
    ImagesText.Text = "24"
    HeightText.Text = "540"
    WidthText.Text = "720"
    DurationText.Text = "0"
End Sub

'*********************************************************
'
' Update the brief.
' Use the GetImageDir method from the Controller object
' to get the location of the files.
' Use the Controller_UpdateBrief method to update the brief.
'.
'*********************************************************
Private Sub Start_Click()
    Dim imageloc As String
    BriefStarted = True
    Call mController_UpdateBrief(ImageType.Text)
End Sub

'*********************************************************
'
' Stop the slide show.
' Terminate the background running PowerPoint application.
' Free up the un-used object.
' Reset the AppRunning flag to false.
'
'*********************************************************
Private Sub Stop_Click()
    If AppRunning Then
        myPPT.ActivePresentation.Close
        myPPT.Quit
        Set myPPT = Nothing
        AppRunning = False
        BriefStarted = False
    End If
End Sub

'*********************************************************
'
' Initialize references to the Monitor and Controller
objects.
'
'*********************************************************
Private Sub UserControl_Initialize()

    Set mControllerConnector = New ControllerConnector
```

```
        Set mController = mControllerConnector.Controller
        Set mMonitorConnector = New MonitorConnector
        Set mMonitor = mMonitorConnector.Monitor
        AppRunning = False
        BriefStarted = False

        ' Add image types to the drop-box in the Continuous
Brief GUI
        Dim intFile As Integer   ' FreeFile variable
        Dim inputStr As String
        Dim cfgFile As String
        Dim typeStr As String
        Dim locationStr As String
        Dim virtualDirStr As String
        Dim tmpFolderStr As String
        Dim tmpFileStr As String
        Dim downloadFileStr As String

        ' Set values for the URL, download folder, and a
temporary filename
        ' %%%%%%%%%%%%%%%%%%%%%
        ' Change config here:
        ServerURL = "http://tampc.spawar.navy.mil/"
        ' %%%%%%%%%%%%%%%%%%%%%
        cfgFile = "cbdata.cfg"
        downloadFolder = Environ("TEMP") & "\cbdownload"
        cfgFolder = downloadFolder & "\cbdata"
        tmpFileStr = cfgFolder & "\" & cfgFile

        ' Download the "cbdata.cfg" file
        downloadFileStr = ServerURL & "/" & cfgFile

        ' Create a temporary directory for downloading data
        Call createFolder(downloadFolder)
        Call createFolder(cfgFolder)
        Call downloadFile(downloadFileStr, tmpFileStr)

        intFile = FreeFile()
        Open tmpFileStr For Input As #intFile
        Do While Not EOF(intFile)
            Line Input #intFile, inputStr
            Call lineInfo(inputStr, typeStr, locationStr,
virtualDirStr)
            ImageType.AddItem typeStr
        Loop
        Close #intFile
    End Sub

    '*******************************************************
```

```
'
' Receive Controller event to do the update for the brief.
' Parameters:
'       in: DataType - the data (images) type
'       in: imageDir - the directory where to find the
images.
'
'*********************************************************
Private Sub mController_UpdateBrief(DataType As String)

    ' Check for the right type of data that the CBWrapper is
showing.
    If (StrComp(ImageType.Text, DataType, vbTextCompare) =
0) And BriefStarted Then
        Dim virtualDir As String
        Dim fileListName As String
        Dim tmpFileStr As String
        Dim tmpURLStr As String
        Call mController.GetImageInfo(ImageType.Text,
ImagesText.Text, _
                                       virtualDir,
fileListName)
        ' Local variables declarations
        Dim myArray() As String
        Dim myPres As Presentation
        Dim fs, f, fc, f1, i, j, K
        Dim s As Slide
        Dim LeftVal As Long
        Dim TopVal As Long
        Dim imageW As Long
        Dim imageH As Long
        Dim ImgFile As String
        Dim intFile As Integer
        Dim inputStr As String

        ' Download the list of image filenames from server
        tmpURLStr = ServerURL & virtualDir & "/CB_listfile/"
& fileListName
        tmpFileStr = cfgFolder & "\" & fileListName
        Call downloadFile(tmpURLStr, tmpFileStr)

        ' Download image files from server
        intFile = FreeFile()
        Open tmpFileStr For Input As #intFile
        Do While Not EOF(intFile)
            Line Input #intFile, inputStr
            tmpURLStr = ServerURL & virtualDir & "/" &
inputStr
            tmpFileStr = downloadFolder & "\" & inputStr
```

```
            Call downloadFile(tmpURLStr, tmpFileStr)
        Loop
        Close #intFile

        ' Get reference to the PowerPoint Application
object.
        On Error Resume Next
        Set myPPT = GetObject(, "PowerPoint.application")
        If Err.Number <> 0 Then
            Set myPPT =
CreateObject("PowerPoint.application")
        End If

        ' Set the AppRunning flag so that it will be
        ' checked when the STOP button is clicked.
        AppRunning = True

        ' Stop the current running slide show (if any)
        If myPPT.Presentations.Count <> 0 Then
            myPPT.ActivePresentation.Close
        End If

        ' Create new presentation with the new update data
        Set myPres = myPPT.Presentations.Add(True)

        ' Create a FileSystemObject for manipulating the
file system
        Set fs = CreateObject("Scripting.FileSystemObject")
        Set f = fs.GetFolder(downloadFolder)
        Set fc = f.Files
        i = 1
        K = 1

        ' Store all filenames from the image directory
        ' to an array for sorting purpose.
        ReDim myArray(1 To fc.Count)
        For Each f1 In fc
            myArray(i) = f1.Name
            i = i + 1
        Next
        ' Sort the array.
        Call mMonitor.dhBubbleSort(myArray)

        ' Calculate the positions and dimensions for the
images.
        Call GetDimensions(LeftVal, TopVal, imageW, imageH)

        ' Add the images to the PowerPoint presentation.
        For j = (fc.Count - ImagesText.Text + 1) To fc.Count
```

```
                    ImgFile = downloadFolder & "\" & myArray(j)
                    myPres.Slides.Add K, ppLayoutBlank
                    myPres.Slides.Item(K).Shapes.AddPicture
ImgFile, True, True, _

LeftVal, TopVal, imageW, imageH
                    K = K + 1
            Next
            'Free up the FileSystemObject when done
            Set fs = Nothing
            Set f = Nothing
            Set fc = Nothing

            ' Configure the slide show properties and run the
show
            For Each s In myPPT.ActivePresentation.Slides
                With s.SlideShowTransition
                    .AdvanceOnTime = True
                    .AdvanceTime = DurationText.Text
                End With
            Next

            With myPPT.ActivePresentation.SlideShowSettings
                .StartingSlide = 1
                .EndingSlide = ImagesText.Text
                .AdvanceMode = ppSlideShowUseSlideTimings
                .LoopUntilStopped = True
                .Run
            End With

        ' Delete the images when done creating the brief
        For i = 1 To fc.Count
            If fs.FileExists(downloadFolder & "\" & myArray(i))
Then
                Set f = fs.DeleteFile(downloadFolder & "\" &
myArray(i), True)
            End If
        Next
        End If
    End Sub

    '*******************************************************
    '
    ' The GetDimensions subroutine calculates the positions
    ' (Left, Top), and the dimensions (Height, Width)
    ' for the images.
    ' Parameters:
    '       in/out: L - the Left value
    '               T - the Top value
```

```
'               W - the Width value
'               H - the Height value
'
'*********************************************************
    Private Sub GetDimensions(L As Long, T As Long, W As Long, H
As Long)

        ' Local variables declarations
        Dim DeltaX As Long
        Dim DeltaY As Long

        DeltaX = myPPT.ActivePresentation.PageSetup.SlideWidth -
WidthText.Text
        DeltaY = myPPT.ActivePresentation.PageSetup.SlideHeight
- HeightText.Text
        If DeltaX <= 0 Then
            L = 0
        Else
            L = DeltaX / 2
        End If
        If DeltaY <= 0 Then
            T = 0
        Else
            T = DeltaY / 2
        End If
        W = WidthText.Text
        H = HeightText.Text
        If W > 720 Then W = 720
        If H > 540 Then H = 540
    End Sub
    '*********************************************************
    '
    ' The lineInfo subroutine parses a line input from the
    ' configuration file (cbdata.cfg). It separates information
    ' of the key, the directory, and the virtual directory
    ' from the line string input.
    ' Parameters:
    '       in:
    '           searchStr - the string is being parsed.
    '       in/out:
    '           K - a variable that holds the key string
    '           D - a variable that holds the directory string
    '           V - a variable that holds the virtual directory
string
    '
    '*********************************************************
    Private Sub lineInfo(searchStr As String, K As String, D As
String, V As String)
        Dim istart As Integer
```

```
        Dim istop As Integer
        istart = 1
        istop = 0
        istop = InStr(istart, searchStr, "=", vbTextCompare)
        ' Get the key string
        K = Mid(searchStr, istart, istop - 1)
        istart = istop + 1
        istop = InStr(istart, searchStr, "|", vbTextCompare)
        ' Get the directory string
        If istop > istart Then
            D = Mid(searchStr, istart, istop - istart)
            istart = istop + 1
            'Get the location string
            V = Mid(searchStr, istart)
        Else
            D = Mid(searchStr, istart)
            V = ""
        End If
    End Sub
    '*********************************************************
    '
    ' The downloadFile subroutine uses the OpenURL method to
    ' download a file from the current open connection using
    ' HTTP protocol.
    ' Parameters:
    '       in:
    '           URLStr - the URL for download the file from.
    '           saveFile - the filename for storing the
    '                      downloaded file on the client machine.
    '
    '*********************************************************
    Private Sub downloadFile(URLStr As String, saveFile As
String)
        Dim bData() As Byte      ' Data variable
        Dim intFile As Integer   ' FreeFile variable
        intFile = FreeFile()      ' Set intFile to an unused
file.

        ' The result of the OpenURL method goes into the Byte
        ' array, and the Byte array is then saved to disk.
        bData() = Inet1.OpenURL(URLStr, icByteArray)
        Open saveFile For Binary Access Write As #intFile
        Put #intFile, , bData()
        Close #intFile
    End Sub
    '*********************************************************
    '
    ' Creating a folder on client machine.
    ' Parameter:
```

```
'       in: path - a qualify name of the folder being
created.
'
'*********************************************************
Private Sub createFolder(path As String)
    Dim fs, f
    Set fs = CreateObject("Scripting.FileSystemObject")
    If Not fs.FolderExists(path) Then
        Set f = fs.createFolder(path)
    End If
    Set fs = Nothing
    Set f = Nothing
End Sub
'*********************************************************
'
' Deleting a folder on a client machine.
' Parameter:
'       in: path - a qualify name of the folder being
deleted.
'
'*********************************************************
Private Sub deleteFolder(path As String)
    Dim fs, f
    Set fs = CreateObject("Scripting.FileSystemObject")
    If fs.FolderExists(path) Then
        fs.deleteFolder path, True
    End If
    Set fs = Nothing
End Sub
'*********************************************************
'
' Clean up all temporary folder created when exiting.
'
'*********************************************************
Private Sub UserControl_Terminate()
    ' Delete the download folder
    deleteFolder downloadFolder
End sub
```

3. Object Components (Continuous Brief)

a) Global Variable Declarations

```
Attribute VB_Name = "GlobalDeclarations"
'#########################################################
###
'#  File: GlobalDeclarations.bas
'#  Date                 Author              History
'#  5/31/2000            Tam Tran            Created.
```

```
      '#####################################################
###
      Option Explicit
      '*****************************************************
***
      '
      ' The cfgInfo type is a record that stores the
information
      ' that read from the cvdata.cfg file (i.e., Key,
Directory,
      ' Virtual Directory, and the stamped date, which is
the last
      ' time the data is checked.)
      '
      '*****************************************************
***
      Public Type cfgInfo
          key As String
          path As String
          vir_path As String
          stampdate As Date
      End Type

      '*****************************************************
***
      '
      'Global variables used by the ControllerConnector
      '
      '*****************************************************
***

      Public gController As Controller    ' Reference to
controller object
      Public gControllerUseCount As Long  ' Global reference
count


      '*****************************************************
***
      '
      ' Global variables used by the MonitorConnector
      '
      '*****************************************************
***
      Public gMonitor As Monitor          'Reference to
monitor object
      Public gMonitorUseCount As Long     ' Global reference
count
```

```
      '*****************************************************
***
      '
      ' Global variables used by the Monitor and Controller
objects.
      '
      '*****************************************************
***
      Public gCfgArray() As cfgInfo
```

b) Timer

```
      VERSION 5.00
      Begin VB.Form Timing
         Caption         =   "Form1"
         ClientHeight    =   3195
         ClientLeft      =   60
         ClientTop       =   345
         ClientWidth     =   4680
         LinkTopic       =   "Form1"
         ScaleHeight     =   3195
         ScaleWidth      =   4680
         StartUpPosition =   3  'Windows Default
         Begin VB.Timer Clock
            Left            =   2160
            Top             =   1200
         End
      End
      Attribute VB_Name = "Timing"
      Attribute VB_GlobalNameSpace = False
      Attribute VB_Creatable = False
      Attribute VB_PredeclaredId = True
      Attribute VB_Exposed = False
      '#####################################################
###
      '#  File: Timing.frm
      '#  Date                 Author              History
      '#  5/31/2000            Tam Tran            Created.
      '#####################################################
###
      '*****************************************************
***
      '
      ' Set the clock interval to 5 second.
      ' The Monitor component uses this timer event to poll
the
      ' storage directory for new data (images).
      '
      '*****************************************************
***
```

```
Private Sub Form_Load()
    Clock.Interval = 5000
End Sub
```

c) Controller

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "Controller"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
'#########################################################
###
'#  File: Controller.cls
'#  Date                Author              History
'#  5/31/2000           Tam Tran            Created.
'#########################################################
###

Option Explicit

'*********************************************************
***
'
' The Controller component uses this UpdateBrief event
to
' notify the Continuous Brief wrapper (CBWrapper) for
' updating the brief.
' Event's parameters:
'       imageType: the type of images
'       imageLoc: the location where to find the
images.
'
' The Glue component will raise the event to notify
the
' Controller when it's done with storing data.
'
' The Monitor component will raise the event to notify
the
' Controller when the new data come in.
```

```
      ' WithEvents causes the component(s) which raise the
event(s)
      ' to run asynchronously.
      ' MonitorConnector component allows multiple
connections to
      ' single Monitor object.
      '
      '*****************************************************
***
      Event UpdateBrief(imageType As String)

      Public WithEvents mGlue As Glue
      Attribute mGlue.VB_VarHelpID = -1
      Private WithEvents mMonitor As Monitor  ' Get Monitor
events
      Attribute mMonitor.VB_VarHelpID = -1
      Private mMonitorConnector As MonitorConnector

      '*****************************************************
***
      '
      ' Connect to the Monitor component
      '
      '*****************************************************
***
      Private Sub Class_Initialize()

        Set mMonitorConnector = New MonitorConnector
        Set mMonitor = mMonitorConnector.Monitor

      End Sub

      '*****************************************************
***
      '
      ' Receive the notification from the Monitor component
      ' The Controller passes the information to the Glue
component
      ' for storing data to the database.
      ' Event's paramenter:
      '       DataType: the data (images) type
      '
      '*****************************************************
***
      Private Sub mMonitor_NewData(DataType As String)
          Set mGlue = New Glue
          Call mGlue.StoreData(DataType)
      End Sub
```

```
      '*****************************************************
***
      '
      ' Receive the notification from the Glue component
that
      ' Asynchronous glue component is done.
      ' The Controller notifies the CBWrapper(s) and passes
the
      ' information for the wrapper(s) to update the
brief(s).
      ' Event's paramenter:
      '       DataType: the data (images) type
      '
      '*****************************************************
***
      Private Sub mGlue_GlueDone(DataType As String)
          Set mGlue = Nothing     ' Free the Glue object

          ' Notify the CBWrapper for updating the brief
          RaiseEvent UpdateBrief(DataType)
      End Sub
      '*****************************************************
***
      '
      ' Get all the image's filenames, which is being
requested
      ' from the CBWrapper, and make the makeFileList
function
      ' call to store the filenames to the CB_DATA.LST file.
      ' Parameters:
      '       in:
      '           ImageID - the image type
      '           fileCounts - the number of images
requested.
      '           virtualDir - the virtural directory
associated
      '                        with the images' directory.
      '       in/out:
      '           fileListName - a variable that holds the
filename,
      '                which contains the list of images'
filenames.
      '
      '*****************************************************
***
      Public Sub GetImageInfo(ImageID As String, fileCounts
As Integer, _
                              virtualDir As String,
fileListName As String)
```

```
        Dim i As Integer
        For i = 1 To UBound(gCfgArray)
            If (StrComp(ImageID, gCfgArray(i).key,
vbTextCompare) = 0) Then
                virtualDir = gCfgArray(i).vir_path
                fileListName = "CB_DATA.LST"
                Call makeFileList(fileCounts,
gCfgArray(i).path, fileListName)
            End If
        Next
    End Sub
    '*****************************************************
***
    '
    ' Write all filenames from a specified directory to a
file.
    ' This subroutine is called by GetImageInfo()
    ' Parameters:
    '       in:
    '           fileCounts - number of files is being
read.
    '           path - a specified directory for getting
the filenames.
    '           filename - the file used for storing the
filenames.
    '
    '*****************************************************
***
    Private Sub makeFileList(fileCounts As Integer, path
As String, _
                                        filename As
String)
        Dim fs, f, fc, f1, i, j, a
        Dim myCount As Integer
        Dim listfileStr As String
        Dim myArray() As String

        ' Create a FileSystemObject for manipulating the
file system.
        Set fs =
CreateObject("Scripting.FileSystemObject")
        Set f = fs.GetFolder(path)
        Set fc = f.Files
        myCount = fc.Count
        i = 1

        ' Store the name of the files to an array for
sorting purpose
```

```
            ReDim myArray(1 To myCount)
            For Each f1 In fc
                myArray(i) = f1.Name
                i = i + 1
            Next

            ' Sort the array
            Call mMonitor.dhBubbleSort(myArray)
            listfileStr = path & "\" & "CB_listfile"
            createFolder listfileStr
            Set a = fs.CreateTextFile(listfileStr & "\" &
filename, True)
            For j = (myCount - fileCounts + 1) To myCount
                a.WriteLine (myArray(j))
            Next
            a.Close
            ' Free up the objects, which are no longer be
used.
            Set fs = Nothing
            Set f = Nothing
            Set fc = Nothing
            Set a = Nothing
        End Sub
        '*****************************************************
***
        '
        ' This createFolder is used for creating a specified
folder.
        ' Parameter:
        '       in: path - the qualified name of the folder
being created.
        '
        '*****************************************************
***
        Private Sub createFolder(path As String)
            Dim fs, f
            Set fs =
CreateObject("Scripting.FileSystemObject")
            If Not fs.FolderExists(path) Then
                Set f = fs.createFolder(path)
            End If
            Set fs = Nothing
            Set f = Nothing
        End Sub
```

d) Controller Connector

```
        VERSION 1.0 CLASS
        BEGIN
          MultiUse = -1  'True
```

```
    Persistable = 0  'NotPersistable
    DataBindingBehavior = 0  'vbNone
    DataSourceBehavior  = 0  'vbNone
    MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "ControllerConnector"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
'#########################################################
###
'#  File: ControllerConnector.cls
'#  Date                Author              History
'#  5/31/2000           Tam Tran            Created.
'#########################################################
###

Option Explicit

'*******************************************************
***
'
' This property allows other components to get
reference
' to the Controller object.
'
'*******************************************************
***
Public Property Get Controller() As Controller
    Set Controller = gController
End Property

'*******************************************************
***
'
' Initilize Controller and reference count.
'
'*******************************************************
***
Private Sub Class_Initialize()
   If gController Is Nothing Then
      Set gController = New Controller
   End If
   gControllerUseCount = gControllerUseCount + 1
End Sub

'*******************************************************
***
```

```
      '
      ' Terminate controller when reference count = 0
      '
      '*****************************************************
***
      Private Sub Class_Terminate()
         gControllerUseCount = gControllerUseCount - 1
         If gControllerUseCount = 0 Then
            'Set gList = Nothing
            Set gController = Nothing
         End If
      End Sub
```

e) Monitor

```
      VERSION 1.0 CLASS
      BEGIN
        MultiUse = -1  'True
        Persistable = 0  'NotPersistable
        DataBindingBehavior = 0  'vbNone
        DataSourceBehavior  = 0  'vbNone
        MTSTransactionMode  = 0  'NotAnMTSObject
      END
      Attribute VB_Name = "Monitor"
      Attribute VB_GlobalNameSpace = False
      Attribute VB_Creatable = True
      Attribute VB_PredeclaredId = False
      Attribute VB_Exposed = True
      '#########################################################
###
      '#  File: Monitor.cls
      '#  Date                Author              History
      '#  5/31/2000           Tam Tran            Created.
      '#########################################################
###
      Option Explicit
      '*****************************************************
***
      ' The VISStamDate, IRStampDate, and VAPORStampDate
variables
      ' store the created date of the latest stored data.
      '
      ' WithEvents causes the component(s) which raise the
event(s)
      ' to run asynchronously.
      ' Event's parameter:
      '       DataType: the data (images) type
      '
      ' The Monitor component will raise the event to notify
the
```

```
      ' Controller when the new data come in.
      '*******************************************************
***
      Private VISStampDate As Date
      Private IRStampDate As Date
      Private VAPORStampDate As Date

      Private mTiming As Timing
      Private WithEvents mClock As Timer
      Attribute mClock.VB_VarHelpID = -1

      Event NewData(DataType As String)
      '*******************************************************
***
      '
      ' The tasks done when a new Monitor object is created.
      '
      '*******************************************************
***
      Private Sub Class_Initialize()

          ' Start Monitor Timer and create instance of form
          Set mTiming = New Timing
          Load mTiming

          ' Connect timers' events to associated event
procedures in Monitor
          Set mClock = mTiming.Clock

          ' Get the config information from the
configuration file
          Call GetConfig
      End Sub

      '*******************************************************
***
      '
      ' The tasks done when the Monitor object is
terminated.
      '
      '*******************************************************
***
      Private Sub Class_Terminate()    ' Terminate Monitor

          ' Free up the timer object.
          Set mClock = Nothing

          ' Unload and free up the form.
          Unload mTiming
```

```
        Set mTiming = Nothing
    End Sub

    '*****************************************************
***
    '
    ' Process Timer Event.
    ' This timer event causes the Monitor to poll the
storage
    ' directories for new data.
    ' The Monitor will raise the event(s) if it found a
new data.
    '
    '*****************************************************
***
    Private Sub mClock_Timer()
        Dim i As Integer
        For i = 1 To UBound(gCfgArray)
            If IsNewFile(gCfgArray(i).path, i) Then
                RaiseEvent NewData(gCfgArray(i).key)
            End If
        Next
    End Sub

    '*****************************************************
***
    '
    ' The IsNewFile function is used to determine whether
or
    ' not a new data exists.
    ' Paramenters:
    '       in: StrDir - the directory where to check for
    '                    new data.
    '       in: StampDate - the created date of the latest
    '                       data from the previous
checked.
    ' Return:
    '       TRUE if there's new data, and FALSE otherwise.
    '
    '*****************************************************
***
    Private Function IsNewFile(StrDir As String,
arrayIndex As Integer) As Boolean
        ' Local variables declarations.
        Dim fs, f, fc, f1, i
        Dim myStamp As Date
        Dim myArray() As String
```

```
        ' Create a FileSystemObject for manipulating the
file system.
        Set fs =
CreateObject("Scripting.FileSystemObject")
        Set f = fs.GetFolder(StrDir)
        Set fc = f.Files
        i = 1

        ' Store the name of the files to an array for
sorting purpose
        ReDim myArray(1 To fc.Count)
        For Each f1 In fc
            myArray(i) = f1.Name
            i = i + 1
        Next

        ' Sort the array
        Call dhBubbleSort(myArray)

        ' Check for new file based on the file's created
date.
        myStamp = fs.GetFile(StrDir & "\" &
myArray(fc.Count)).DateCreated
        If (DateDiff("s", gCfgArray(arrayIndex).stampdate,
myStamp) <> 0) Then
            gCfgArray(arrayIndex).stampdate = myStamp
            IsNewFile = True
        Else
            IsNewFile = False
        End If

        ' Free up the objects, which are no longer be
used.
        Set fs = Nothing
        Set f = Nothing
        Set fc = Nothing
    End Function

    '*****************************************************
***
    ' Standard bubblesort.
    ' DON'T USE THIS unless you know the data is already
    ' almost sorted! It's incredibly slow for
    ' randomly sorted data.

    ' There are many variants on this algorithm.
    ' There may even be better ones than this.
    ' But it's not even going to win any
    ' speed prizes for random sorts.
```

```
        ' From "Visual Basic Language Developer's Handbook"
        ' by Ken Getz and Mike Gilbert
        ' Copyright 2000; Sybex, Inc. All rights reserved.

        ' In:
        '   varItems:
        '       Array of items to be sorted.
        ' Out:
        '   VarItems will be sorted.
        '*****************************************************
***
      Public Sub dhBubbleSort(varItems As Variant)

          Dim blnSorted As Boolean
          Dim lngI As Long
          Dim lngJ As Long
          Dim lngItems As Long
          Dim varTemp As Variant
          Dim lngLBound As Long

          lngItems = UBound(varItems)
          lngLBound = LBound(varItems)

          ' Set lngI one lower than the lower bound.
          lngI = lngLBound - 1
          Do While (lngI < lngItems) And Not blnSorted
              blnSorted = True
              lngI = lngI + 1
              For lngJ = lngLBound To lngItems - lngI
                  If varItems(lngJ) > varItems(lngJ + 1)
Then
                      varTemp = varItems(lngJ)
                      varItems(lngJ) = varItems(lngJ + 1)
                      varItems(lngJ + 1) = varTemp
                      blnSorted = False
                  End If
              Next lngJ
          Loop
      End Sub
      '*****************************************************
***
      '
      ' The lineInfo subroutine parses a line input from the
      ' configuration file (cbdata.cfg). It separates
information
      ' of the key, the directory, and the virtual directory
      ' from the line string input.
      ' Parameters:
```

```
'       in:
'           searchStr - the string is being parsed.
'       in/out:
'           K - a variable that holds the key string
'           D - a variable that holds the directory
string
'           V - a variable that holds the virtual
directory string
'
'*****************************************************
***
    Private Sub lineInfo(searchStr As String, K As String,
D As String, V As String)
        Dim istart As Integer
        Dim istop As Integer

        istart = 1
        istop = 0
        istop = InStr(istart, searchStr, "=",
vbTextCompare)
        ' Get the key string
        K = Mid(searchStr, istart, istop - 1)
        istart = istop + 1
        istop = InStr(istart, searchStr, "|",
vbTextCompare)
        ' Get the directory string
        If istop > istart Then
            D = Mid(searchStr, istart, istop - istart)
            istart = istop + 1
            'Get the location string
            V = Mid(searchStr, istart)
        Else
            D = Mid(searchStr, istart)
            V = ""
        End If
    End Sub
    '*****************************************************
***
    '
    ' The GetDateArrayIndex function returns an index of
the
    ' dateArray, where the specified image type (ID) is
stored.
    '
    '*****************************************************
***
    Public Function GetArrayIndex(key As String) As
Integer
        Dim tmpInfo As cfgInfo
```

```
        Dim bFound As Boolean
        Dim i As Integer
        bFound = False
        i = 1
        Do While Not bFound
            tmpInfo = gCfgArray(i)
            If (StrComp(tmpInfo.key, key) = 0) Then
                GetArrayIndex = i
                bFound = True
            End If
            i = i + 1
        Loop
    End Function
    '*****************************************************
***
    '
    ' The GetConfig subroutine reads information stored in
    ' the configuration file, and adds them to the link
list.
    '
    '*****************************************************
***
    Private Sub GetConfig()

        Dim cfgpath As String
        Dim inputStr As String
        Dim keyStr As String
        Dim dirStr As String
        Dim virDirStr As String
        Dim intFile As Integer
        Dim tmpInfo As cfgInfo

        ' Initialize the size the gCfgArray
        ReDim gCfgArray(0)
        ' Get the path for the configuration file
        cfgpath = Environ("CB_HOME") & "\cbdata.cfg"

        ' Store the configured info to the array
        intFile = FreeFile()
        Open cfgpath For Input As #intFile
        Do While Not EOF(intFile)
            Line Input #intFile, inputStr
            Call lineInfo(inputStr, keyStr, dirStr,
virDirStr)
            With tmpInfo
                .key = keyStr
                .path = dirStr
                .vir_path = virDirStr
```

```
                .stampdate = -1      ' initialize the date
to before Dec. 30, 1899
            End With
            ReDim Preserve gCfgArray(UBound(gCfgArray) +
1)
            gCfgArray(UBound(gCfgArray)) = tmpInfo
        Loop
        Close #intFile
    End Sub
```

f) Monitor Connector

```
    VERSION 1.0 CLASS
    BEGIN
      MultiUse = -1  'True
      Persistable = 0  'NotPersistable
      DataBindingBehavior = 0  'vbNone
      DataSourceBehavior  = 0  'vbNone
      MTSTransactionMode  = 0  'NotAnMTSObject
    END
    Attribute VB_Name = "MonitorConnector"
    Attribute VB_GlobalNameSpace = False
    Attribute VB_Creatable = True
    Attribute VB_PredeclaredId = False
    Attribute VB_Exposed = True
    '#########################################################
###
    '#  File: MonitorConnector.cls
    '#  Date                Author              History
    '#  5/31/2000           Tam Tran            Created.
    '#########################################################
###

    Option Explicit
    '*******************************************************
***
    '
    '  This property allows other components to get
reference
    ' to the Monitor object.
    '
    '*******************************************************
***
    Public Property Get Monitor() As Monitor
        Set Monitor = gMonitor
    End Property
    '*******************************************************
***
    '
    ' Initialize Monitor and reference count.
```

```
      '
      '*****************************************************
***
      Private Sub Class_Initialize()
         If gMonitor Is Nothing Then
            ' Creates a new link list for holding the
configuration info.
            Set gMonitor = New Monitor
         End If
         gMonitorUseCount = gMonitorUseCount + 1
      End Sub
      '*****************************************************
***
      '
      ' Terminate Monitor when reference count = 0
      '
      '*****************************************************
***
      Private Sub Class_Terminate()
         gMonitorUseCount = gMonitorUseCount - 1
         If gMonitorUseCount = 0 Then
            Set gMonitor = Nothing
         End If
      End Sub
```

g) Glue

```
      VERSION 1.0 CLASS
      BEGIN
        MultiUse = -1  'True
        Persistable = 0  'NotPersistable
        DataBindingBehavior = 0  'vbNone
        DataSourceBehavior  = 0  'vbNone
        MTSTransactionMode  = 0  'NotAnMTSObject
      END
      Attribute VB_Name = "Glue"
      Attribute VB_GlobalNameSpace = False
      Attribute VB_Creatable = True
      Attribute VB_PredeclaredId = False
      Attribute VB_Exposed = True
      '#########################################################
###
      '#  File: Glue.cls
      '#  Date                Author              History
      '#  5/31/2000           Tam Tran            Created.
      '#########################################################
###

      Option Explicit
```

```
      '*****************************************************
***
      '
      ' The Glue component uses this event to notify the
      ' Controller when done with its task.
      ' Event's parameter:
      '       DataType: the data (images) type.
      '
      '*****************************************************
***
      Event GlueDone(DataType As String)

      '*****************************************************
***
      '
      ' Notify the Controller when done storing data.
      '
      '*****************************************************
***
      Public Sub StoreData(DataType As String)   ' Start glue
task
          ' <Insert glue task here>
          ' ...
          RaiseEvent GlueDone(DataType)
      End Sub
```

THESIS PROPOSAL

A. THESIS TITLE: XML Schema Integration

B. GENERAL INFORMATION

1. Name: Robert F. Halle
2. Curriculum: Software Engineering (369)
3. Thesis Advisor: Dr. Valdis Berzins
4. Second Reader: CAPT Paul Young?

C. AREA OF RESEARCH

Examination of four separate legacy database schemas to determine similarities and recommend common Extensible Markup Language (XML) data elements/schemas that could be used to support scalability towards modern C4I systems.

D. RESEARCH QUESTIONS

PRIMARY

1. Can an XML schema be defined to support scalability of components from multiple legacy database systems to modern C4I systems?

SUBSIDIARY

2. What components of existing legacy database sharing schemas can be carried forward?
3. What methods are required to assure scalability of legacy migration to C4I systems?
4. What XML schemas can be recommended to address the database mitigation?

E. DISCUSSION

An extensive amount of digital data information is available to the Army, Air Force, Navy, and Marine Corps leadership to support the planning and execution of military deployments. The planning effort is a complex undertaking requiring the consideration of multiple databases containing varied information before an effective plan can be developed. In order to deal with the growing amounts of differentiated databases available to the leadership, automated planning and database management C4I systems are being developed. As new mission requirements are defined, existing planning systems must be upgraded or new C4I systems must be developed. An analysis is required to determine what parts of the legacy database systems could be migrated to the modern C4I systems by using XML based methods that can identify common elements between databases. These identified common elements then can be used to assist in the database migration process.

F. SCOPE OF THESIS

The primary focus of this effort will be to execute an analysis of database components and of XML based database analysis schemas. The results of this analysis will be used to develop a recommendation on how common data elements can be identified using XML based analyses. These identified common elements can then be employed in the support scalability of the databases to meeting the growing C4I requirements. Originally proposed databases to be analyzed included AFATDS (Interbase) and JCDB (Informix). **Note:** Due to the unavailability of the AFATDS database, an in-depth examination of a more universal XML analysis approach will be presented in this research effort. This XML database analysis approach could be employed to identify common elements between most types of databases.

G. METHODOLOGY

This thesis describes an XML based analysis method that could be used to identify equivalent components of similar databases. There currently exists in the Department of

Defense multiple databases required to support command and control of some portion of the battlefield force. Interoperability between forces will be crucial as the force structure continues to be reduced. This interoperability will be facilitated through the integration of these command and control databases into a singular joint database or by developing inter-communication schemas to support inter-database communications. The first step in either of these alternatives is the identification of equivalent components/elements between the multiple databases.

This thesis will describe how XML can be used to facilitate the process of equivalent database component identification. Each step of the process will be described in detail accompanied by explanations of the XML tools/resources required to execute the step and rationale of why the step is necessary. Detailed graphics and examples will be employed whenever possible to simplify and justify the step by step explanations. This thesis will conclude with discussions of the overall value of this XML based analysis process and potential future work that could be pursued to further exploit this XML process.

H. CHAPTER OUTLINE

Introduction:
Background:
- Description of C4I systems.
- Description of related databases used in research (JCDB)
- Objective of research
- Description of XML
- Description of scope of research: Defines the bounds of research (JCDB/XML)
- Limitations: Anything that limited the research effort.
- Assumptions: Any assumptions made during the research.

Data and Physical Schema of Databases:
- Description of JCDB Data and Physical Schemas (including size of database).

Survey and Assessment of Previous Work:
- XML and Databases
- Description of DIICOE XML Namespace Registry/DIICOE SHADE.
- Description of XML/MTF efforts.
- Identification of previous methods for identifying common or semantically equivalent parts of databases.
- Opportunities for a new method.

Developed Method Description:
- Method components required and description.
- Process description and rationale why each step is required.
- Example of method using approximation of databases.
- Method advantages.
- Method disadvantages and limitation.

Conclusion/Recommendation/Future Work
- Conclusion: Final interpretation of research results
- Recommendation: Value of this method.
- Future Work: Description of additional research work that can be pursued.

Appendices
- Glossary
- References: References cited in thesis.
- Bibliography: Research references used in research.

THESIS PROPOSAL

I. SCHEDULE

1.	Comprehensive examination of databases, C4I system, XML schemas	01 Aug – 14 Aug
2.	Pursuit of Databases	14 Aug – 2 Nov
3.	Conduct research	21 Aug – 2 Nov
4.	Draft thesis	2 Nov – 31 Jan
5.	Final Thesis Submission/Signature	15 Feb

J. BENEFITS OF STUDY

This research will provide an assessment to JBC of the technical issues related to the use of XML to achieve data interoperability of military systems. This effort will add intellectual capital to the JBMI assessment and provide material on which to base future JBC XML projects.

K. ANTICIPATED TRAVEL/FUNDING REQUIREMENTS

None.

L. PRELIMINARY BIBLIOGRAPHY

Web Pages:

- DISA Homepage: www.disa.mil
- Microsoft XML Developers Homepage: msdn.microsoft.com/xml/default.asp
- Information Technology Standards Institute Homepage: www.itsi.disa.mil
- XML.Com Homepage: www.xml.com
- Oasis Homepage: www.oasis-open.org
- Zdnet Homepage: www.zdnet.co.uk
- World Wide Web Consortium Homepage: www.w3c.org
- Global Command and Control System (GCCS) Homepage: dod-ead.mont.disa.mil/aug_home/index.html
- Semi-Structured Data and Metadata Sub-Panel Homepage: disa.dtic.mil/coe/aog_twg/twg/ssdmd/ssd-md_page.html

Database Information:

- Joint Common Database
- GCCS API Reference Manual for NIPS Developers Segment (NDEV)
- GCCS Database Design Document for General Military Intelligence Database
- Implementation Guidelines for Interoperability with the GCCS-COP for Joint Warrior Interoperability Demonstration (JWID) 2000

Books:

- P. Anderson (and many others), Professional XML, WROX Press, 2000
- E. Tittel and F. Boumphrey, XML for Dummies, IDG Books Worldwide, 2000

Other:

- XML Recommendation 1.0, W3C, 10 Feb 1998
- G. Ray, Overview of XML, Microsoft Corp, Software Technology Conference Brief, 4 May 2000
- N. Nada, XML Technology Assessment, Naval Postgraduate School, 2000

THESIS PROPOSAL

A. THESIS TITLE: XML As A Data Exchange Medium In Real-Time Systems

B. GENERAL INFORMATION

1. Name: Kris Pradeep
2. Curriculum: Software Engineering (369)
3. Thesis Advisor: Dr. Valdis Berzins
4. Second Reader: Paul E. Young

C. AREA OF RESEARCH

Analysis of GCCS TDBM, GCCS I3 ISDS, JCDB and AFATDS legacy databases to determine a representative benchmark data. Create programs to generate random data sets that might be encountered in system interoperation.

D. RESEARCH QUESTIONS

PRIMARY

1. What components of the legacy databases schemas can be used as a subset of the overall data?

SUBSIDIARY

2. What message formats are used for transfer of messages?
3. What are the permissible ranges of values for the message parameters?

E. DISCUSSION

As the number of joint military operations being conducted around the world increases, interoperability of military systems becomes absolutely essential. Exchange of data between military computing systems is needed to establish interoperability. A large number of real-time systems are in use by today's military. Real-time systems often operate under very tight timing constraints. In proposing the use of XML as a data modeling and interchange standard, it is recognized that generating XML documents from native data, and later parsing that document into a different native data format creates a time delay in data delivery that some real-time systems may not be able to tolerate. This NPS project proposes to evaluate the impact of such a data interchange process on real-time systems.

F. SCOPE OF THESIS

The primary focus of this effort will be to develop programs that generate random data sets according to selected subsets of the database schemas. The subsets chosen will be representative of typical interactions encountered in system interoperations. This data set will form the benchmark data for later analysis.

G. METHODOLOGY

The thesis effort will evaluate four legacy databases schemas and will determine the subset to be used. In a normal system operation, information from these databases are searched and transmitted to other systems. Hence, the messaging formats employed for message transmission will be analysed. The subset schemas will be further

populated with the message formatting rules and parameter ranges. Finally, programs will be written to create messages from the subset containing random (valid) data for transmission.

H. CHAPTER OUTLINE

Introduction

Background
- Description of existing databases
- Description of evolving C4I systems
- XML in Real Time environment

Database Analysis
- Subset determination
- Message format determination
- Parameters evaluation

Algorithm
- Description of algorithm used to generate messages

Benchmark Data Set
- Evaluation of generated data subset

Conclusions

Appendices

Bibliography

I. SCHEDULE

1. Database Analysis	Aug 15 - Sep 30
2. Conduct research	Oct 01 - Nov 15
3. Draft thesis	Nov 30
4. Final Thesis	Dec 30

J. BENEFITS OF STUDY

This effort is part of a Professor Valdis Berzins coordinated effort with JBC. This research will provide an assessment to JBC of the technical issues related to the use of XML to achieve data interoperability of military systems. This effort will add intellectual capital to the JBMI assessment and provide material on which to base future JBC XML projects.

K. ANTICIPATED TRAVEL/FUNDING REQUIREMENTS

None.

L. PRELIMINARY BIBLIOGRAPHY

Web Pages:

- DISA Homepage: www.disa.mil
- Microsoft XML Developers Homepage: msdn.microsoft.com/xml/default.asp
- Information Technology Standards Institute Homepage: www.itsi.disa.mil
- XML.Com Homepage: www.xml.com
- Oasis Homepage: www.oasis-open.org
- Zdnet Homepage: www.zdnet.co.uk
- XMLInfo Home page: www.xmlinfo.com
- World Wide Web Consortium Homepage: www.w3c.org

Database Information:

- Joint Common Database
- GCCS API Reference Manual for NIPS Developers Segment (NDEV)
- GCCS Database Design Document for General Military Intelligence Database
- Implementation Guidelines for Interoperability with the GCCS-COP for Joint Warrior Interoperability Demonstration (JWID) 2000

Books:

- N. Bradley, The XML Companion, Addison Wesley, 2000
- B. Marchal , XML by Example, Que-Programming, 2000

Other:

- N. Nada, XML Technology Assessment Briefing, Naval Postgraduate School, 2000

THESIS PROPOSAL

A. THESIS TITLE: Common Data Attributes

B. GENERAL INFORMATION

1. Name: Hamza A. Zobair
2. Curriculum: Software Engineering (369)
3. Thesis Advisor: Dr. Valdis Berzins
4. Second Reader:

C. AREA OF RESEARCH

Analysis of GCCS MIDB, and JCDB legacy database schemas to determine similarities and recommend common data elements/schemas that could be used to support scalability towards modern C4I systems.

D. RESEARCH QUESTIONS

PRIMARY

1. What components of legacy database sharing attributes can be carried forward to modern C41 systems?

SUBSIDIARY

2. What methods are required to find common data attributes in legacy DOD systems?
3. What XML schemas can be recommended to address the database mitigation?

E. DISCUSSION

An extensive amount of digital data information is available to the Army, Air Force, Navy, and Marine Corps leadership to support the planning and execution of military deployments. The planning effort is a complex undertaking requiring the consideration of multiple databases containing varied information before an effective plan can be developed. In order to deal with the growing amounts of differentiated databases available to the leadership, automated planning and database management C4I systems have been developed. As new mission requirements are defined, existing planning systems must be upgraded or new C4I systems must be developed. An analysis is required to determine what parts of the legacy database systems could be transferred to the modern C4I systems and which data retrival methods can be to simplify this migration and support future migrations.

F. SCOPE OF THESIS

The primary focus of this effort will be to conduct an analysis of the two database components and their schemas. The results of this analysis will be used to develop a recommendation of how data elements can be employed to support scalability of the databases to support growing C4I requirements. Databases to be analyzed include JCDB and MIDBG.

G. METHODOLOGY

This thesis effort will evaluate two similar legacy databases and will derive the common data elements that are required to support scalability of these databases during the migration to more modern C4I systems. Based on our analysis we will recommend common XML based data element that could support the scalability of the legacy databases. The methodology to be employed in this effort will include analyses of each database along with side by side comparison of the databases to identify common elements. The current and future C4I systems database requirements will be acquired from program management offices and

analyzed to identify the scalability requirements of the databases. The portions of the legacy database sharing schemas that are required to the future C4I systems will be derived. Currently available XML schemas that support similar data sharing attributes will be examined to determine if there are any reusable components or approaches that could be employed in this research effort. XML schemas will be derived that support scalability of the existing data to meet future C4I requirements.

H. CHAPTER OUTLINE

Introduction

Background
- Description of existing databases and data sharing schemas.
- Description of evolving C4I systems.

Database Analysis/Comparison
- Analysis of existing databases and data sharing schema.
- Comparison and identification of common data elements.

XML Schema
- Analysis of similar XML schemas in SHADE.
- Derivation of XML schemas to support data sharing objectives.

Conclusions

Appendices

Bibliography

I. SCHEDULE

1.	Comprehensive examination of databases, C4I system, XML schemas	01 Aug – 21 Aug
2.	Construct research design process	21 Aug – 30 Aug
3.	Conduct research	21 Aug – 18 Sep
4.	Draft thesis	14 Aug – 29 Sep
5.	Final Thesis Submission/Signature	30 Sep

J. BENEFITS OF STUDY

This effort is part of a Professor Valdis Berzins coordinated effort with JBC. This research will provide an assessment to JBC of the technical issues related to the use of XML to achieve data interoperability of military systems. This effort will add intellectual capital to the JBMI assessment and provide material on which to base future JBC XML projects.

K. ANTICIPATED TRAVEL/FUNDING REQUIREMENTS

None.

L. PRELIMINARY BIBLIOGRAPHY

Web Pages:

- DISA Homepage: www.disa.mil
- Microsoft XML Developers Homepage: msdn.microsoft.com/xml/default.asp
- Information Technology Standards Institute Homepage: www.itsi.disa.mil
- XML.Com Homepage: www.xml.com
- Oasis Homepage: www.oasis-open.org
- Zdnet Homepage: www.zdnet.co.uk
- XMLInfo Home page: www.xmlinfo.com
- World Wide Web Consortium Homepage: www.w3c.org

Database Information:

- Joint Common Database
- GCCS API Reference Manual for NIPS Developers Segment (NDEV)
- GCCS Database Design Document for General Military Intelligence Database
- Implementation Guidelines for Interoperability with the GCCS-COP for Joint Warrior Interoperability Demonstration (JWID) 2000

Books:

- N. Bradley, The XML Companion, Addison Wesley, 2000
- B. Marchal , XML by Example, Que-Programming, 2000

Other:

- N. Nada, XML Technology Assessment Briefing, Naval Postgraduate School, 2000

Introduction

Traditionally databases have been developed by independent organizations to meet their immediate needs without concern for integration with other organizations. This type of developments is also known as stovepipe development. As the need arises to communicate with other organization it has been a difficult process to share common data and, this can result in expensive time consuming new database developments.
This problem has come to the forefront with recent acquisition and merger trend in industry where large established organizations are combining with other organizations. A need to share information between the new partners arises if their information systems were developed in a stovepipe method then sharing of information becomes difficult. Department of Defense (DOD) is faced with similar situation amongst its weapon platforms, the different services and it coalition partners.

Problem Statement

Our goal is to find common data attributes in Joint Common Database (JCDB) and the Modernized Integrated Database (MIDB). The search process will attempt to find best matches for each data attribute in MIDB to one equal to or similar in concept in the JCDB. We have chosen to find MIDB components because JCDB is a joint data base system that already has overlap between some Air Force and Army databases. Our understanding is that there was a previous effort to merge some data between JCDB and MIDB and that data related to Enemy clusters have already been incorporated into JCDB. In this effort we will follow this track by now finding common attributes in the following cluster groups: Target, Track and Observation.

The Joint Common Database system is a databases system of systems managed by the Air Force. It integrates the various Air Force specific systems as well as some Army and Navy database systems. MIDB is a database system that integrates Navy specific as well as intelligence data. A portion of the MIDB data has already been integrated into JCDB.

In the JCDB there are 187 primary tables; 268 look-up or reference set tables that are the data provider library to some of the columns in the primary tables. There are a total of 1251 columns in the JCDB, of which 974 are unique to a single table. The others appear in more than one table, i.e., Record_Status appears in all tables, or migrates to other tables through foreign key constraints.

In the MIDB there are ---- primary tables; ---look-up or reference set tables that are the data provider library to some of the columns in the primary tables. There is a total of ---- - columns in the MIDB, of which ----- are unique to a single table. FILL IN BLANKS Information Exchanged by the MIDB includes : Enemy Installation Data, Basic Encyclopedia # (BE), Latitude, Longitude, and General Military Intelligence (GMI).

MIDB includes regularly updated national and theater-level intelligence on facilities, Order-of-Battle (OB), equipment and targets, as well as locally derived intelligence entered by tactical intelligence assets. Data for the MIDB is generated by the Intelligence Shared Data Services (ISDS). ISDS is a component of the MIDB. ISDS also links

imagery to tracks and facilities identified in the Common Operating Picture (COP). The ISDS contains the Modernized Integrated Database (MIDB). MIDB supplies the information in this data exchange.

In our investigation we were provided raw data dictionaries. We were not provided with much additional detail information on the contents and specific applications of the database data. Thus we were not able to verify the accuracies of our results by domain experts from either of the database systems.

We plan on proceeding with our effort by first evaluating several search and data retrieval techniques so we can conduct our comparisons. The most popular techniques we have found include boolean logic, natural language, clustering, vector space, fuzzy searches, and neural networks. Some of these techniques can further be enhanced by allowing users to find matches with tools such as commercial thesaurus packages, user defined thesauruses, stemming, stopwords, destemming, proximity searches and indexing. We will discuss each of theses search techniques and tools below.

Search Types

Boolean Logic Searching with Boolean Logic involves constructing search queries using keywords and logic operators such as AND, OR and NOT. Such searches result in finding documents that contain one or more words that are specified in the user query. Some examples are AIR FORCE AND NAVY, AIRPLANE OR AIRCRAFT, and CODE AND (NOT ZIP CODE). When conducting a Boolean query it is good practice to start out with a broad search and gradually narrow the search to more specific topics. This can help prevent overlooking matching sets. For example as seen in referenced paper The Art of Text Query #, a market competitive analysis is carried out. Different sets are created using the following terms: market/competitive analysis/supermarket /exclusionary monopoly as key words in the queries. These key words are combined in different subsets to broaden or narrow the search. Similarly in our project as discussed below (section #) we use different sets to broaden or narrow our search.

Boolean logic searches have problems evaluating synonyms and homonyms. A Boolean search for the term stock would result in finding anything form the type of stocks that are traded on wall street to stock used in soup, stock yards associated with farming as well as merchandise that is on hand at a store. Synonym conflicts occur when two different databases use different names to describe the same concept. A homonym conflict occurs when two schemas use the same name to describe different concepts: for example an entity type PART may represent computer parts in one-schema and furniture parts in another schema. Boolean logic would also come up shy if someone were searching for all documents related to the term airport. All documents filed under the term airfield would be ignored unless they specifically mentioned airport. In order to reduce such synonym conflicts between databases, commercial thesaurus software packages and user-defined thesaurus can be used when conducting a search. WordNet® is a commercial

thesaurus developed by the Cognitive Science Lab at Princeton University that is used in one of the search engines used in my investigation.

Vector Space Model

A Vector Space Model is a representation of documents and queries where these are converted into vectors. The features of these vectors are usually words in the document or query, after stemming and removing stopwords (see definitions below for stemming and stopwords). The vectors are weighted to give emphasis to terms that exemplify meaning, and are useful in retrieval. In retrieval, the query vector is compared to each document vector. Those that are closest to the query are considered to be similar, and are returned. The vector model views each request as a series of 'n' dimensions in space, with 'n' corresponding to the number of word in the search request. The formula looks for the smallest vector angle between the search request and other documents, also viewed as 'n' dimensions in space.

Natural language

A natural language query is one that is expressed using normal conversational syntax; that is, you phrase your query as if making a spoken or written request to another person. There are no syntax rules or conventions for you to learn, as is the case in a query language. A natural language query is a query in which the search engine will typically look for all words within a search request. This process give result based on automatic terms weighting. The natural language searching technique uses a vector space model.

Natural language queries generally find more relevant information in less time than traditional Boolean queries, which, while precise, require strict interpretation that can often exclude information that is relevant to your interests.

Fuzzy Search

Fuzzy and phonic search technique search for words that match one or two deviations in letters: *aircraft*, and *aircaft*. Fuzzy search engines typically come with a feature that lets you control the amount of deviation so words such as artcraft would also match if the amount of deviation is increased. Fuzzy search can be useful for misspelled word or in the case when words are abbreviated such as in the DOD dictionaries like the ones we will be evaluating.

Phonic search has the capability to find word that sound the same but are spelled differently. For example a phonic search can find two and to or color and colour.

Stemming

This process typically remove prefixes and suffixes from words in a document or query in the formation of terms in the system's internal model. This is done to group words that have the same conceptual meaning, such as *Observe, Observation, Observing, and Observer*. Hence the user doesn't have to be so specific in a query. In general one must be careful when using the stemming functions because a search on *Aids* the disease could also find multiple hits on the topic *Aid*. Some search engines let users

modify/create stemming rules based on common prefixes and suffixes found in their data. Stemming, and proximity search techniques are also used to increase the likeliness of finding a match. These methods when used in boolean, vector space and natural language searches can enhance likelihood of finding appropriate matches.

Stopwords

Stopwords are words such as a preposition or article that have little semantic content. Typically search engines do not index stopwords. Stopword filters can also filter out words that have a high frequency in a document. Since stopwords appear in many documents, and are thus not helpful for retrieval, these terms are usually removed from the internal model of a search engine of a document or query. Some systems have a predetermined list of stopwords. However, stopwords could depend on context. The word COMPUTER would probably be a stopword in a collection of computer science journal articles, but not in a collection of articles from Consumer Reports. Depending on how the data dictionary is organized, words such as type and tables could be considered stopwords unless the stop word filter has been specifically turned off for these words.

Indexing

Indexing is the process of converting a collection into a form suitable for easy search and retrieval.

Weighting

Usually referring to terms, the process of giving more emphasis to the parameters for important terms is called weighting. In a vector space model, this is applied to the features of each vector. A popular weighting scheme is TF*IDF. Other possible schemes are Boolean (1 if the term appears, 0 if not), or by term frequency alone. In a vector model, the weights are sometimes normalized to sum to 1, or by dividing by the square root of the sum of their squares.

Cluster

A cluster is a grouping of representations of similar documents. In a vector space model, one can perform retrieval by comparing a query vector with the centroids of clusters. One can continue search in those clusters that are in this way most promising. Several programs have been developed to automatically cluster data into groups using clustering algorithms and formulas.

Content-Based Filtering

Content-based filtering refers to the process of filtering by extracting features from the text of documents to determine the documents' relevance. (Also called "cognitive filtering".)

Information Filtering

Given a large amount of data, information filtering returns the data that the user wants to see. This is the standard problem in information retrieval (IR).

Inverse Document Frequency

Abbreviated as IDF, this is a measure of how often a particular term appears across all of the documents in a collection. It is usually defined as log(collection size/number of documents containing the term). Common words will have a low IDF and words unique to a document will have a high IDF. This is typically used for weighting the parameters of a model.

Inverted File

An inverted file is a representation for a collection that is essentially an index. For each word or term that appears in the collection, an inverted file lists each document where it appears. This representation is especially useful for performing Boolean queries.

Precision

A standard measure of IR performance, precision is defined as the number of relevant documents retrieved divided by the total number of documents retrieved. For example, suppose there are 80 documents relevant to widgets in the collection. System X returns 60 documents, 40 of which are about widgets. Then X's precision is 40/60 = 67%. In an ideal world, precision is 100%. Since this is easy to achieve (by returning just one document), a system attempts to maximize both precision and recall simultaneously.

Query

A query is a string of words that characterizes the information that the user seeks. Note that this does not have to be an English language question.

Query Expansion

A query expansion is any process which builds a new query from an old one. It could be created by adding terms from other documents, as in relevance feedback, or by adding synonyms of terms in the query (as found in a thesaurus).

Recall

A standard measure of IR performance, recall is defined as the number of relevant documents retrieved divided by the total number of relevant documents in the collection. For example, suppose there are 80 documents relevant to widgets in the collection. System X returns 60 documents, 40 of which are about widgets. Then X's recall is 40/80 = 50%. In an ideal world, recall is 100%. However, since this is trivial to achieve (by retrieving all of the documents), a system attempts to maximize both recall and precision simultaneously.

Relevance

An abstract measure of how well a document satisfies the user's information need. Ideally, your system should retrieve all of the relevant documents for you. Unfortunately, this is a subjective notion and difficult to quantify.

Signature File

A signature file is a representation of a collection where documents are hashed to a bit string. This is essentially a compression technique to permit faster searching.

Similarity

Similarity is the measure of how alike two documents are, or how alike a document and a query are. In a vector space model, this is usually interpreted as how close their corresponding vector representations are to each other. A popular method for calculating similarity is to compute the cosine of the angle between the vectors.

Research

We reviewed several techniques for finding common data attributes in databases. The techniques we evaluated are Delta process, SEMINT, Query Flocks, and Eric Steirna's requirements matching technique.

Delta

The DELTA process uses a technique where they first collect all the data from the various databases then they reformat all the data into a standard text format. Then they suggest grouping the elements into "basic concept areas" or BCAs. The BCA process is a manual process where it is up to the user to group the data into BCAs. BCAs do not have to be completed before the attribute correlation is started, as other concepts will be found as the correlation process is carried out. BCAs are used to organize data element searches. The main reason for grouping the attributes into concept areas is to break the daunting task into smaller more manageable pieces. Delta finds attribute correspondences using a manual natural search process with a commercially available text search engine. Once correspondences are identified similar attributes or those attributes that would be used together are combined into a spreadsheet. A data model is created with the aid of entity relation diagrams for the data. Once the data model is created it is evaluated by domain experts for accuracy.

The process described in this section is not automatic. The search engine does the difficult time-consuming work of finding candidates among the thousands of possibilities. An analyst is still needed to decide the best match amongst the multiple candidates the search engine comes up with. With DELTA a single analyst was able to correlate about 200 data elements across four databases in one workweek. Assuming one workweek is 40 hours that equates to 5 data element matches per hour. Although it does not seem very quick it is still much quicker than finding matches without the aid of the natural language search engine. DELTA paper states that finding matches manually without the aid of any search engine required 4 hours per attribute. In our case with 1300 attributes, if we followed the DELTA process it would take about 260 hours or 6.5 forty hour work weeks.

Query Flocks

Query Flocks Association Rule Mining is a technique for optimizing data extraction from very large databases. Query Flocks is a generate and test model for data mining. They parameterize queries using filter conditions to eliminate values that are uninteresting.

Query safety is a well known condition that lets us enumerate the queries that are candidates for use in a query optimization technique that generalize a priori. One way to do this is to see generalized a-priori as a cast-based optimization principally involving order and selection of some useful subqueries. Another approach is to view the technique as one that is applied dynamically when the decision to perform an extra filtering step depending on the size of some intermediate relations.
In the Query flock strategy, the market basket problem is described to represent an attempt by a retail store to learn what items its customers frequently purchase together. In the query problem we are given a database containing information about "market basket". Each time a customer makes a purchase, information about what they bought is entered in the database. The aim of the market basket analysis is to find sets of items that are associated and the fact of their association is usually called an association rule. The precise measure of association rule includes support (the items must appear in many baskets) , confidence (the probability of the item given that the others in the basket should be high), and interest (that probability is significantly greater or lesser then the expected probability if the items were purchased at random).

A priori is a trick for speeding up the search for high -support set of items. It assumes the fact that if a set of items S appear in C baskets, then any subset of S appears an at least C baskets.

Eric Steirna
ERIC'S thesis develops both a manual process and tool to automate the identification of common requirements in two requirement documents. The outputs of both are reports that detail the requirements overlap between the two systems.

He used a manual matching process based on guidance received by the combat developers at AEC [WALE99] to establish initial pairs of matched requirements. He then used the insights gained in that process to develop a tool to partly automate requirements reuse. The Java-based tool extracts requirements systematically for an analyst with experience in the domain. The tool matches words between pairs of requirements and calculates a similarity rating based on word statistics. The tool provides the option to transform extracted requirements and domain entities into XML text files for integration into a reusable domain model.

The software that Eric developed used some of the searching techniques and tools discussed above. I have been able to find COT software to replicate much of the work that Eric conducted. For my analysis I will use the COTS software.

Semint
Neural Networks Using neural networks for data correlation is a recent technique employed by Lei and Clifton. The technique uses the metadata characteristics of data elements to train a neural network. Once the network is trained, the network assigns a signature based on the characteristics of each element within the metadata. The neural network is then used to find corresponding elements with signatures close to the one

being searched for. The process is automated and can do the search in seconds versus what was previously done in days.

The ability to integrate data from multiple databases can lead to many new applications. The problem with this is that many databases are heterogeneous in many aspects. Trying to merge individual schemas into large global databases faces problems also.

The steps of database integration include

- extracting semantics
- traversing formats
- identifying attribute correspondences and modifying heterogeneity
- multi-database query process
- data integration

The schema integration process includes schema transformation, followed by correspondences identification and an object integration and mapping step. The fundamental question in an approach to database system interoperability is identifying semantically related elements and than resolving the schematic differences. The key step is identifying attribute correspondences. Because manually comparing all possible pairs of attributes is an unreasonably large task, an automation process is desired. The goal of the research in the SEMINT paper is to develop a semi-automated semantic integration process that utilizes the metadata available in the database systems to identify attribute correspondences.

Attributes in different databases that represent the same real world concepts will have similar schema design, constraints and data value patterns. Three levels that can be automatically extracted from data bases and used to determine attribute correspondences are 1) attribute names (dictionary level), 2) schema information (field specification level) and 3) data contents and statistics (data content level). In this paper's approach neural networks are used for metadata extraction. How to match corresponding attributes and determine their similarity is learned during the training process directly from metadata.

SEMINT focuses on utilizing the metadata at the field specification level and data content level. The schema used by SEMINT include data type length and the existence of constraints, such as primary key, foreign keys, candidate keys, and value and range constraints, disallowing null values and restrictions. SEMINT automatically extracts schema information and constraints from the database catalogs and statistics on the data contents using queries on the data. The information extracted from different databases is then transformed into a single format and normalized. The advantage of having different parsers provided by SEMINT are:

- the queries to access the data dictionaries of various DBMS can be preprogrammed using C with embedded SQL
- AS DBMS specific parsers are preprogrammed the metadata extraction process is fully automated so no user intervention is needed

- The SEMINT users are not required to be aware of the differences of various DBMS. these differences are resolved by SEMINT DBMS specific parsers

In order to load the attribute data into a neural network it must be cleaned and converted to a readable or normalized form. SEMINT is able to automate this process by extracting data from the database itself. We do no have access to the database or the SEMINT software so we are unable to follow this process. It is the opinion of this author that attempting to manually normalize the data would take longer than it would to manually identify common attributes. However we do a multidata conjunctive search to identify our attribute. Our multidata search is similar to the vector signature search SEMINT conducts.

My Process

In my analysis I used a hybrid process which includes feature of Delta, SEMINT, and some of Eric Stierna's process. I initially started off by using the Delta process and the commercial PL software they used. However I found the process to be lacking in the sense it did not allow me to create a user defined thesaurus. As a result I would frequently miss potential matches because of synonyms or use of acronyms. Another problem with DELTA was the PL software requires one to convert the data to a specific format acceptable to PL. The process can take some time depending on the format the data dictionaries are provided in. In our case we had to conduct significant transformation of JCDB data to be accepted by PL. See figure # below to see a JCDB data element in original format and after reformatting. As a result I searched for additional text retrieval search engines and came up with dtSearch. The dtSearch software accepts data in most formats and does not require reformatting.

Format Data

First you must format the two databases into a common format using macros. The data attributes for the JCDB and MIDB have different formats. To simplify finding common attributes it is suggested that the user convert the attributes into a common format as displayed in Figure 2.

ATTRIBUTE NAME	PHYSICAL NAME	DEFINITION	DATA TYPE	NULL OPTION	ATTRIBUTE ENTITY[1]
ADDRESS postal code	POSTAL_CODE	The assigned "zip-code" for a specific POSTAL-ADDRESS	varchar(30)	NULL	ADDRESS

JCDB before formatting

```
-HEADER-    JCDB
ELEMENT NAME: ADDRESS postal code
ATTRIBUTE NAME: POSTAL_CODE
DEFINITION:  The assigned "zip-code" for a specific POSTAL-ADDRESS
DATA TYPE:     varchar(30)
NOPTIONS:NULL
TABLES:      ADDRESS
-END-
```

JCDB after formatting

1.	Element Name:	POSTAL_CODE
2.	Screen Label:	POSTAL CODE
3.		Description: Indicates the postal district of the entity.
4.	Structure:	varchar(30), NULL
5.	Permissible Values:	RUL_FREE_TEXT_EXP SPECIAL CHARACTERS. Special characters are restricted to apostrophe ('), at sign (@), parenthesis (), comma (,), period (.), semicolon (;), plus sign ( ), and dash (-). SPECIAL CHARACTERS. Special characters are restricted to apostrophe ('), at sign (@), parenthesis (), comma (,), period (.), semicolon (;), plus sign (+), and dash (-). Excluded characters are exclamation mark (!), pound sign (#), dollar sign ($), percent sign (%), up caret (^), ampersand (&), asterisk (*), underscore (_), equal sign (=), pipe (|), back and forward slashes (\/), grave accent (`), tilde (~), open and closed curly brackets ({}), double quotes ("), colon (:), question mark (?), greater and less than signs (><), and open and closed brackets ([]). These contraints are necessary on text fields to enable automated data exchange with systems with more restrictive data exchange formats.
6.		Tables: _address (IND_ADDRESS, FAC)

MIDB before formatting

```
-HEADER-    MIDB
 1.     Element Name:          POSTAL_CODE
 2.     Attribute Name:        POSTAL CODE
 3.     Definition:            Indicates the postal district of the entity.
 4.     Data Type:             varchar(30), NULL
 5.     Permissible Values:    RUL_FREE_TEXT_EXP
```

---

[1] In order to keep the length of the table within useable means, attributes sharing the same definition, data types, and nell option and found in multiple Entities will have the Entities listed alphabetically in the same row.

```
                    SPECIAL CHARACTERS.  Special characters are  restricted to
apostrophe ('), at sign (@), parenthesis (),  comma (,), period (.), semicolon (;), plus
sign (
                    ), and  dash (-).
                          SPECIAL CHARACTERS.  Special characters are  restricted to
apostrophe ('), at sign (@), parenthesis (),  comma (,), period (.), semicolon (;), plus
sign (+), and  dash (-).  Excluded characters are exclamation mark (!),  pound sign (#),
dollar sign ($), percent sign (%), up caret (^),  ampersand (&), asterisk (*), underscore
(_), equal sign (=),  pipe (|), back and forward slashes (\/), grave accent (`),  tilde
(~), open and closed curly brackets ({}),  double quotes ("), colon (:), question mark
(?), greater  and less than signs (><), and open and closed brackets  ([]).  These
contraints are necessary on text fields to  enable automated data exchange with systems
with  more restrictive data exchange formats.
 6.    Tables:              _address (IND_ADDRESS, FAC)

-END-___
```

Formatting is required as indicated by Benkly et all in their Delta process. According to Benkly et all when a manual search was conducted it took an average of 4 hours per attribute. This of course depends on the number of databases being evaluated and the number of elements in each data base. We are evaluating two databases with 1030 elements in one and 1340 elements in the other. Our technique, which involves incorporating portions of various techniques, was able to reduce average identification of common elements from 4 hours per element to 10 elements per hour. This is a considerable improvement compare to manually doing it by hand, however it still would take over a hundred hours of mundane work to complete the process. Neural networks offer opportunity to decrease the time, however converting or normalizing data is a key step that is required. In our case since we only received data dictionaries of the documents we were not able to normalize the data suitable for a neural network in a time efficient manner.

My Thesaurus

The semantic detail available in each database varies significantly. The JCDB does not offer as much detail as the MIDB. So a natural language search which uses frequency of occurrence of key word or strings in the search phrase is not the best alternative in some cases.

My first step was to develop a list of abbreviations and acronyms that are used in each of the databases. In the case of JCDB this information was readily available from the Data Dictionary for the Joint Common Database (JCDB) Version 4.3 in Appendix C Physical Naming Conventions file. The list of acronyms and abbreviations can then be added to the dtSearch software's user thesaurus (see Fig 1). A physical naming convention document was not provided for the MIDB for this effort. In its place we were able to scan through the data dictionary elements and determine the abbreviation and acronyms used by the GCCS MIDB. A compiled list of common synonyms, abbreviations and acronyms is provided in Table 1 below.

A user thesaurus allows one to add acronyms and common abbreviations in queries that typically would not be found in commercial thesaurus packages such as WordNet. A search conducted with a user thesaurus is a more restrictive search than one done using the WordNet thesaurus. Depending on the specific search strategies one can use a users thesaurus, a WordNet Thesaurus or the combination of the two. If a user is having difficulty finding any common attributes they may want to use a combination. If the combination returns multiple results, many of which are too general, a more restrictive search should be conducted using just the user defined thesaurus.

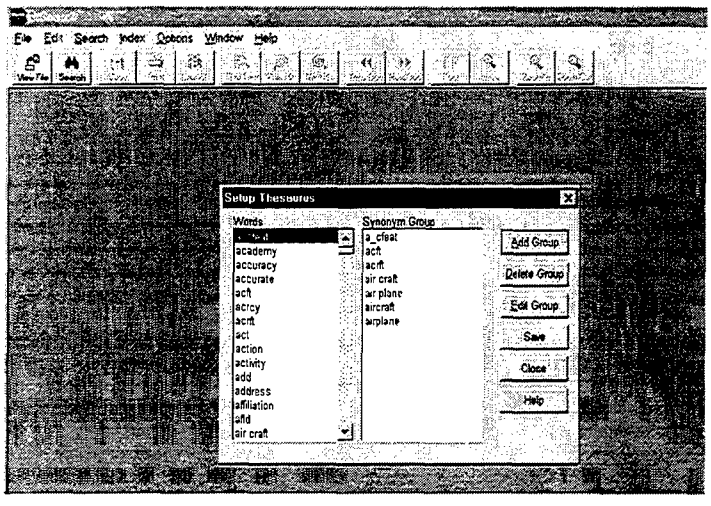

Fig. 1 dtSearch User Thesaurus Menu

accuracy , accurate , acrcy	level , lvl
action , act , msn , engagement	longitude , long
address , add , location	maximum , max
air space, aspace , airspc , airspace	measurement , msrmnt
airplane , aircraft , "air plane" , "air craft" , acrft , acft , a_cfeat	message , msg
airport , runway , airfield , airstrip , aport , afld , rnwy air trfc cntrl"	mil , military
allegence , coalition , affiliation	mile , mi
alternate , alt	minimum , min
altitude , alt , altd	name , nm , aka
amount , amt	num , number
angle , ang	object , obj , objct
association , assoc , assc	obs , observation , ob
atmosphere , atmos , atmps , atms	olay , overlay
battlefield , batfld , batlfld	on hand , oh
bio , biology , biological	operating , opng
bridge , brij	operation , oper , ops

capability , capa	operational , ready , available , readiness , opl
category , cat	order , command
change , modify , update	organization , org , e_org
channel, chnls , chnl	output , yield
char , varchar	person , per , pers
classify , restrict , classified , restricted , classification	point , pt , pnt
cloud , cld	population , pop
code , cd	priority , rank , relevent , pri
combat , cmbt , cbat	probability , prob
command , cmnd, cmd	production , prod
commander , cmndr , cmdr	profession , mos , msn , occ_spec_cd
condition , cndtn , cndn	qualification , background , education , qfn , training
confirm , validate , validated , ratify , corroborate	qualifier , qal
control , cntrl	quantity , quant , qty
coordinate , coord , coordin , corr , grid	radar , scan , image
critical , significant	radiation , rad
data , information	railway , rwy
date , dt , datetime , dttm	range , rng
datum , dttm , "origination point"	release , launch , fire , shoot , strike , engage
dead , died , death , kill , killed , casualty , missing	remark , comment , assessment , remarks , rmrks
depth , dpth , dpt	require , mandate , order
describe , descr , describtion , descrp , desc	resource , res
descriminator , dscr	right , rt
detail , dtl	route , rte
dock , port	school , academy
document , doc , report , observation , observ , obrep , message , detect	sector , sct , grid , zone
e_ , enemy , en_	segment , seg , portion , partial
effective , eff	sensor , snsr
element , elmnt	serial , ser
elevation , elev , elvat	signal , sig , signature
employment , emp , job , work , profession , skill_lvl	start , strt
enemy , en eorg , e_per , e_org , eper	status , stat , update
eqp , equipment , equip , material , mat , matrl , materiel , equipt	subject , sbjt , sbjct , sub
establishment , estb	surface , srfc
estimate , est	symb , symbol
evaluation , eval	system , sys
event evnt activity	target , trgt , tgrt , tgt
facility , fac , facl	task , tsk , objective , mission , ato , plan , assignment , order , ob
factory , depot , "manufacturing plant" , plant , warehouse	tech , technology
feature , feat	temp , temperature
feet , ft	temp , temporary
frequency , freq	text , txt
function , funct , functional , func	time , tm
group , grp	total , tot
height , ht	traffic , trfic
holding , hldng	type , typ , tp
identification , id , ident , idx , "call sign" , identifier	um , "unit of measure", dimension , dims , dim
image , photo , graphic , display , imagery , overlay , olay	unit , unt
index , indx	veh , vehicle
intelligence , intel , recon	vertical , vrt
interval , intrvl , cycle , period	vicinity , region , zone
item , itm	volts , vlts
kilometers , km	water craft , ship, boat , submarine , barge , platform
land , lnd	weapon , wpn , ord , muntin , muntn
latitude , lat	weight , wt
left , lft	width , wdth
length , lgth	

Table 1. Synonyms, abbreviations and acronyms used in JCDB and MIDB data dictionaries

My Clusters

Clustering is a process that groups or clusters like terms or terms that are associated with each other in clusters. Search for individual or corresponding elements are then done within the clusters to reduce the number of searches that must be conducted. It is similar to the Basic Concept Areas (BCA) that the DELTA process conducts.

By clustering the data into groups we increase the likelihood of finding equivalent or similar terms. We limited our search analysis to finding common attributes in the targeting, track and observations clusters. We based our restriction on the clusters we felt would achieve the highest matching results since both databases had data that represented each of these. I grouped the data attributes in __# of_ clusters. The clusters are displayed in Table_. concepts.
Entities: Airport, Bridge, Fac, Feature, Materiel, Organization, Person, seaport

According to SHADE data **Materiel** should contain data related to Supply and Transportation information relative to materiel assets (equipment, supplies, ammunition, fuel) important to the accomplishment of the DoD mission.

According to SHADE Common **Track Data** Store: Provides tables containing track identity, contact report data, and amplification data for several different track types such as platform, acoustic, and ELINT. Data in this segment is dynamic and typically provided by applications that interface to near real time track processing systems. This segment is a draft development segment and is still undergoing refinement.

Feature: Stores different types of features and their locations

Plan: Shows the development and management of a plan over time

Person: Identifies data about PERSONs of interest to the DoD, knowledge about whom is essential to the achievement of the DoD mission. It includes data on military and civilian PERSONs, including actual or potential friends and foes to the U.S.

Organization: Provides a hierarchical view of an organization

Reference Sets: Contains all the reference code tables, including domain values, for each JC2DS Segment

Facility: Identifies the data relative to physical structures consisting of buildings, warehouses, airport, docks, dams, power plants, bridges, utility systems and roadways. [MIDB AND (facility OR warehouse OR airport OR dam OR bridge OR railway OR tower OR tunnel OR center OR building OR bunker OR depot OR road OR dock OR port OR power plant or Traffic)]

Software (PL, Excel, dtSearch)

The three main COTS software packages that we used in our study were Personal Librarian Software, Microsoft Excel 97, and dtSearch. PL is a search software that allows one to conduct natural language searches as well as fuzzy searches. It also has a thesaurus feature. The software was used in the DELTA paper. A nice feature of PL is that it will rank the result in order for you. PL does however require you to reformat your data in order for it to be accepted by the software. The DELTA paper goes over the formatting process. The software is available for free from AOL.

dtSearch is another text search in software. It has some useful features which PL does not have. They include the Wordnet Thesaurus as well as the ability to accept a user defined thesaurus. It allows users to conduct natural language search, stemming, fuzzy searches, and boolean searches. It has a nice feature which lets you index or segment your file so you can search for specific items in certain portion of your file for example we could search for hits on the word aircraft only when it is in the definition meta data and not when it appears in the attribute name meta data. dtSearch is available for a free day 30 trial. Another nice feature of dtSearch is that it does not require special reformatting of your data in order to work. It accepts most file types such as txt, xls, doc, etc.

My Queries

Data Type Conflicts As was identified by Premerlani in his *Approach For Reverse Engineering of Relational Databases,* datatypes do not always match even when attribute names may match. A type of char in one database was equivalent to data type varchar in the other. Database specific issues or idiosyncrasies include: JCDB does not have any char types; it only has varchar. MIDB has both varchar and char. JCDB database only has 3 attributes with float type whereas MIDB has over a hundred. JCDB has multiple attributes of decimal type whereas MIDB has no decimal types. MIDB has multiple attributes of type tinyint; JCDB has no tinyint. MIDB has 9 smallint types whereas JCDB has over 400. JCDB has over 100 attributes of serial type whereas MIDB has none.

Data Type	MIDB	JCDB
Varchar	Multiple	Multiple
Char	Multiple	None
Float	Multiple	3
Decimal	None	Multiple
Serial	None	Multiple
Tinyint	Multiple	None
Smallint	9	Multiple
Integer	Multiple	Multiple
Numeric	Multiple	Multiple

Table , JCDB and MIDB data types

As a result some assumptions were made after several matches were identified. The assumptions are type Char in MIDB is equivalent to type varchar in JCDB since JCDB has no char data types. A float data type in MIDB is most likely equal to a decimal or numeric data type in JCDB. A tinyint data type in MIDB is similar to a smallint data type or it could be a integer data type. Since MIDB does not have any serial data types it was assumed that a JCDB serial data type is equivalent to a integer data type in MIDB. In my queries I have addressed this problem of data type conflicts by listing the sets in tables below as synonyms for each other. So whenever I query a datatype of type char it automatically also searches evaluates those documents that have type varchar.

MIDB	JCDB
varchar, char	varchar
float, numeric	decimal, numeric
integer	serial, integer
tinyint	smallint, integer

Most techniques such as query Flocks suggest starting out with as general a search query as possible and then gradually shrinking the query over a few iterations or filtering steps to locate a match. In our case we start a general query by entering the database (MIDB or JCDB) we are trying to find the equivalent attribute in as our first search component.

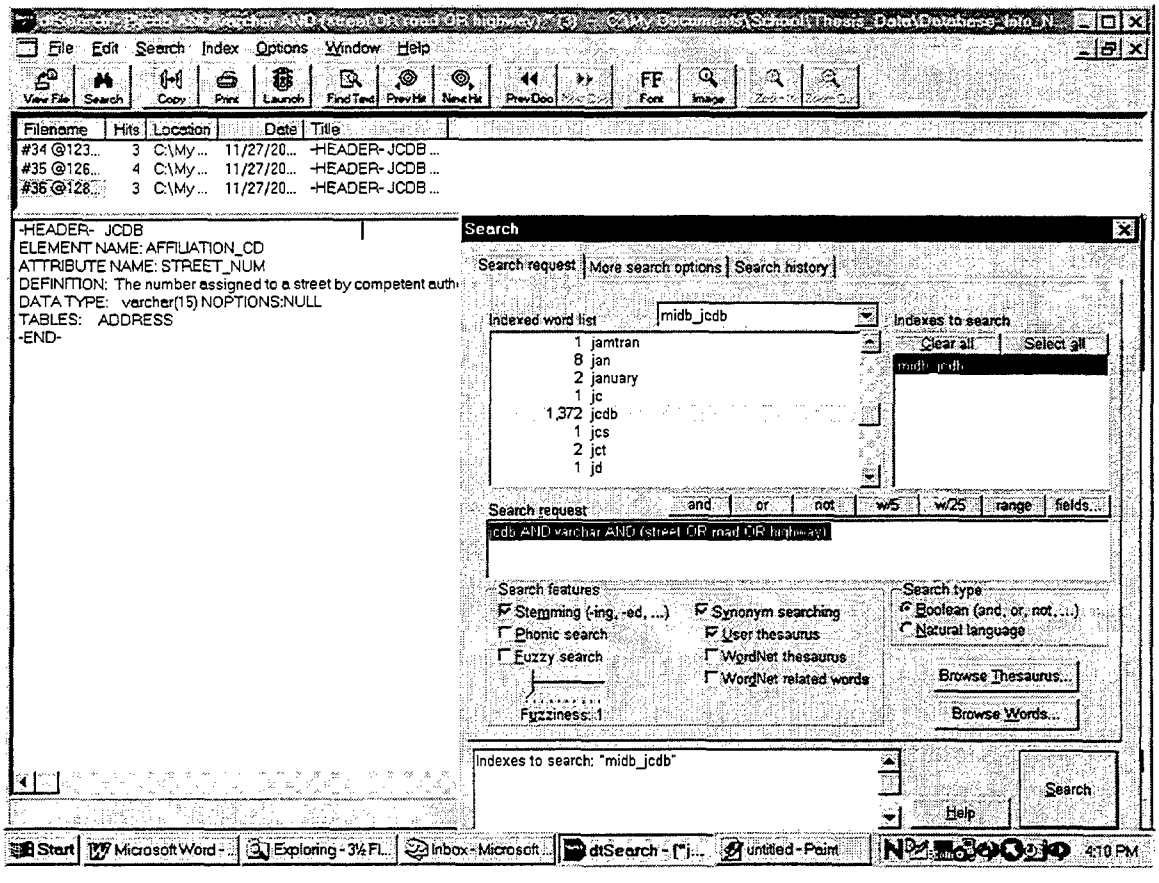

Fig. Search query.

After formatting the dictionaries into a common format and installing them into the PL Software and dtSearch software we followed the following process which encompasses portions from each of the techniques. The process was a dynamic process because we found that using just one of the above procedures did not always get the best results. Therefore we looked at the specific attribute data and developed a strategy based on the available data and prior experience.

Analyze the data elements in each dictionary to determine some natural common groups such as targeting, communication, enemy and organization. Concentrate on evaluating these portions of the data dictionaries first since there is a greater likelihood of finding common elements then in other portions of the database.

Field searching allows one to only display those results that are within a previously defined field. For example in our search we defined the attribute definition and table metadata as separate fields. This way we can ask it to conduct a search for a word facility only when it appears in the attribute definition field and not when it appears in the table field.

Searching strategy: Most sources recommended that searches should be carried out with first doing a very broad search than making it narrower and narrower. By starting off with a broad search one can reduce chances of missing relevant information/data.

My Results

Below are the attributes in the three cluster groups I am evaluating. The attributes were grouped in to clusters by conducting a search for all documents in MIDB with dtSearch that include the terms target, track , and observation. During this search I used the WordNet thesaurus as well as my own thesaurus which I defined earlier to ensure that I found not only the words target, track and observation but also all cases of their synonym as well as acronym or abbreviation that may represent them.

TARGET CLUSTER MIDB			
ACFT INTERVALOMETER UM	EFFECT_IDX_VALUE_UM	MIN_IMPACT_SPEED	TASKED_UNIT_NAME
ACFT MECHANIZATION	ELEVATION	MSN TYPE	TDI
ACFT MODE	ELEVATION ACC	MSN_CALLSIGN	TERMINAL IMPACT AZIMUTH
ACFT QTY	ELEVATION DERIV ACC	MSN_ID	TERMINAL_IMPACT_ANGLE
ACFT_ADD_FACTORS	ELEVATION MSL	MSN_NAME	TERMINAL_IMPACT_SPEED
ACFT_INTERVALOMETER	ELEVATION MSL CONF LVL	MSN_PRIMARY	TGT DTL NAME
ACFT_TYPE	ELEVATION_CONF_LVL	MSN_PRIMARY_SPECIALTY	TGT LIST STATUS
ACTIVITY	ELEVATION_DATUM	MSN_SECONDARY	TGT RESTR
AFFILIATION	ELEVATION_DERIV	MSN_SECONDARY_SPECIALTY	TGT RESTR REASON
AIR DEF AREA	ELEVATION_DERIV_ACC_UM	MSN_SUCCESS	TGT SUSCEPTIBILITY
AKA	ELEVATION_MSL_ACC	NO STRIKE	TGT SYS CODE
AKA_TYPE	ELEVATION_MSL_DERIV	OB_TYPE	TGT SYS NAME
ALERT	ELEVATION_MSL_DERIV_ACC	OBS CONDITION	TGT_DTL_AIMPT_WPN_SK
ALLEGIANCE	ELEVATION_MSL_DERIV_ACC_UM	OBS CONDITION SECONDARY	TGT_DTL_AIMPT_WPN_TIE_SK
ALTITUDE_UM	ELEVATION_MSL_UM	OBS LENGTH UM	TGT_DTL_AKA_SK
AMOUNT_UM	ELEVATION_UM	OBS WIDTH UM	TGT_DTL_ASSESS_SK
AREA EVAL	EMITTER_HEIGHT_UM	OPEN STG UM	TGT_DTL_NAME
AREA UM	ENTRY_WIDTH_EVAL	OPEN_STG_UM	TGT_DTL_SK
ASSESS DATETIME	ENTRY_WIDTH_UM	OPER_STATUS	TGT_DTL_TIE_SK

ASSESS_TYPE	EQP CODE	OPERATION_NAME	TGT_LIST_NAME
ASSOC	ERROR PROB CIRCULAR UM	OUTPUT EVAL	TGT_LIST_NUM
ASSOC BEGIN DATE	ERROR PROB DEFLECTION	OUTPUT UM	TGT_LIST_ORIGINATOR
ASSOC_END_DATE	ERROR PROB DEFLECTION UM	PASSES_AVAIL_QTY	TGT_LIST_SK
ATO_ACFT_TYPE	ERROR PROB RANGE	PASSES_QTY	TGT_LIST_STATUS
ATTACK ANGLE	ERROR_PLANE	PEN EQ THICKNESS UM	TGT_LIST_TIE_ORDER_SK
ATTACK_ANGLE	ERROR_PROB_CIRCULAR	PERCENT DAMAGED	TGT_LIST_TIE_ORDER_TIE_SK
AZIMUTH	ERROR_PROB_CIRCULAR	PERCENT DESTROYED	TGT_LIST_TYPE
AZIMUTH_REF	ERROR_PROB_HIT	PERCENT RECUP	TGT_MSN_SK
BLOCK INTRVL UM	ERROR_PROB_NEAR_MISS	PHOTO_DATE	TGT_MSN_TIE_SK
CAPACITY EVAL	ERROR_PROB_RANGE_UM	POL_SUBDIV	TGT_OBJ_AKA_SK
CAPACITY UM	ERROR_PROB_RANGE_UM	PRESSURE UM	TGT_OBJ_NAME
CASE_NUM	ERROR_RANGE_BIAS	PRIORITY TGT	TGT_OBJ_SK
CATEGORY_NAME	ERROR_STRESS_LVL	PRIORITY TGT EXTERNAL	TGT_OBJ_TIE_SK
CATEGORY_REF	ERROR_SWITCH_SET	PRIORITY TGT PREVIOUS	TGT_RADIUS
CC	ERROR_TGT_CLASS	PRIORITY_LIST	TGT_RESTR
CHNL QTY EVAL	ERROR_TYPE	PRIORITY_OBJ	TGT_RESTR_REASON
CHNL_QTY_EVAL	EVAL	PRIORITY_TASK	TGT_SUSCEPTIBILITY
CMD_CNTL_COMM	EXECUTION_DATE	PROB DAMAGE TOTAL	TGT_SYS_ASSESS_SK
COLLATERAL DAMAGE	EXECUTION_DAY	PROB_DAMAGE_SORTIE	TGT_SYS_CODE
COMBAT EFFECTIVENESS	EXTERNAL_TGT_SYS_ID	QTY OH EVAL	TGT_SYS_EQP_SK
COMBAT STRENGTH	FLOOR SPACE UM	QTY PA EVAL	TGT_SYS_FAC_SK
CONDITION	FLOOR_SPACE_EVAL	QTY WA EVAL	TGT_SYS_SK
CONDITION AVAIL	FPA	QTY_EVAL	TGT_SYS_TIE_SK
COORD	FREQ UM	RADIAL_G_QTY	TGT_SYS_TYPE
COORD BASIS	FUZE_ARM_TIME	RADIUS	TGT_SYS_UNIT_SK
COORD DATETIME	FUZE_DELAY_TIME	RADIUS UM	THICKNESS_UM
COORD DATUM	FUZE_MODE	RECCE RQD	TIE_BOOL
COORD DERIV	FUZE_NAME	RECUP INTRVL UM	TIE_FROM_SK
COORD DERIV ACC	FUZE_SETTING_ALTITUDE	RECUP_INTRVL	TIE_TO_ENTITY
COORD ROA UM	FUZE_SETTING_TIME	RECUP_INTRVL_MAX	TIE_TO_SK
COORD_DERIV_ACC_UM	GEODETIC_PROD	RELEASE ALTITUDE	TOT DATETIME
COORD_ROA	GEOIDAL MSL SEPARATION UM	RELEASE_ANGLE	TOT DATETIME EST
COORD_ROA_CONF_LVL	GEOIDAL_MSL_SEPARATION	RELEASE_MANEUVER	TOT_DATETIME
COVERED_PERCENT	GRAPHIC SERIES	RELEASE_VELOCITY	TOT_DATETIME_EST
CURRENT_SPEED_UM	GRAPHIC_AGENCY	REQUEST	TRAIT EVAL
DAMAGE_CRITERION	GRAPHIC_CC	RMK TYPE	TRAIT EVAL
DEPTH_EVAL	GRAPHIC_ED_DATE	ROCK JOINT SPACING UM	TURN BASIN DEPTH UM
DEPTH_UM	GRAPHIC_ED_NUM	RWY CUT QTY	TURN BASIN DEPTH UM
DESCR_VALUE_UM	GRAPHIC_SCALE	RWY MIN CLEAR LENGTH	TURN BASIN DIAMETER UM
DESIGN_LOAD_UM	GRAPHIC_SHEET	RWY MIN CLEAR WIDTH	TURN_BASIN_DIAMETER
DIAMETER_EVAL	GUN_FIRE_RATE	RWY_CUT_CRATERS_QTY	USEABLE LENGTH UM
DIAMETER_UM	HARDNESS	RWY_OVERRUN_UM	USEABLE_LENGTH_UM
DIGITAL_DATA_RATE_UM	HEIGHT	SCL CODE	UTM
DISPNSR ALTITUDE	HEIGHT EVAL	SEMI UM	VEGETATION_HEIGHT_UM
DISPNSR PAT DIMENSION	HEIGHT UM	SHAPE	VEHICLE_INTRVL_UM
DISPNSR PAT LENGTH	ILAT	SHOULDER CONDITION	VERTICAL CLEARANCE EVAL
DISPNSR PAT RADIUS	ILLUMINATION RATE	SHOULDER WIDTH UM	VERTICAL ORIENT
DISPNSR PAT TYPE	ILON	SLANT_RANGE	VERTICAL_CLEARANCE_UM
DISPNSR PAT WIDTH	JMEM TYPE	SPAN LENGTH UM	WAC
DISPNSR SPIN RATE	LENGTH	SPEED STD DEV	WATERBODY
DISPNSR_PAT_AZIMUTH	LENGTH EVAL	SPEED_UM	WIDTH
DISTANCE_UM	LENGTH UM	STD_SECTION_LENGTH_UM	WIDTH_EVAL
DIVE ANGLE AT DISPENSE	LINE WIDTH UM	STRENGTH UM	WIDTH_UM
DMPI IMPACT ANGLE	LOC EIGHT HOUR	SWELL UM	WPN AZIMUTH AT DISPENSE
DMPI_ID	LOC FOUR HOUR	SYMBOL_CODE	WPN MULTIPLE
DOC_STATUS	LOC_NAME	TASK_ORDER_DTG	WPN NAME
DOC_TYPE	MATERIAL DEPTH EVAL	TASK_ORDER_DTG_BEGIN	WPN PAT LENGTH
ECHELON	MATERIAL DEPTH UM	TASK_ORDER_DTG_END	WPN PAT WIDTH
ECM_TECHNIQUE	MIL_AREA	TASK_ORDER_ID	WPN QTY
EFFECT IDX VALUE	MIL_GRID	TASK_ORDER_ORIGINATOR	WPN_CPD
EFFECT_IDX_TYPE	MIL_GRID_SYS	TASK_ORDER_TYPE	

TRACK CLUSTER MIDB			
ACTIVITY_DESCR	DESTINATION_NAME	MIL_GRID	SCAN_HI
AFFILIATION	DESTINATION_SYMBOL_CODE	MIL_GRID_SYS	SCAN_ITEMS
AIR_DEF_AREA	ECHELON	MSN SECONDARY	SCAN_LO
AKA	ECM_TECHNIQUE	MSN SECONDARY SPECIALTY	SCAN_MEAN
AKA_TYPE	ELEVATION	MSN_PRIMARY	SCAN_STD_DEV
ALERT	ELEVATION_ACC	MSN_PRIMARY_SPECIALTY	SCAN_SUM_W
ALLEGIANCE	ELEVATION_CONF_LVL	NET_LINK_TYPE_SPECIFIC	SCAN_SUM_W_OBS
ALTITUDE	ELEVATION_DATUM	OPER_STATUS	SCAN_SUM_W_OBS

ALTITUDE_UM	ELEVATION_DERIV	OSUFFIX_REF	SCAN_SUM_W_SQ
ANNEX_TYPE	ELEVATION_DERIV_ACC	PGRI MEAN	SCONUM
AOU_CONTAINMENT	ELEVATION_DERIV_ACC_UM	PGRI_HI	SEMI_MAJOR
AOU_LOB_ERROR	ELEVATION_MSL	PGRI_ITEMS	SEMI_MINOR
AOU_TYPE	ELEVATION_MSL_ACC	PGRI_LO	SEMI_UM
ASSOC	ELEVATION_MSL_CONF_LVL	PGRI_STD_DEV	SHIP_CLASS_NAME
ASSOC_BEGIN_DATE	ELEVATION_MSL_DERIV	PIN	SHIP_TRADEMARK
ASSOC_END_DATE	ELEVATION_MSL_DERIV_ACC	POL_SUBDIV	SHIP_TYPE
AZIMUTH	ELEVATION_MSL_DERIV_ACC_UM	PRF_HI	SOURCE_DIGRAPH_FIRST
AZIMUTH_REF	ELEVATION_MSL_UM	PRF_ITEMS	SOURCE_DIGRAPH_LAST
BE_NUMBER_REF	ELEVATION_UM	PRF_LO	SPEED
BLOCK_INTRVL	ELNOT	PRF_STD_DEV	SPEED_UM
BLOCK_INTRVL_MAX	EMITTER_MODE	PRI_HI	SYMBOL_CODE
CALLSIGN	EQP_CODE_REF	PRI_ITEMS	TEMPLATE_FLAG
CATEGORY_REF	EXTERNAL_ID	PRI_LO	THREAT
CC	EXTERNAL_ID_PREV	PRI_MEAN	TIE_BOOL
CONTACT_QTY	EXTERNAL_TGT_SYS_ID	PRI_STD_DEV	TIE_FROM_SK
COORD	FORCE	PRI_SUM_W	TIE_PROB
COORD DATUM	GEOIDAL_MSL_SEPARATION	PRI_SUM_W_OBS	TIE_TO_ENTITY
COORD DERIV	GEOIDAL_MSL_SEPARATION_UM	PRI_SUM_W_SQ	TIE_TO_SK
COORD_BASIS	GRAPHIC_AGENCY	PRIORITY_TGT_PREVIOUS	TRACK_AKA_SK
COORD_DATETIME	GRAPHIC_CC	PULSE DURATION MEAN	TRACK_ELINT_MODE_SK
COORD_DERIV_ACC	GRAPHIC_ED_DATE	PULSE_DURATION_HI	TRACK_LOC_SK
COORD_DERIV_ACC_UM	GRAPHIC_ED_NUM	PULSE_DURATION_ITEMS	TRACK_NAME
COORD_ROA	GRAPHIC_SCALE	PULSE_DURATION_LO	TRACK_SK
COORD_ROA_CONF_LVL	GRAPHIC_SERIES	PULSE_DURATION_STD_DEV	TRACK_TIE_SK
COORD_ROA_UM	GRAPHIC_SHEET	RF_HI	TRACK_TIE_STAT_SK
COURSE	ILAT	RF_ITEMS	TRACK_TYPE
COURSE_REF	ILON	RF_LO	UNIT_ID_REF
DATETIME_LAST_OBS	LAND_TYPE	RF_MEAN	UTM
DELETE_POINTER	LOC_NAME	RF_SUM_W	WAC
DESTINATION_COORD	LOC_REASON	RF_SUM_W_OBS	WATERBODY
DESTINATION_DATETIME	MIL_AREA	RF_SUM_W_SQ	

OBSERVATION CLUSTER MIDB			
ACCESS	DELETE_POINTER	IDENT_SCORE	PIN_OVRWRT
AFFILIATION	DESTINATION_COORD	ILAT	POL_SUBDIV
AIR_DEF_AREA	DESTINATION_DATETIME	ILLUMINATION_RATE	POLARIZATION
ALERT	DESTINATION_NAME	ILLUMINATION_RATE_STD_DEV	PRI_ACTIVITY_CODE
ALLEGIANCE	DESTINATION_SYMBOL_CODE	ILON	PRI_BASE
ALTITUDE_STD_DEV	DURATION	LOAD_CLASS_EVAL	PRI_CALCULATED
AOU_CONTAINMENT	ELEVATION	LOC_NAME	PRI_LEG_QTY
AOU_LOB_ERROR	ELEVATION_ACC	MHS_NUM	PRI_LEG_QTY
AOU_TYPE	ELEVATION_CONF_LVL	MIL_AREA	PRI_SUM_W_OBS
ASSESS_DATETIME	ELEVATION_DATUM	MIL_GRID	PRI_TYPE
AZIMUTH	ELEVATION_DERIV	MIL_GRID_SYS	PULSE_AMPLITUDE
AZIMUTH_REF	ELEVATION_DERIV_ACC	MODULATION_EPL	PULSE_DURATION
BEAM_WIDTH	ELEVATION_DERIV_ACC_UM	MSG_DTG	PULSE_DURATION_STD_DEV
BURST_STD_DEV	ELEVATION_MSL	MSG_NUM	PULSE_QTY
CASE_NOTATION	ELEVATION_MSL_ACC	MSG_ORIGIN	PULSE_STD_DEV
CC	ELEVATION_MSL_CONF_LVL	MSG_PRECEDENCE	RE_IDENT_FAIL
CC_OVRWRT	ELEVATION_MSL_DERIV	MSG_SECTION_NUM	RF_AGILITY_FLAG
CLUSTER_ID	ELEVATION_MSL_DERIV_ACC	MSG_TYPE	RF_CODE_LIMIT
COLL_COORD	ELEVATION_MSL_DERIV_ACC_UM	MSG_UPDATE_NUM	RF_OPER_MODE
COLL_ILAT	ELEVATION_MSL_UM	MSN_NAME	RF_STD_DEV
COLL_ILON	ELEVATION_UM	OB_ASSOC_PRIMARY	RF_SUM_W_OBS
COLL_PROJECT_ID	ELNOT_CHANGE	OB_ASSOC_SECONDARY	RF_TYPE
COLL_SYMBOL_CODE	ELNOT_CONF	OB_TYPE	SCAN
COLL_WEIGHT	ELNOT_CONF_ORIGINAL	OBS_AKA_SK	SCAN_STD_DEV
CONTACT_QTY	ELNOT_ORIGINAL	OBS_COMM_SITE_SK	SCAN_SUM_W_OBS
COORD	ELNOT_RE_IDENT	OBS_CONDITION	SEMI_MAJOR
COORD_BASIS	EMITTER_ID	OBS_CONDITION_SECONDARY	SEMI_MINOR
COORD_DATETIME	EMITTER_NAME	OBS_DATETIME	SEMI_UM
COORD_DATUM	EMITTER_NATO_NAME	OBS_ELINT_PAR_SK	SIG
COORD_DERIV	EXTERNAL_ID	OBS_ELNOT_SK	SIG_MODE
COORD_DERIV_ACC	EXTERNAL_ID_PREV	OBS_LENGTH	SITE_TYPE
COORD_DERIV_ACC_UM	EXTERNAL_RMK_ID	OBS_LENGTH_UM	SOURCE_DIGRAPH
COORD_ROA	EXTERNAL_RMK_QTY	OBS_NAME	SOURCE_NAME
COORD_ROA_CONF_LVL	GEOIDAL_MSL_SEPARATION	OBS_PAR_SK	SOURCE_TRIGRAPH
COORD_ROA_UM	GEOIDAL_MSL_SEPARATION_UM	OBS_REPORT_SK	SYMBOL_CODE
CORR_DATETIME	GRAPHIC_AGENCY	OBS_SK	TIE_PROB
CORR_OVRWRT	GRAPHIC_CC	OBS_TIE_SK	TIE_TO_ENTITY

CORR_OVRWRT	GRAPHIC_ED_DATE	OBS_TIE_STAT_SK	UTM
CORR_STEP	GRAPHIC_ED_NUM	OBS_WIDTH	VERIF_FIX_NAME
CORR_STEP	GRAPHIC_SCALE	OBS_WIDTH_UM	WAC
COVERED_PERCENT	GRAPHIC_SERIES	PERIODICITY	WATERBODY
DATETIME_LAST_OBS	GRAPHIC_SHEET	PGRI	
DEGREE_INTEREST	ICON_CODE	PIN	

Conclusion and Future Effort

TBD

LIST OF REFERENCES

[ABI00] Abiteboul, S., Buneman, P., and Suciu, D., *Data on the Web: From Relations to Semistructured Data and XML.* Morgan Kaufmann Publishers, 2000.

[LU99] Lu, Zhihong, "Scalable Distributed Architectures for Information Retrieval," University of Massachusetts, Amherst, 1999.

[RIL96] Riloff, E. and Hollar, L. "Text Databases and Information Retrieval," Department of Computer Science, University of Utah, 1996

[RAG97] Raghavan, P. "Informational Retrieval Algorithms: A Survey," Proceedings of Eight Annual ACM-SIAM Symposium on Discrete Algorithms

[LEA00] Leah, L., Connel, M., and Callan, J. "Collection, Selection and Results Merging with Topically Organized U.S. Patents and TREC Data," Univeristy of Massachusetts

[HER95] Hernandez, M. and Stolfo, S., "The Merge/Purge Problem for Large Databases," Department of Computer Science, Columbia University

[TSU98] Tsur, D., Ullman, J., Abiteoul, S., Clifton, C., Motwani, R., Nestrov, S., Rosenthal. A., "Query Flocks: A Generalization of Association-Rule Mining,"

[BOU99] Bourret, R., "XML and Databases," Technical University of Darmstadt, http://www.informatik.tu-darmstadt.de/DVS1/staff/bourret/xml/XMLAndDatabases.htm, September 1999.

[BOX00] Box, D., et al., "SOAP: Simple Object Access Protocol," http://msdn.microsoft.com/xml/general/soapspec.asp, March, 2000.

[BRA98] Bray, T., "The Annotated XML Specification," http://www.xml.com/axml/axml.html, September 1998.

[BUC00] Buck, L., "Modeling Relational Data in XML," Extensibility, Inc., http://www.extensibility.com/xml_resources/modeling.htm, March 2000.

[CAR00] Carnevale, R., "The Joint Common Database," February 2000.

[DAT00] IBM, "Datajoiner," document available online at http://www.software.ibm.com/data/datajoiner.

[DAV99] David, M., "SQL-based XML Structured Data Access," Web Techniques, San Francisco, June 1999.

[DOM00] Wood, L., et al., "Document Object Model (DOM) Level 1 Specification," W3C Working Draft, http://www.w3.org/TR/2000/WD-DOM-Level-1-20000929/, September 2000.

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, Code 52 2
   Naval Postgraduate School
   Monterey, CA 93943-5100

3. Research Office, Code 09 1
   Naval Postgraduate School
   Monterey, CA 93943-5000

4. Mr. Mike Francher 5
   Joint C4ISR Battle Center
   116 Lake View Parkway, Suite 150
   Suffolk, VA 23435-2697

5. Dr. Luqi, CS/Lq 5
   Software Engineering Automation Center
   Naval Postgraduate School
   Monterey, CA 93943