

AUTOMATING LABORATORY OPERATIONS BY INTEGRATING
LABORATORY INFORMATION MANAGEMENT SYSTEMS (LIMS) WITH
ANALYTICAL INSTRUMENTS AND SCIENTIFIC DATA MANAGEMENT
SYSTEM (SDMS)

Jianyong Zhu

Submitted to the faculty of the University Graduate School

in partial fulfillment of the requirements

for the degree

Master of Science

in the School of Informatics,

Indiana University

June 2005

Master's Committee

Mahesh Merchant, Ph.D

Douglas Perry, Ph.D

Michael Evans, Ph.D

Accepted by the Graduate Faculty, Indiana University,
in partial fulfillment of the requirements for the degree of Master's of Science in
Laboratory Informatics Graduate Program

ACKNOWLEDGEMENTS

During my two-year academic career I am grateful to many people who helped me in my research and studies. Firstly most important of all I would like to thank Dr. Mahesh Merchant and Dr. Douglas Perry for their guidance, encouragement, support and unconditional faith in my abilities. I would also like to thank Dr. Michael Evans for giving me the opportunity to intern in his laboratory and for his generosity of letting me use his analytical instruments, software and hardware. In the end I want to thank my friends and family, particularly my wife, Ping, without their endurance and relentless support it would have been impossible for me to coordinate my studying and working schedules and fulfill my responsibilities.

ABSTRACT

The large volume of data generated by commercial and research laboratories, along with requirements mandated by regulatory agencies, have forced companies to use laboratory information management systems (LIMS) to improve efficiencies in tracking, managing samples, and precisely reporting test results. However, most general purpose LIMS do not provide an interface to automatically collect data from analytical instruments to store in a database. A scientific data management system (SDMS) provides a “Print-to-Database” technology, which facilitates the entry of reports generated by instruments directly into the SDMS database as Windows enhanced metafiles thus to minimize data entry errors. Unfortunately, SDMS does not allow performing further analysis. Many LIMS vendors provide plug-ins for single instrument but none of them provides a general purpose interface to extract the data from SDMS and store in LIMS.

In this project, a general purpose middle layer named LabTechie is designed, built and tested for seamless integration between instruments, SDMS and LIMS. This project was conducted at American Institute of Technology (AIT) Laboratories, an analytical laboratory that specializes in trace chemical measurement of biological fluids. Data is generated from 20 analytical instruments, including gas chromatography/mass spectrometer (GC/MS), high performance liquid chromatography (HPLC), and liquid chromatography/mass spectrometer (LC/MS), and currently stored in NuGenesis SDMS

(Waters, Milford, MA). This approach can be easily expanded to include additional instruments.

TABLE OF CONTENTS

ABSTRACT.....	IV
LIST OF FIGURES	VIII
LIST OF TABLES	X
LIST OF ABBREVIATIONS	XI
1. INTRODUCTION	1
1.1. Laboratory Information Management Systems (LIMS)	2
1.2. Analytical instruments	5
1.3. Chromatography Data Systems (CDS)	5
1.4. Scientific Data Management System (SDMS)	8
1.5. LabTechie	10
2. BACKGROUND	11
2.1. History and related research.....	11
2.2. Current practice and understanding	16
2.3. Intended project	17
3. METHODS	18
3.1. Materials and instruments	18
3.1.1. Networking	18
3.2. Data subjects	21
3.2.1. Data subject clearance.....	21
3.2.2. Data size.....	21
3.3. Procedures and interventions	21
3.4. Project analysis	28
3.5. Expected results	28
4. RESULTS	30
4.1. General results	30
4.1.1. User and project management in the middle layer.....	31
4.1.2. Collect and review instrument data in the middle layer	42
4.1.3. Storing data into LIMS database.....	43

4.2.	Important highlights.....	45
4.2.1.	Analytical instrument data collection	45
4.2.2.	Data file parsing.....	46
4.2.3.	Development of a secure XML format of instrument data files	51
4.2.4.	Storage of XML format of data in the archiving database.....	53
4.3.	Performance	65
4.3.1.	Parsing and processing XML files.....	65
4.4.	Summary	67
5.	CONCLUSION	68
5.1.	Overview of significant findings	68
5.1.1.	Parsing ASCII delimited file to XML file on Microsoft .Net platform	68
5.1.2.	Storing of secure XML data in Oracle 10g XML DB	69
5.2.	Consideration of findings in context of current knowledge.....	70
6.	DISCUSSION	71
6.1.	Limitations of the study	71
6.1.1.	Limitation of XML parsing and transforming	71
6.1.2.	Limitation of knowledge on XML storage	72
6.1.3.	Lack of dedicated computer systems	73
6.2.	Recommendations for further research.....	73
6.2.1.	Design and manage XML format of instrument data using a popular interchange format	73
	REFERENCES.....	76

LIST OF FIGURES

Figure	Page
Figure 1.1: LIMS and laboratory operations.....	4
Figure 1.2: Agilent 6890 GC/MS.....	6
Figure 1.3: A chromatography data system (CDS).....	7
Figure 1.4: A scientific data management system (SDMS).....	9
Figure 2.1: LIMS and instruments integration using drivers.....	12
Figure 3.1: Project data flow diagram.....	24
Figure 4.1: Login window.....	32
Figure 4.2: User settings	33
Figure 4.3: Project settings	34
Figure 4.4: Project settings – instruments.....	35
Figure 4.5: Project settings – project instruments usage.....	36
Figure 4.6: Project settings summary report.....	37
Figure 4.7: Project settings summary report – instrument usage.....	38
Figure 4.8: Project settings summary report – project setting	39
Figure 4.9: Project settings summary report – instrument setting	40
Figure 4.10: Project settings summary report PDF file export	41
Figure 4.11: Data retrieval from instruments.....	42
Figure 4.12: Review instrument Data	43
Figure 4.13: Submit data to LIMS database after reviewing	44
Figure 4.14: Search data by sample id.....	45

Figure 4.15: A comma-delimited ACSII instrument data file	46
Figure 4.16: An XML format of original instrument data file.....	47
Figure 4.17: An XSLT file to parse instrument sample data	49
Figure 4.18: An XML format of instrument sample data file.....	49
Figure 4.19: An XSLT file to parse instrument raw data file	50
Figure 4.20: An XML format of instrument raw data file	51
Figure 4.21: An XML format of instrument raw data file	52
Figure 4.22: A secured XML format of instrument data file.....	52
Figure 4.23: Storing XML data summary	55
Figure 4.24: A secured XML format of instrument data file.....	59
Figure 4.25: Complextype and annotations used in schema definition	62
Figure 4.26: Querying XML DB using XPath.....	65

LIST OF TABLES

Table	Page
Table 1.1: Features of LIMS functions	3

LIST OF ABBREVIATIONS

AIT	American Institute of Toxicology, a contract laboratory based in Indianapolis, Indiana.
ASCII	American Standard Code for Information Interchange. ASCII specifies a correspondence between digital bit patterns and the symbols/glyphs of a written language, thus allowing digital devices to communicate with each other and to process, store, and communicate character-oriented information.
AnDI	Analytical Data Interchange
JCAMP	Joint Committee on Atomic and Molecular Physical Data
BLOB	Binary Large Objects
CDS	Chromatography Data System
CLOB	Character Large Objects
DBMS	Database Management System
Delimited File	File using specific characters (delimiters), such as comma, tab, vertical bar (also referred to as pipe) and space, to separate the data.
Document Fidelity	Store XML while maintaining complete fidelity to the original
DOM	Document Object Model (DOM) is a form of representation of structured documents as an object-oriented model. DOM is the official World Wide Web Consortium (W3C) standard for representing structured documents in a platform- and language-neutral manner.
GC	Gas Chromatography
GC/MS	Gas Chromatography/Mass Spectrometry
HPLC	High Performance Liquid Chromatography
JCAMP	Joint Committee on Atomic and Molecular Physical Data
LAN	Local Area Network
LC/MS	Liquid Chromatography/Mass Spectrometry
LIMS	Laboratory Information Management Systems
RDBMS	Relational Database Management System
SDMS	Scientific Data Management System
SPC	Thermo Galactic SPC
TCP/IP	Transmission Control Protocol/Internet Protocol
XML	Extensible Markup Language
XMLSec	XML Security Library. XML security implementation.
XMLType	The Oracle database native structured XML storage is a shredded decomposition of XML into underlying object-relational structures (automatically created and managed by Oracle) for better SQL queriability.

XML complexType	XML Schema type containing structure of an element.
XML DB	Oracle XML DB is a set of Oracle DBMS built-in high-performance XML storage and retrieval technologies conforming to World Wide Web Consortium (W3C) XML data model.
XML Schema Definition	An instance of XML Schema. Defines a type of XML document in terms of constraints upon what elements and attributes may appear, their relationship to each other, what types of data may be in them, and other things.
XML Schema-based Objects	These objects are stored in Oracle XML DB as LOBs or in structured storage (object-relationally) in tables, columns, or views.
XML Schema-based validation	Ensures that a XML document's structure complies (is "valid" against) with specific XML Schema
XML simpleType	XML Schema type containing the value of an element or an attribute
XPath	XML Path Language. XPath makes it possible to “Cherry-Pick” individual components of an XML document.
XQuery	XML Query Language
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformation
W3C	World Wide Web Consortium
WAN	Wide Area Network

1. INTRODUCTION

Technological advances in biology and chemistry have made it possible for laboratories to generate unprecedented amounts of data. DNA sequencing, for example, has seen an increase in throughput of over 400-fold in recent years [1]. The large volume of data generated by commercial and research laboratories, along with requirements mandated by regulatory agencies, has forced companies to use laboratory information management systems (LIMS) to improve efficiency in tracking and analyzing samples and reporting test results and facilitate regulatory compliance. However, most general purpose LIMS do not provide an interface to automatically collect data from an analytical instrument to store in a database. Data are still needed to be transferred manually between instruments and LIMS, which results in the increasing need for integrating instruments and LIMS.

In this project, a generic middle layer was created between LIMS and various analytical instruments. The project was carried out at two locations: AIT Laboratories, Inc., a commercial analytical laboratory that specializes in trace chemical measurement of biological fluids, and the LIMS Laboratory of the Indiana University School of Informatics, located on the campus of Indiana University Purdue University Indianapolis. Sample data are generated from analytical instruments including gas chromatography/mass spectrometer (GC/MS), high pressure liquid chromatography (HPLC) and liquid chromatography/mass spectrometer (LC/MS) in the commercial analytical laboratory. The middle layer, which was designed, built, and tested for these

instruments, allows seamless integration of the instruments and LIMS. By using this middle layer, manual data entry from the instruments into LIMS is eliminated. In the meanwhile, security and integrity are maintained to meet regulatory requirements from U.S. Food and Drug Administration (FDA) [7].

1.1. Laboratory Information Management Systems (LIMS)

LIMS are collections of software, communication devices, and computers that acquire, store, analyze, and present data and information on laboratory samples and their processing [2]. LIMS are used to coordinate workflow and the movement of samples and information through different laboratory processes. These systems centralize data storage, automate data analysis, and provide quality assurance reports for process monitoring. The central component of a modern LIMS is a relational database management system (RDBMS) running on a computer with one or more software interfaces allowing users to enter, view, and process data.

LIMS usually have four functional areas: data and information capture, data analysis and reports, laboratory management, and system management. The detailed function and features are listed in Table 1.1 [3].

The middle layer built in the project is involved in the first two functions: data and information capture, and data analysis and reports. The relationship between these two LIMS functions and the laboratory operations is further illustrated in Figure 1.1. The middle layer is intended to automate the operation of data entry.

Function	Features
Data and information capture	data entry; file transfers and simple barcode entries; communication with laboratory devices such as data collection instruments or robotic devices
Data analysis and reports	Perform calculations, result verification, data analysis with integrated analytical procedures that link different types of experimental data or integrated external software systems, reports notification system
Laboratory management	Workflow scheduling and monitoring; inventory, sample storage, and tracking systems, decision-making process, revenue and costs tracking, and multi-site project management
System management	Disk backup and recovery, system performance tuning, links to external communications

Table 1.1: Features of LIMS functions

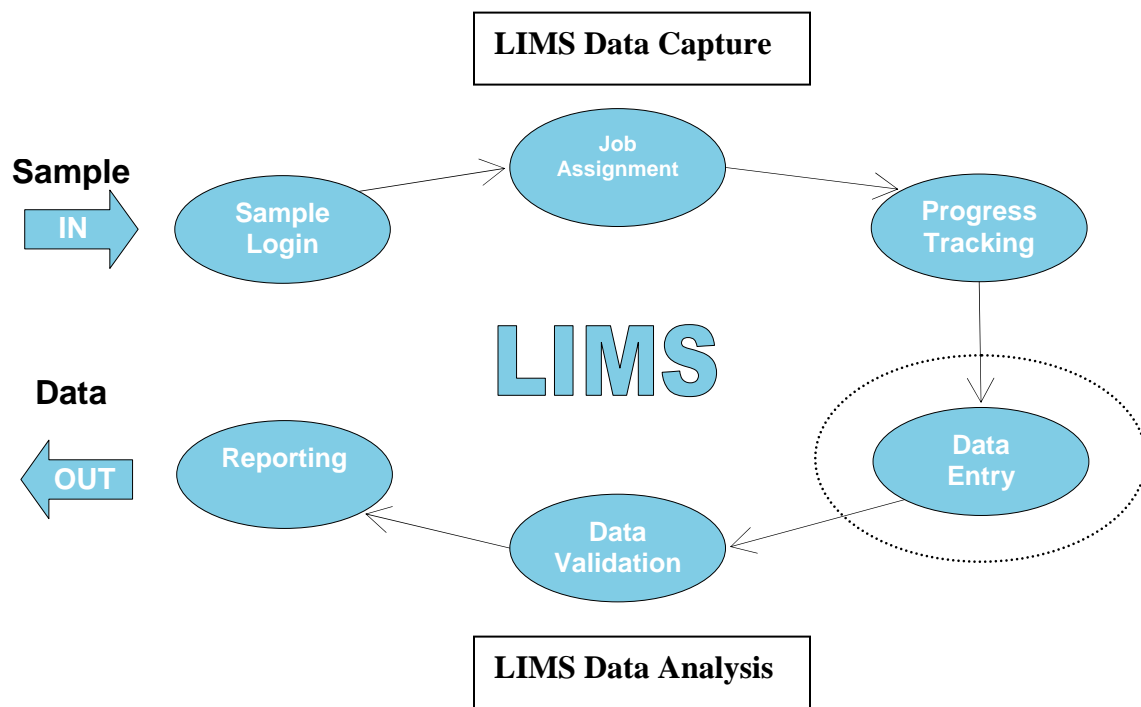


Figure 1.1: LIMS and laboratory operations

1.2. Analytical instruments

An analytical instrument is defined as laboratory equipment that analyzes samples and provides the information of samples. Figure 1.2 shows a GC/MS (Agilent, Palo Alto, CA). GC/MS is used to measure organic compounds in liquid or gas samples.

1.3. Chromatography Data Systems (CDS)

A CDS has a number of functions that it can perform. These are dependent on the use of the system by the laboratory and the nature of the chromatographic equipment used (Figure 1.3).

In general, the process used by most CDS consists of all or most of the steps outlined below [4]:

- 1) Set up the method and analytical run information.
- 2) Instrument control.
- 3) Acquire data from each injection, together with injection number from the auto-sampler and any chromatographic conditions.
- 4) Process the acquired data first into peak areas or heights and then into analyze amounts or concentrations.
- 5) Store the resultant data files and other information acquired during the run for reanalysis.
- 6) Interface with other data or information systems for import of data relating to CDS set-up or export of data for further processing or collation of results.



Figure 1.2: Agilent 6890 GC/MS



Figure 1.3: A chromatography data system (CDS)

1.4. Scientific Data Management System (SDMS)

A scientific data management system (SDMS) is used to collect, organize, index, store, archive, search, and share electronic records. It provides a secure, central repository, and rich content services to allow organizations to manage and re-use business critical information, comply with regulatory and corporate mandates, and enable collaboration for any type of electronic record. Eventually, SDMS improves knowledge worker productivity, facilitates compliance, reduces operational costs, and helps make better decisions to gain an edge on the competition [5]. An example of SDMS is shown in Figure 1.4.

SDMS have some features as follows:

- 1) Manage both raw binary data and any type of human-readable file.
- 2) Collect record manually or automatically.
- 3) Provide filters for viewing records in varied formats such as chromatogram.
- 4) Allow annotation, record accessing, and traceable information being attached to the report.
- 5) Extract key information from the electronic records and store them in the Oracle or SQL Server database for searching, reporting, and integrating with other application such as LIMS.
- 6) Enable a secure and regulatory or corporate compliant environment.
- 7) Allow for accessing all records through a standard web browser.

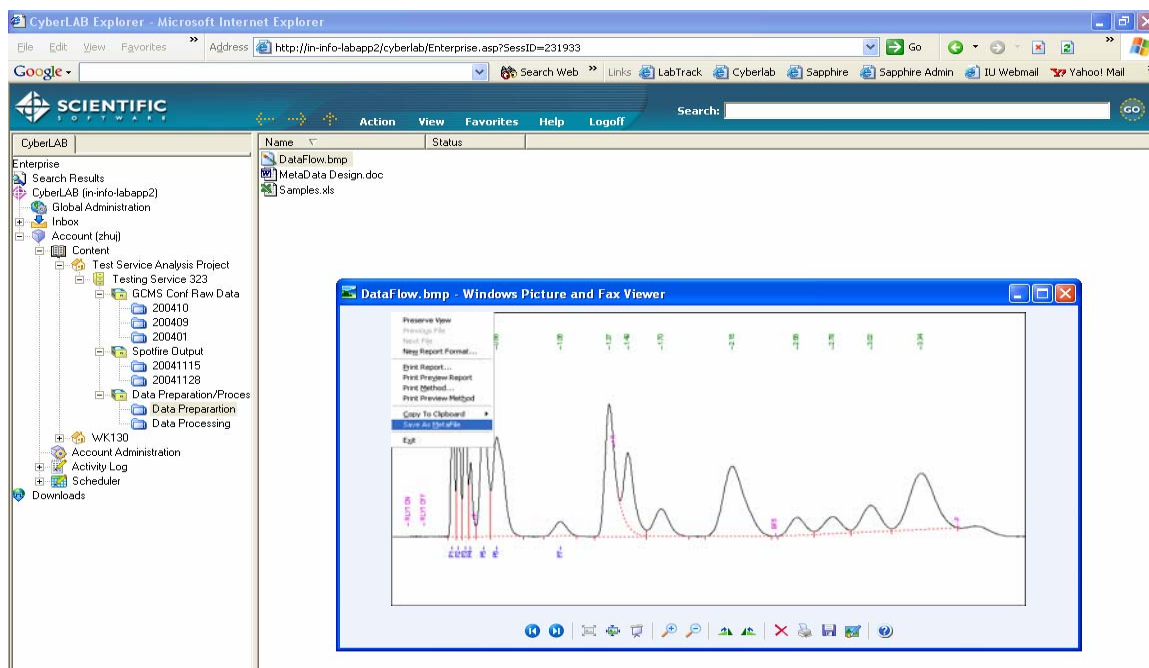


Figure 1.4: A scientific data management system (SDMS)

1.5. LabTechie

LabTechie is the middle layer created in this project. It streamlines data flow in the laboratory to increase productivity, improve quality standards, and facilitate regulatory compliance.

The manual process of entering large amounts of data into LIMS and then reviewing the data is time consuming and costly. Moreover, the possibility of generating errors when entering data into LIMS manually is considerably higher than doing it automatically. This middle layer addresses these problems by automating data entry to LIMS. Thus, the turn-around time of sample analysis will be reduced significantly. Also, the data quality will be improved and cost of managing the data is reduced.

As required by the U.S. FDA, LIMS is becoming an integral part of any laboratory that needs to meet government regulations. The middle layer designed in this research will be able to meet the 21 CFR Part 11 Electronic Records and Electronic Signatures and the record retention requirements of Cross-Media Electronic Reporting and Record-keeping Rule (CROMERR) proposed by FDA.

2. BACKGROUND

2.1. History and related research

LIMS is an information management system designed for analytical laboratories. Various types of laboratory data, ranging from sample log-in, analysis task assignment to analysis results, are entered into LIMS. Then, these data are sorted and organized into meaningful information [6]. Different formats of reports are then created to present the information. Although LIMS has these advantages, most of LIMS do not provide an interface to extract data from analytical instruments and enter them directly into LIMS. Therefore, manually entering vast amounts of data generated by instruments into LIMS is becoming a bottleneck for analytical laboratories. Particularly, some advanced instruments such as high through-put screening equipment generate huge amount of data, increase the demand for interfacing these instruments with LIMS.

The conventional method of integrating LIMS with instruments is to create drivers between LIMS and instruments. One approach is that a driver can be created in either instrument or LIMS and then communicate with the other one. Another approach could be drivers are created on both instrument and LIMS. These two drivers then communicate with each other to transfer data. A scheme of these two approaches is illustrated in Figure 2.1.

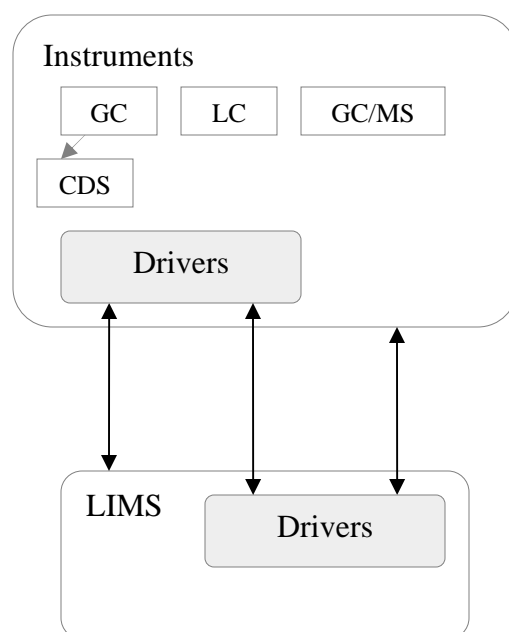


Figure 2.1: LIMS and instruments integration using drivers

There are some problems with this approach. One problem is that drivers are not generic, which means a driver created based on the configuration of one instrument may not work for another instrument, even though both instruments are from the same vendor. Another problem is that the analysis results generated from analytical instruments are still raw data. Data review may be needed to process the instrument output before they are ready to be sent to LIMS. However, drivers do not provide user options to view the raw data.

Taking these problems into consideration, it is possible that this design will generate a large number of different drivers for a laboratory that has a variety of instruments. Each driver accommodates a certain configuration for a specific instrument. Obviously, instead of thousands of different drivers, it is preferable to have a generic middleware that can be applied with very little configuration to work with most instruments. In addition to data collection from any instrument, this middleware will also provide user options for viewing data and storing final results in LIMS.

In the generic middleware, data collected from instruments will be stored in an intermediate document before it is sent to LIMS. Currently there are no standard formats of documents used to store and report data. Instruments from different vendors generate outputs in different formats. Thus, it is difficult to select one generic format for data storage from various instruments. Some “standard formats,” such as AnDI, JCAMP and SPC, have been used by a variety of vendors to store data and results from all kinds of analytical instruments. These “standard formats” were developed to allow data interchange between various software packages. However,

none of these standard format developers (either public or private) have the required support for large groups of users, instrument vendors, or government regulators to promote their standards. Therefore, the World Wide Web Consortium (W3C, <http://www.w3.org>), an independent standard body that governs the definitions of the format used on the internet, developed the Extensible Markup Language (XML), a universal format for exchanging structured documents and data on the Web to address this issue. In addition, relevant data are not always stored in one format. The data retention period is usually longer than the lifetime of an analytical instrument. As a result, storage of outdated software and hardware for a long time may be necessary, which creates a potential problem for companies to maintain records over the lifetime of their products.

One solution to the problem of data integration is to create a middle layer that can save the data in a neutral and secure format to transfer and maintain the analytical results. Data would be platform independent and accessible from multiple applications. As a result, obsolete hardware and software currently needed to access data can be eliminated.

XML is tailor-made to serve as the basis of a file format for long-term archiving of analytical instrument data for a number of reasons listed below:

- 1) XML is based on a public domain standard controlled by a completely independent body, the W3C [9].
- 2) XML is currently used as a data interchange mechanism by many mainstream business applications. It has been proposed to be a universal data interchange standard for all types of data via networked applications (i.e., intranets and

the internet), which means it is highly likely that it will be supported by future computer platforms.

- 3) The schema or data type definitions (DTD) that defines the structures for a particular type of data can be widely distributed (e.g., a web site, database, or file server) and software applications can use it to automatically validate the "correctness" of the formatted data whenever an XML file is opened.

XML has the security standards recommended by W3C, which defines XML vocabularies and processing rules in order to meet security requirements. These standards use legacy cryptographic and security technologies, as well as emerging XML technologies to provide a flexible, extensible and practical solution toward meeting security requirements. The XML security standards include XML Digital Signature for integrity and signing solutions, XML Encryption for confidentiality, XML Key Management (XKMS) for public key registration, location and validation. By implementing these security standards on the XML format of data files from instrument output, the middle layer will be able to meet the security requirements from the regulations such as 21 CFR Part 11 [7].

How to implement these standards is the main focus of this proposed project. There are three major XML security implementation tools available to use for free [8]. One tool is Apache XML Security, created by Apache Software Foundation [9]. Apache XML Security relies on a Java-based security library and its related application. In 2004, a C++-based security library was introduced by this company; however, this library only provides basic functions in comparison to its Java counterpart. The second XML security implementation tool is IBM XML Security

Suite, developed by the IBM alphaWorks Group, which is also a Java-based implementation tool [9]. The third tool is XMLSec Library, a C-based implementation that has been developed and maintained by Aleksey Sanin in the XML Security Library [9].

XMLSec Library has been selected as the security implementation for the middleware in this research. The decision was made based on some of the features of XMLSec Library listed below:

- 1) Meet all the XML Signature and XML Encryption syntax and processing standards from W3C.
- 2) Interact with other XML security implementations completely. In other words, the application developed by other security implementations will be able to use the XMLSec library, and vice versa. For instance, the signature created using XMLSec Library could be verified by other applications designed by using IBM XML Security Suite.
- 3) Provide a Software Development Kit for application development.
- 4) Use a C-based library which is more easily incorporated into the .NET platform.
- 5) Offer free source code and some implementation examples.

2.2. Current practice and understanding

There are two major vendors (Labtronics and CSols) in the market that provide products for interfacing the instruments and LIMS. However, these two vendors are

charging exorbitant amounts of money for integrating each instrument with LIMS to transfer the data from the instruments to LIMS.

2.3. Intended project

This project is intended to create a “general purpose” middleware to bridge the instruments and LIMS. The interface of the middleware will create a secure intermediate environment for the data extraction from the instruments.

3. METHODS

3.1. Materials and instruments

3.1.1. Networking

LAN/WAN based on TCP/IP protocol

3.1.2. Hardware

Servers: one Windows 2003 application server (Microsoft, Redmond, WA), one Oracle 9i database server (Oracle, Redwood Shores, CA) and one Oracle 10g database server (Oracle, Redwood Shores, CA), a PerkinElmer Chromatography Instrument Simulator (PerkinElmer, Wellesley, MA), and a Agilent 6890 GC/MS

3.1.3. Software

3.1.3.1. LabVantage Sapphire™ LIMS

LabVantage Sapphire™ LIMS (LabVantage, Bridgewater, MA) helps laboratories to increase productivity, quality, and reduce operating costs. Sapphire's automatic configuration and communication process enable laboratories to modify their business-specific workflows, integrate with robots and analytical instruments, effectively manage samples, and meet government regulations. In addition, the browser-based architecture provides the ability to access Sapphire™ LIMS from virtually any Internet

access device for both internal and external users. It also offers the functionality, flexibility, and scalability that can meet customers' short and long-term needs [10].

3.1.3.2. LabWare LIMS

LabWare LIMS (LabWare, Wilmington, DE) is one of earliest client-configurable LIMS in the world. It provides more out-of-the-box functions than other LIMS products, which means that customers can set up and maintain their LIMS without the need of additional programming [11].

3.1.3.3. PerkinElmer TotalChrom

PerkinElmer TotalChrom is the most advanced software among chromatography software. It offers a computing strategy to manage chromatography data quickly and efficiently. Once configured with GCs, the TotalChrom CDS serves as a controller and a data manager for the combined system. Compatible with all commercial GCs, TotalChrom is one of the best choices for chromatography data in the demanding multi-user, multi-instrument laboratory environments.

3.1.3.4. Agilent ChemStation

The Agilent ChemStation Series (Agilent, Palo Alto, CA) software is most widely sold data system in the analytical industry. With the ability to handle a wide range of chromatographic applications such as LC, LC/MS,

GC, GC/MS, A/D, CE and CE/MS, ChemStation has an easy-to-use graphical user interface and built-in standard report templates to reduce the amount of time spent on routine tasks.

3.1.3.5. CyberLAB ECMS

CyberLAB ECMS (Scientific Software, Pleasanton, CA) is a web-based electronic library that can collect, organize, index, store, archive, and share electronic records from raw analytical instrument data to reports, Microsoft Office documents, PDF documents, molecular drawings, pictures, and video. CyberLAB provides powerful search capabilities. For example, it can automatically extract searchable metadata from every file. Knowledgeable operators will be able to locate useful information faster and thus make wise decisions.

3.1.3.6. NuGenesis SDMS

Featuring application-independent architecture, NuGenesis SDMS (Waters, Milford, MA) utilizes unique “file and print capture” technology that consolidates and manages critical scientific information generated from any instrument or application. The Web-based NuGenesis Scientific Data Management System (SDMS) platform provides a foundation for scientific data preservation and integration among the collaborative flow throughout the organization.

3.1.4. Other equipment

Two PCs and one laptop

3.2. Data subjects

3.2.1. Data subject clearance

Data files generated from analytical instruments were used for this project. Data files were in either tab-delimited or comma-delimited ASCII format.

3.2.2. Data size

Each file contains one sample's analysis data generated from the instrument.

3.3. Procedures and interventions

3.3.1. Design

3.3.1.1. Requirement analysis

This project will create a “general purpose” middleware to bridge the analytical instruments and LIMS. The interface in the middleware will create a secure intermediate document for the data extraction from the instruments.

3.3.1.2. Functionality analysis

The project will handle the following functions:

- 1) Create a secure document for the data extracted from instruments.
- 2) Review and manipulate data in the interface of the middleware.
- 3) Store data into LIMS.

3.3.1.3. Preliminary research

Some preliminary research was conducted to provide information for middleware design.

- 1) Implementation of the XML security onto the XML files containing the extracted data.
- 2) Research on testing if the implementation of XML Security could be set up as a Web Service.
- 3) Decision of the application format. Two types of applications (i.e., A Windows or a Web application) were tested.

3.3.1.4. Project design

- 1) Selection of instruments that represent the typical analytical job to generate data for the project.
 - Perkin-Elmer Chromatography Instrument Simulator with TotalChrom CDS
 - Hewlett-Packard GCMS with ChemStation CDS
- 2) Selection of the LIMS

- LabVantage Sapphire LIMS
 - Labware LIMS
- 3) Selection of the SDMS system
- CyberLAB ECMS
 - NuGenesis SDMS
- 4) Selection of communication methods between instruments in the middle layer
- TCP/IP, RS232, or file sharing through the network were tested to decide the communication method used in middle layer.
 - Build communication between the middle layer and the instruments by using the selected communication methods. A network space is created in the network if the selection of a shared network space is necessary.
- 5) Decision on the format of file used for data transportation.
- The format could be ASCII tab-delimited or comma-delimited format or others, which mainly rely on the instrument output. No matter which format is used, a generic ASCII-to-XML parser is designed to convert the file to XML file.

3.3.2. Process description

3.3.2.1. A diagram in Figure 3.1 shows how data flows in the project.

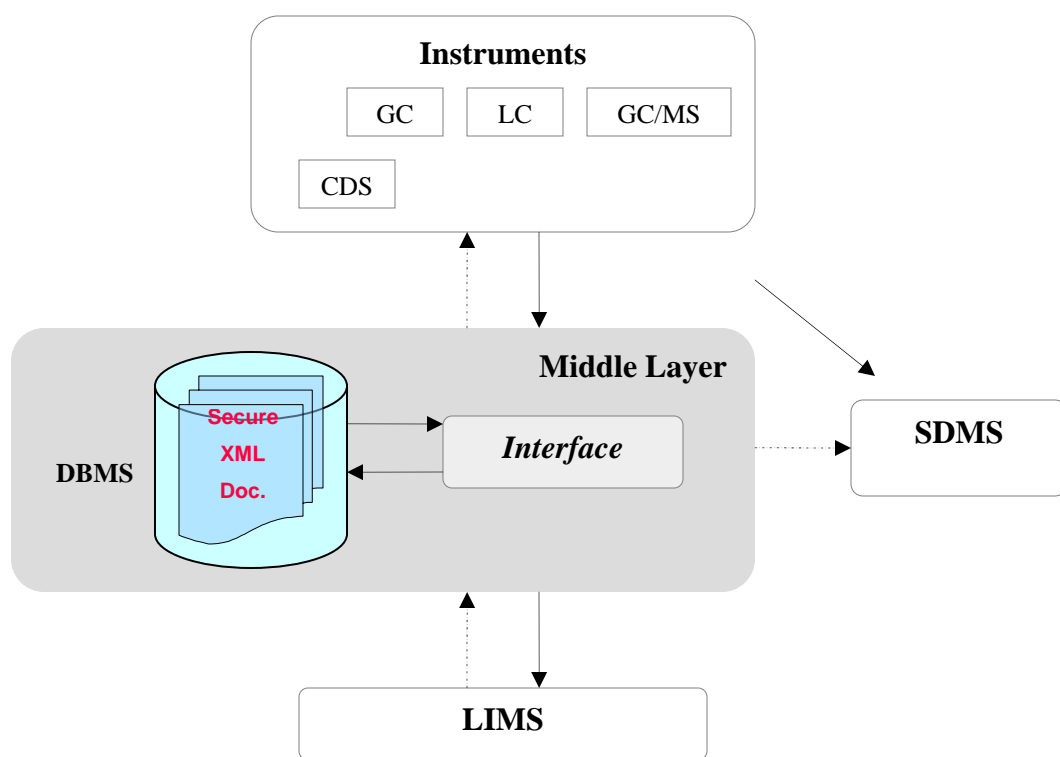


Figure 3.1: Project data flow diagram

3.3.2.2. Instrument data collection

First of all, related analysis information from LIMS was collected by the middle layer and stored in a LIMS database. Second, tables containing sample information and analytical information were identified. Third, the information was retrieved and a secure intermediate file containing the information was converted into an ASCII file. Then efforts were focus on creating an ASCII file directly. The file created was used as the sequence file when the instrument was TotalChrom.

3.3.2.3. Middle layer development

The middle layer sends the ASCII file containing sample data to the instruments. A network space is preferred for data sharing. Therefore, the instrument can read the data from the shared network space and run the samples. However, there is no auto-sampler equipped with the TotalChrom; the sample sequence is not able to automatically run. Accordingly, the samples in the sequence need to be run manually. Then, the instruments are set up to collect data consistently into one series of files with unique identification.

3.3.2.4. Reviewing data in the middle layer

Based on the identification of the data files, data can be viewed from the shared space in the network. First, the data generated from the instruments could be previewed in their original format. Second, web services were used to automatically convert data into a secure XML format. The spreadsheet format of the middleware interface was used to view the data. It is relatively easy to convert a tab- or comma-delimited files into a spreadsheet [MS Excel] file.

3.3.2.5. Management of data storage into LIMS

At this stage, decisions were made on which raw data and related information should be posted into LIMS. The selected data were then stored in the secure XML file and later sent to LIMS.

3.3.2.6. LIMS system setup

- 1) Classify users based on different privileges
- 2) Instrument related information and specification
- 3) Chemicals testing items and specifications
- 4) Testing result review
- 5) Functions or jobs to retrieve sample data and generate XML files.

Store XML files containing test results into a LIMS database. This part could be done in the middle layer.

3.3.2.7. SDMS setup

- 1) Create projects to store related information
- 2) Set up a chromatogram viewer with CyberLAB and other add-on functions to view the analysis results.
- 3) Project implementation (Will redo it in MS Project)

- Requirements Analysis

- Function Analysis

- ◇ Data Flow Diagram

- ◇ Process Description

- ◇ Constraints Analysis

- ◇ Risk and Contingency Analysis

- ◇ Risk and Contingency Matrix

- Interface and Database Design

- Interface and Database Programming

- Function Integration

- Testing and Debugging

- Implementation

- Installation

- Training

- Release

- ◇ Documentation

- ◇ User Manual

- ◇ Maintenance Manual

3.4. Project analysis

3.4.1. Project evaluation plan

The following questions were intended to be answered for evaluating the project.

- 1) Is the project successful?
- 2) Does the project meet the overall goal(s)?
- 3) Did the participants benefit from the project?
- 4) What components are the most effective?
- 5) Are the results worth the project's cost?
- 6) Is this project replicable and transportable?

3.4.2. Method of results analysis

- 1) Check the raw data and prepare data for analysis.
- 2) Conduct initial analysis based on evaluation plan.
- 3) Conduct additional analyses based on initial results.
- 4) Integrate and synthesize findings.

3.5. Expected results

This project will result in the development of a middle layer between the instruments and Laboratory Informatics Management Systems (LIMS).

After implementing this middle layer, the laboratory will be able to reduce the turn-around time and improve the efficiency of laboratory operations by automating the data entry process. The analytical results will show up in the LIMS as soon as the analyst reviews them in the middle layer and gives the final approval. Therefore, significant amounts of working time will be saved by minimizing the time spent in processing the instrumental data. In addition, any risk of error in the data transfer process could be eliminated by removing the human factor involved in even the most carefully monitored manual systems.

Raw data and analytical results securely archived in SDMS could be retrieved anytime for the reviewing, which will meet regulatory requirements. Also, due to the generic nature of the middle layer, users will be able to take ownership of future implementation and maintenance of the interface to include all the analytical instruments in their laboratories.

The project will also give an opportunity to graduate students to work with LIMS, SDMS and CDS in an academic environment, thus giving them the ability to develop creative solutions.

4. RESULTS

4.1. General results

In this project, a middle layer was successfully developed to achieve the proposed objectives. The middle layer is able to collect data from the instruments, convert data to secure XML format of data, store the XML format of data into an Oracle database for archiving purpose, allow users to review the data on the interface of the middle layer, and eventually store the data to LIMS database. The application of the middleware will be able to streamline the laboratory operation from instrument to LIMS and archive the raw data in the database. In addition, the middleware provides secure login for different levels of users including administrators and regular users. Both administrators and the regular users can operate the integrated instruments such as collecting, reviewing and storing data. In addition, the administrators are able to manage the integrated projects. For example, they can add, edit, and delete instruments or user accounts when it is necessary for specific project.

The interface is the core of the middle layer. The functionalities of the interface are listed below by dividing into two groups. One group is the interface management settings, which includes user management, project management, and project summary reports. The other group is the operations including retrieving data from instruments and signing the digital signatures on the data, reviewing

and searching data from archiving database, and manually or automatically posting data to LIMS database.

The interface is built on Microsoft .Net platform using Visual Basic .Net programming language. Oracle10g DBMS is the archiving database server. The interface is named as “LabTechie Total Instrument Interfacing Solutions”. The detailed functionalities of the middle layer are illustrated as follows.

4.1.1. User and project management in the middle layer

1) User management

Figure 4.1 shows the login window of the interface. It has two modes: administrator mode and user mode. An administrator can access all the functionalities and a user is only able to access interface operation group of functions.

User settings form (Figure 4.2) to manage user accounts by adding, editing and deleting username and password.

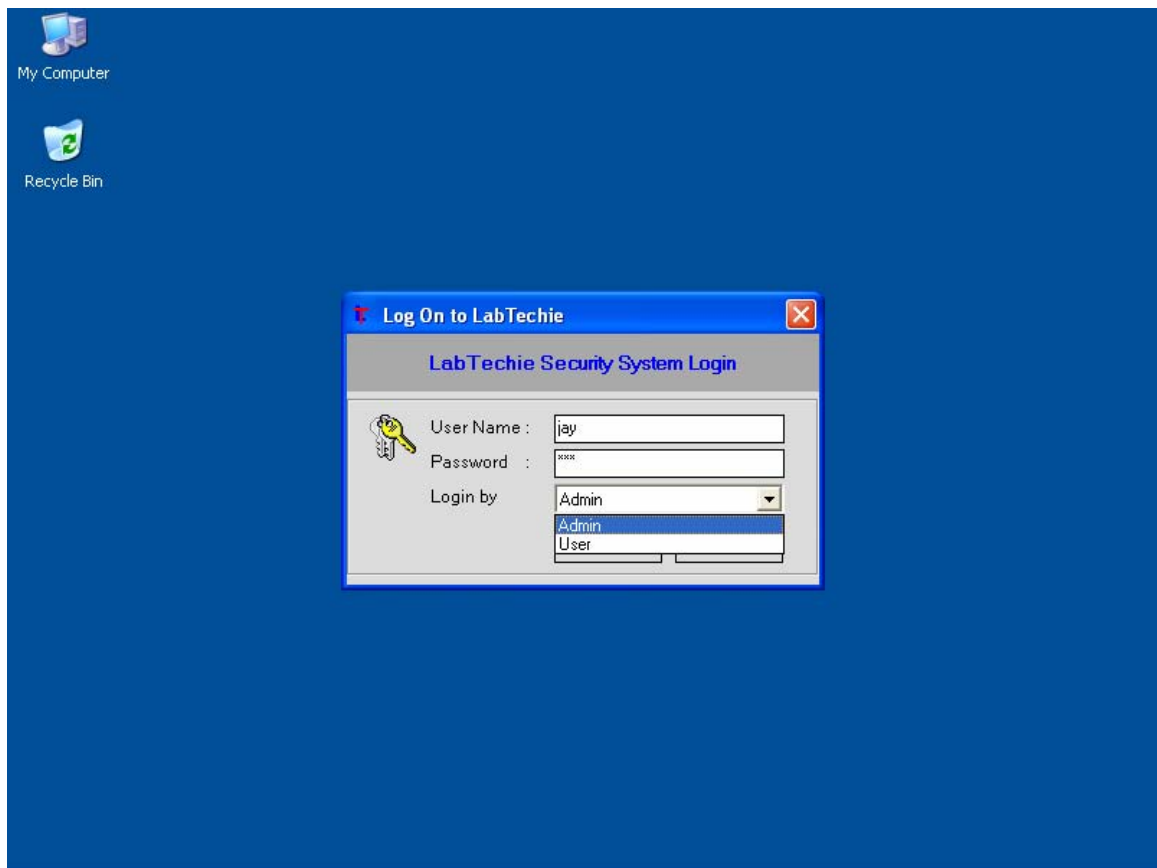


Figure 4.1: Login window

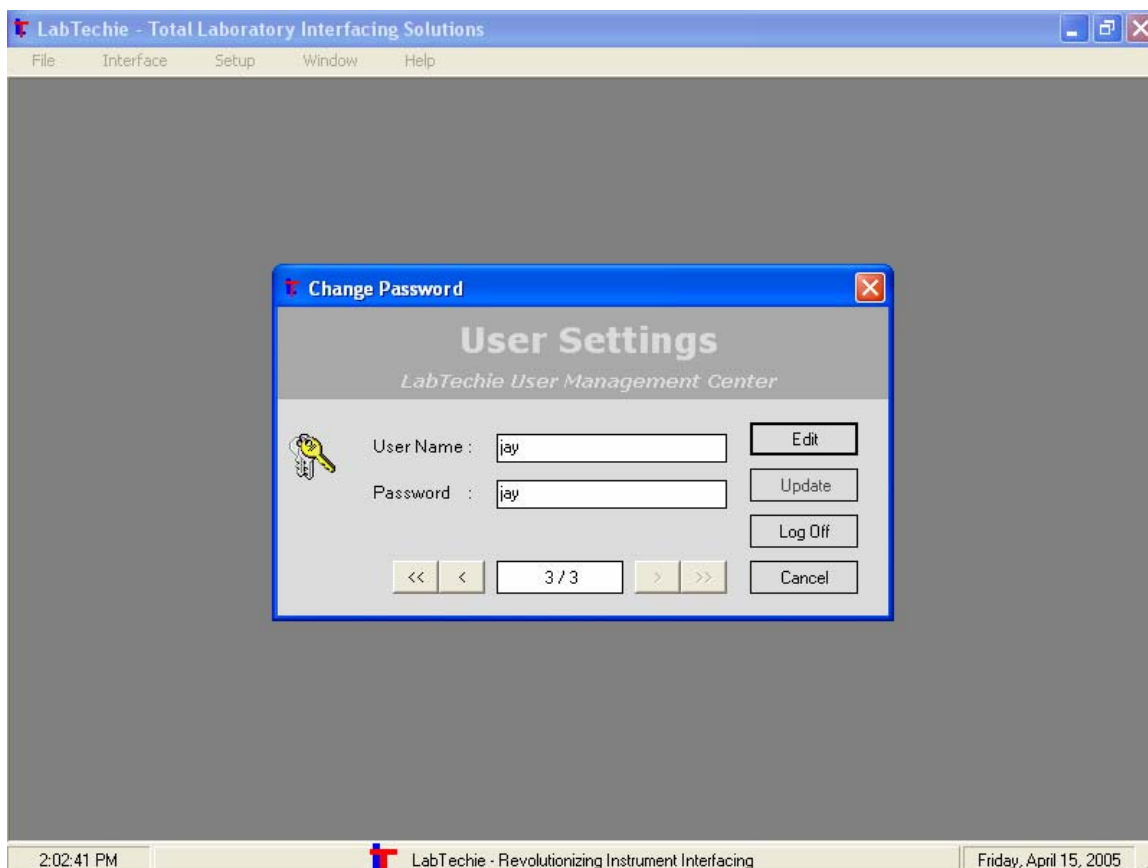


Figure 4.2: User settings

2) Project management for administrator

The projects are managed on the project settings form as shown in Figure 4.3. There are three management functions the administrator can work with to set up a project.

- Project management

A new project can be added, edited, searched, and deleted from the system as shown in Figure 4.3.

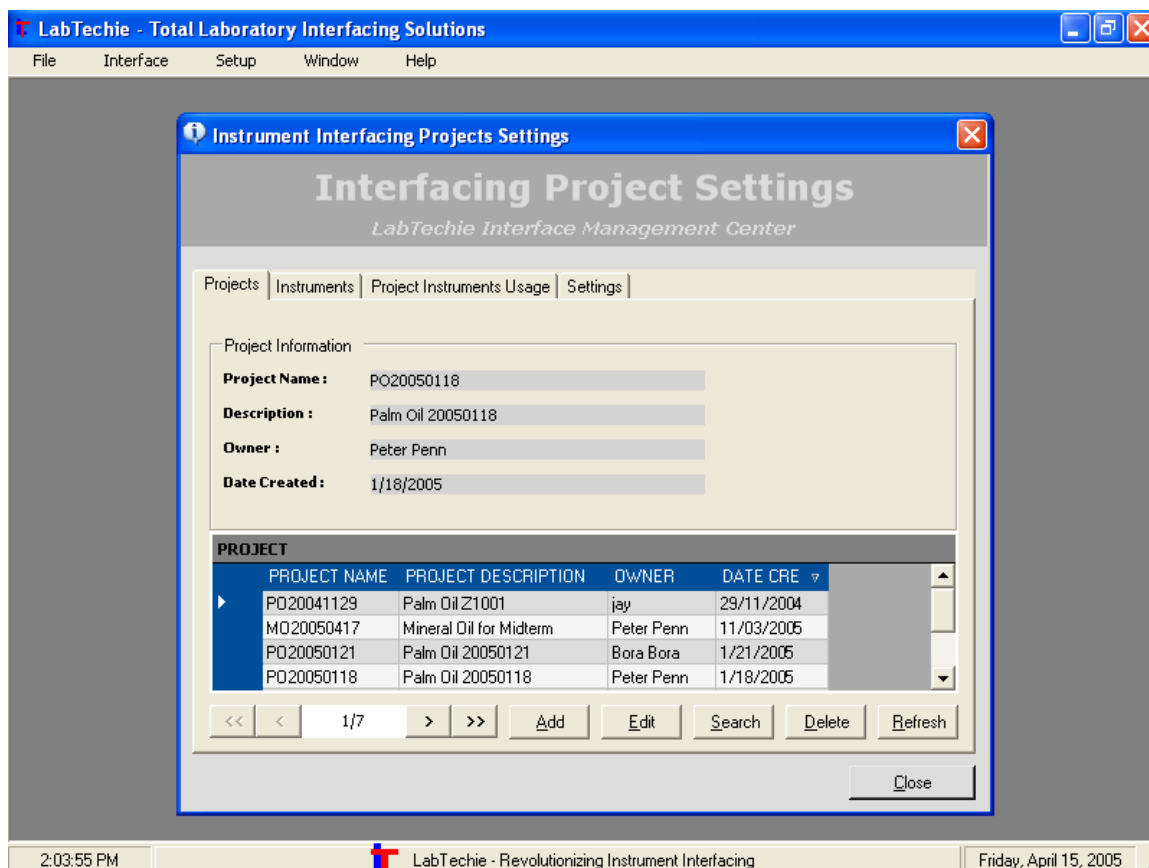


Figure 4.3: Project settings

- Instrument management

A new instrument can be added, edited, searched and deleted from the system as shown in Figure 4.4.

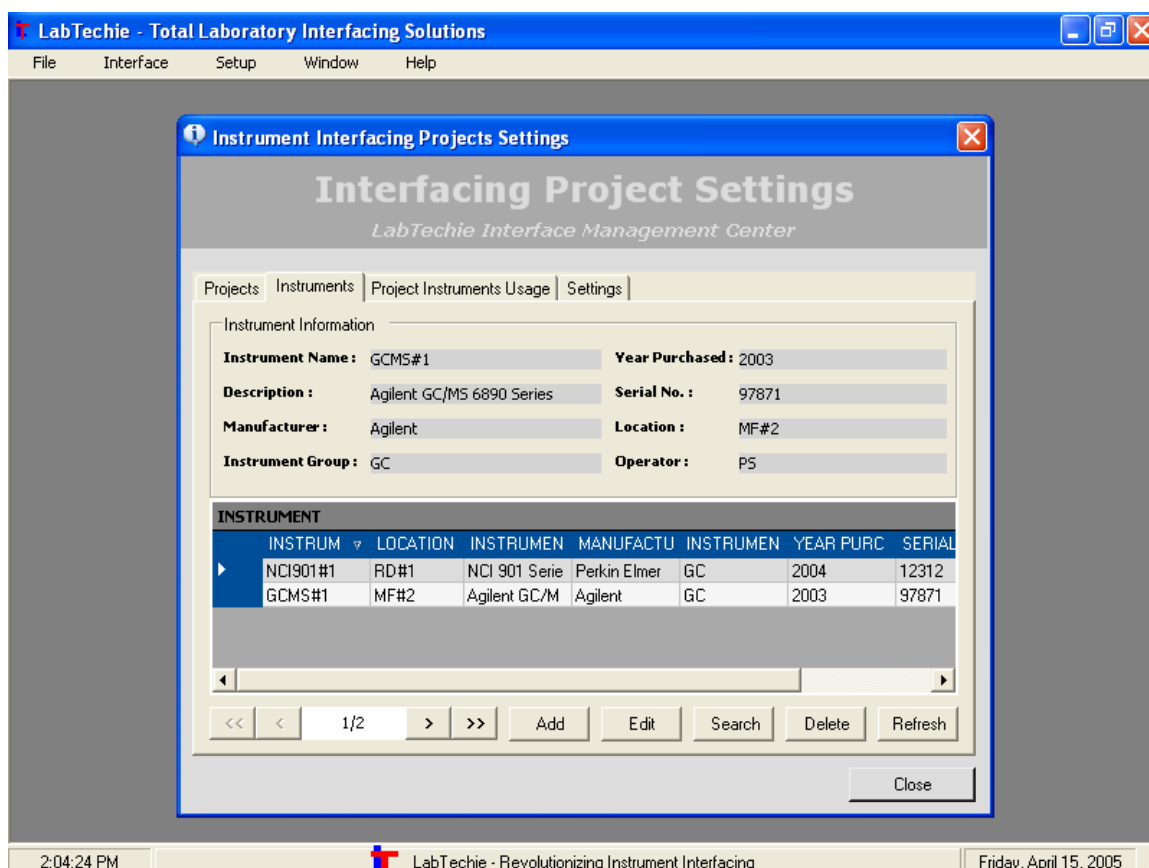


Figure 4.4: Project settings – instruments

- Project instrument usage management

As shown in Figure 4.5., administrators are able to view the information of which interfaced instruments are included in a project. An integrated instrument available in the system can be added to a certain project. In addition, the administrator can search, add and delete an instrument from a project.

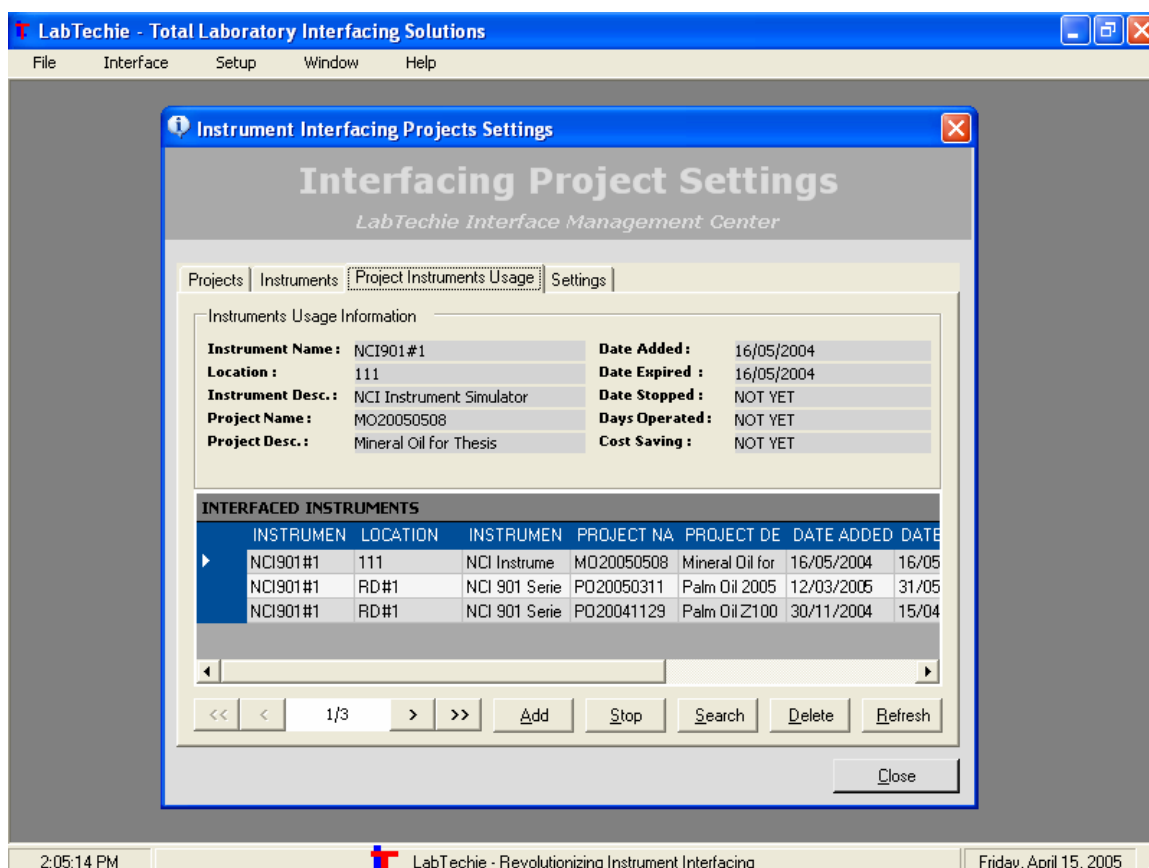


Figure 4.5: Project settings – project instruments usage

3) Project management summary report

Crystal reports are generated to inform users the instrument interfacing projects summarized information (see Figure 4.6). Summary reports for the above three project management functions – project management, instrument management , and instrument usage management can be printed out as shown in Figure 4.7, 4.8, and 4.9, respectively. The reports can also be exported as PDF files (shown in Figure 4.10) when it is necessary.

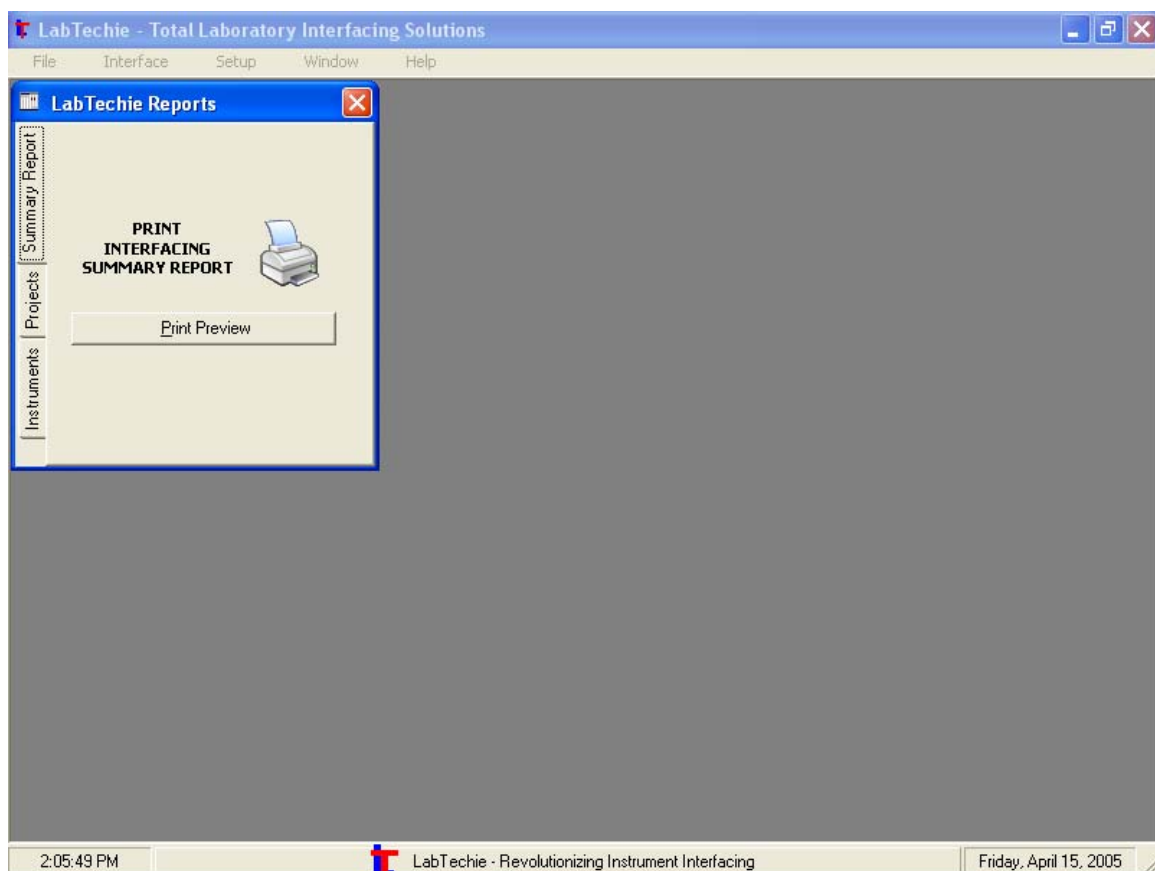


Figure 4.6: Project settings summary report

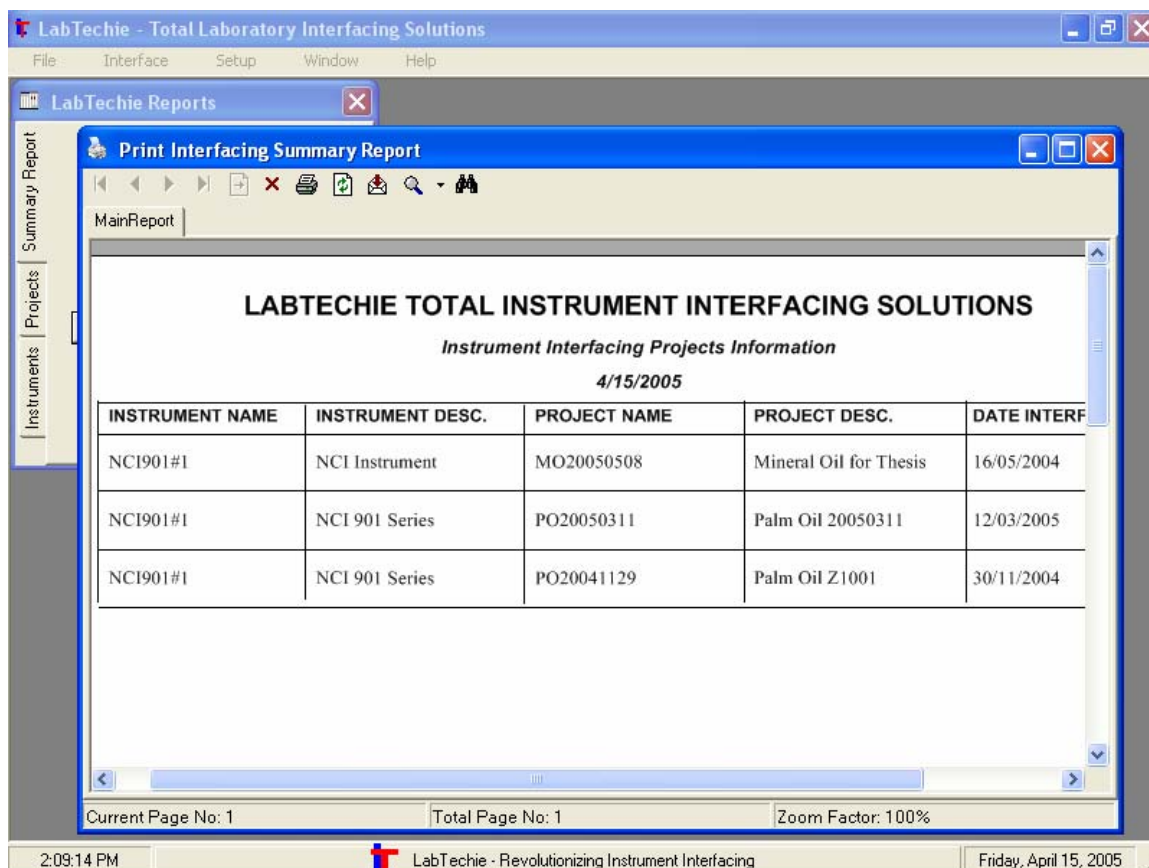


Figure 4.7: Project settings summary report – instrument usage

LabTechie - Total Laboratory Interfacing Solutions

File Interface Setup Window Help

LabTechie Reports

Print Projects Info

MainReport

LABTECHIE TOTAL INSTRUMENT INTERFACING SOLUTIONS

Projects Information
4/15/2005

PROJECT NAME	PROJECT DESCRIPTION	OWNER	DATE CREATE
PO20050118	Palm Oil 20050118	Peter Penn	1/18/2005
PO20050121	Palm Oil 20050121	Bora Bora	1/21/2005
MO20050508	Mineral Oil for Thesis	Jay "SUNNY" Zhu	08/04/2005
MO20050507	Mineral Oil for Graduation	Johnny Forde	02/04/2005
MO20050417	Mineral Oil for Midterm	Peter Penn	11/03/2005
PO20050311	Palm Oil 20050311	Peter Penn	08/04/2005
PO20041129	Palm Oil Z1001	jay	29/11/2004

Current Page No: 1 Total Page No: 1 Zoom Factor: 100%

2:10:48 PM LabTechie - Revolutionizing Instrument Interfacing Friday, April 15, 2005

Figure 4.8: Project settings summary report – project setting

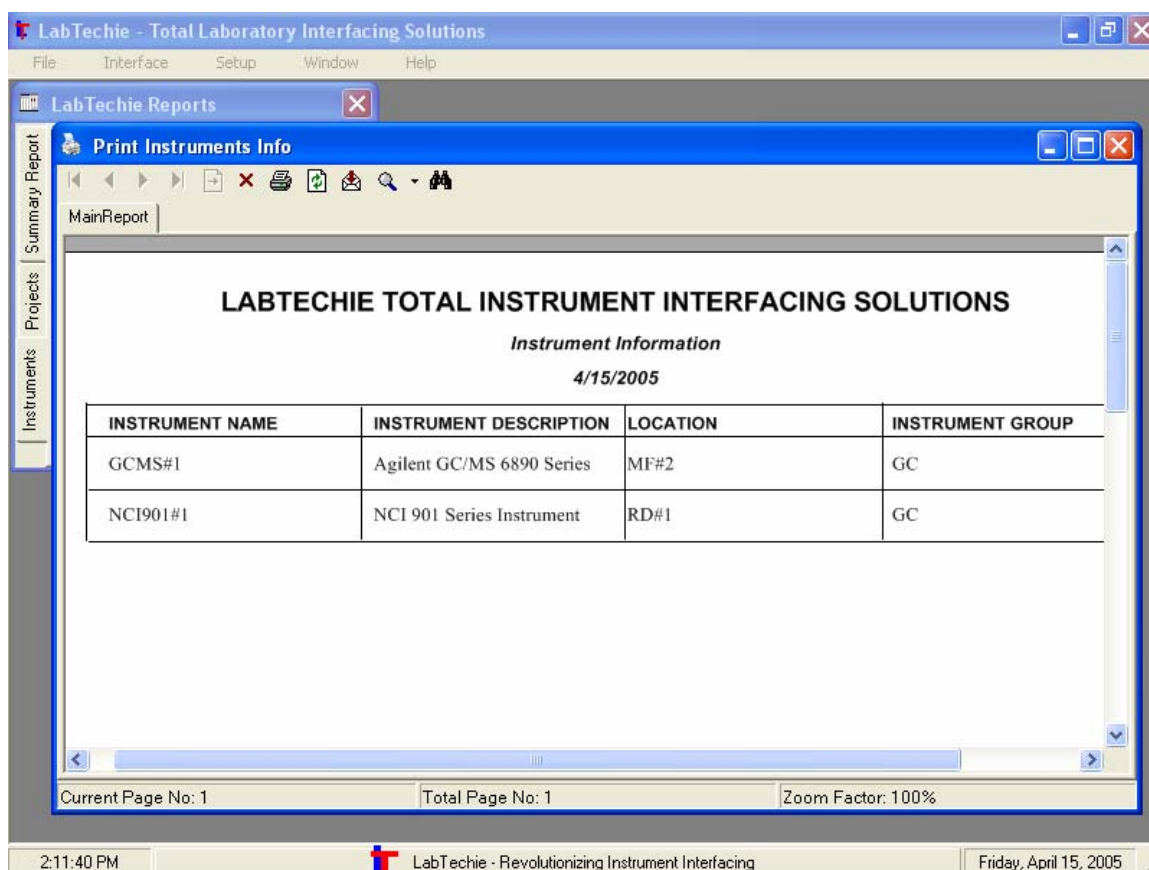


Figure 4.9: Project settings summary report – instrument setting

LABTECHIE TOTAL INSTRUMENT INTERFACING SOLUTIONS

Instrument Interfacing Projects Information

4/15/2005

INSTRUMENT NAME	INSTRUMENT DESC.	PROJECT NAME	PROJECT DESC.	DATE INTERFACED
NCI901#1	NCI Instrument	MO20050508	Mineral Oil for Thesis	16/05/2004
NCI901#1	NCI 901 Series	PO20050311	Palm Oil 20050311	12/03/2005
NCI901#1	NCI 901 Series	PO20041129	Palm Oil Z1001	30/11/2004

Figure 4.10: Project settings summary report PDF file export

4.1.2. Collect and review instrument data in the middle layer

Data is collected from the instrument using the form shown in Figure 4.11. Users can preview data on the interface for the purpose of testing interface or decide if this is the data to be stored in LIMS database. Data shown in the text boxes are in XML format with digital signatures not yet being signed.

LabTechie - Total Laboratory Interfacing Solutions

File Interface Setup Window Help

Data Management Center - Auto-Posting Records

Auto-Posting Instrument Data Records
LabTechie Data Management Center

Select Records :
[Select Input Data File](#) [Select Output Raw Data XML File](#) [Select Output Sample Info XML File](#)
..\data\ncidata001.tx0 ..\data\nci2Raw.xml ..\data\nci2Sample.xml

Select XSL Files :
[Select Sample Info XSL File](#) [Select Raw Data XSL File](#)
..\data\nci2Sample.xsl ..\data\nci2Raw.xsl

Preview Data :
Sample Information :
<?xml version="1.0" encoding="utf-8"?>
<SampleData>
<Operator>zhu</Operator>
<SampleName>Palm Oil Z1001</SampleName>
<SampleID>001</SampleID>
<Study>Palm Oil Analysis</Study>
<InstrumentName>NCI 901</InstrumentName>
</SampleData>

Raw Data Information :
<?xml version="1.0" encoding="utf-8"?>
<GCD data SampleID="001">
<PeakInfo>
<PeakNo>1</PeakNo>
<RetTime>0.082</RetTime>
<Component></Component>
<BL>BV</BL>
<Area>844.82</Area>

Preview Submit Close

6:13:18 PM LabTechie - Revolutionizing Instrument Interfacing Friday, April 15, 2005

Figure 4.11: Data retrieval from instruments

4.1.3. Storing data into LIMS database

As shown in Figure 4.12, user has access to the sample information and raw data information related to each interfaced instruments. Once user decides to store data to LIMS database after reviewing data, user can click post button. Another window (Figure 4.13) will be displayed that allows the user to select an instrument and start sending data collected from that instrument to LIMS.

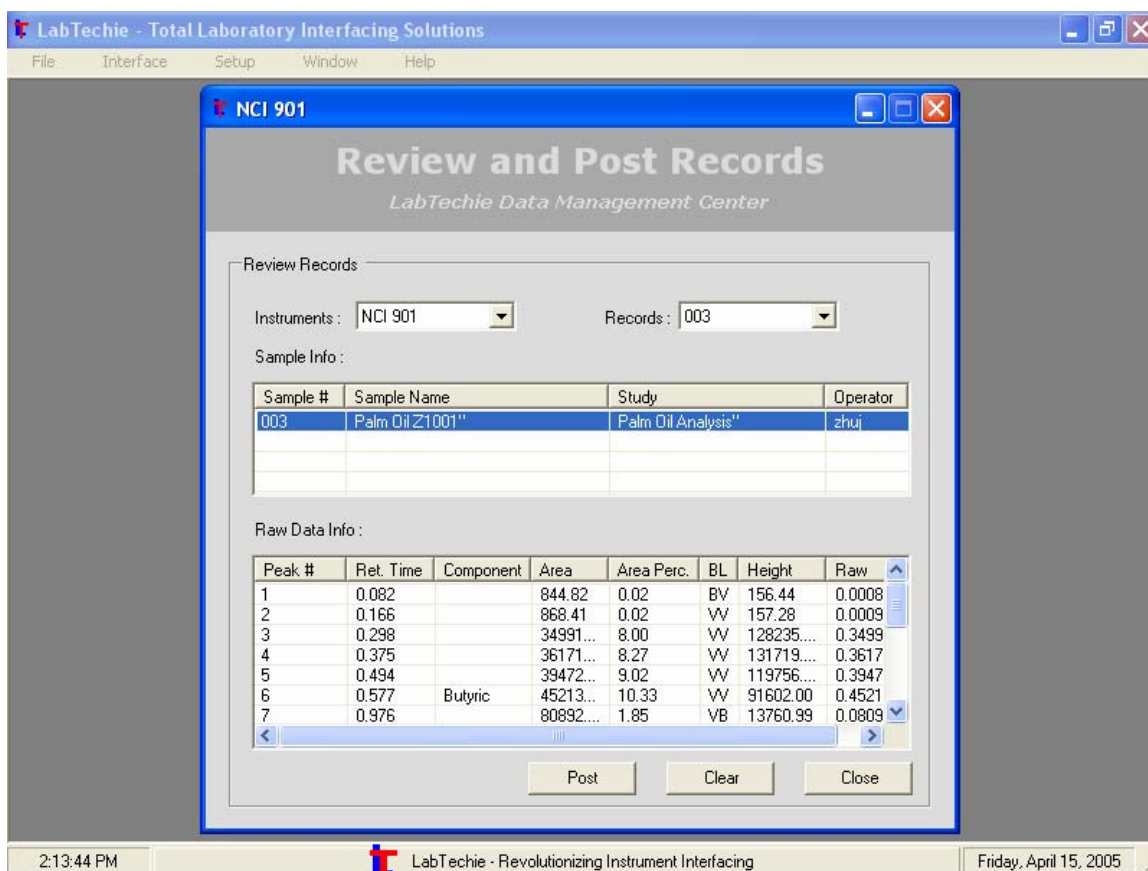


Figure 4.12: Review instrument Data



Figure 4.13: Submit data to LIMS database after reviewing

4.1.4. Search secure XML format of data archived in the database

Figure 4.14 shows the search form that users can use to find the instrument data after it has been archived to the Oracle10g database. Users can query the database with one of the four criteria: project name, instrument name, sample identification, and sample name. After selecting search criteria, conforming data will pop-up the search term dropdown list. User then can click search button and the completed secured XML data will fill up the textbox beneath.

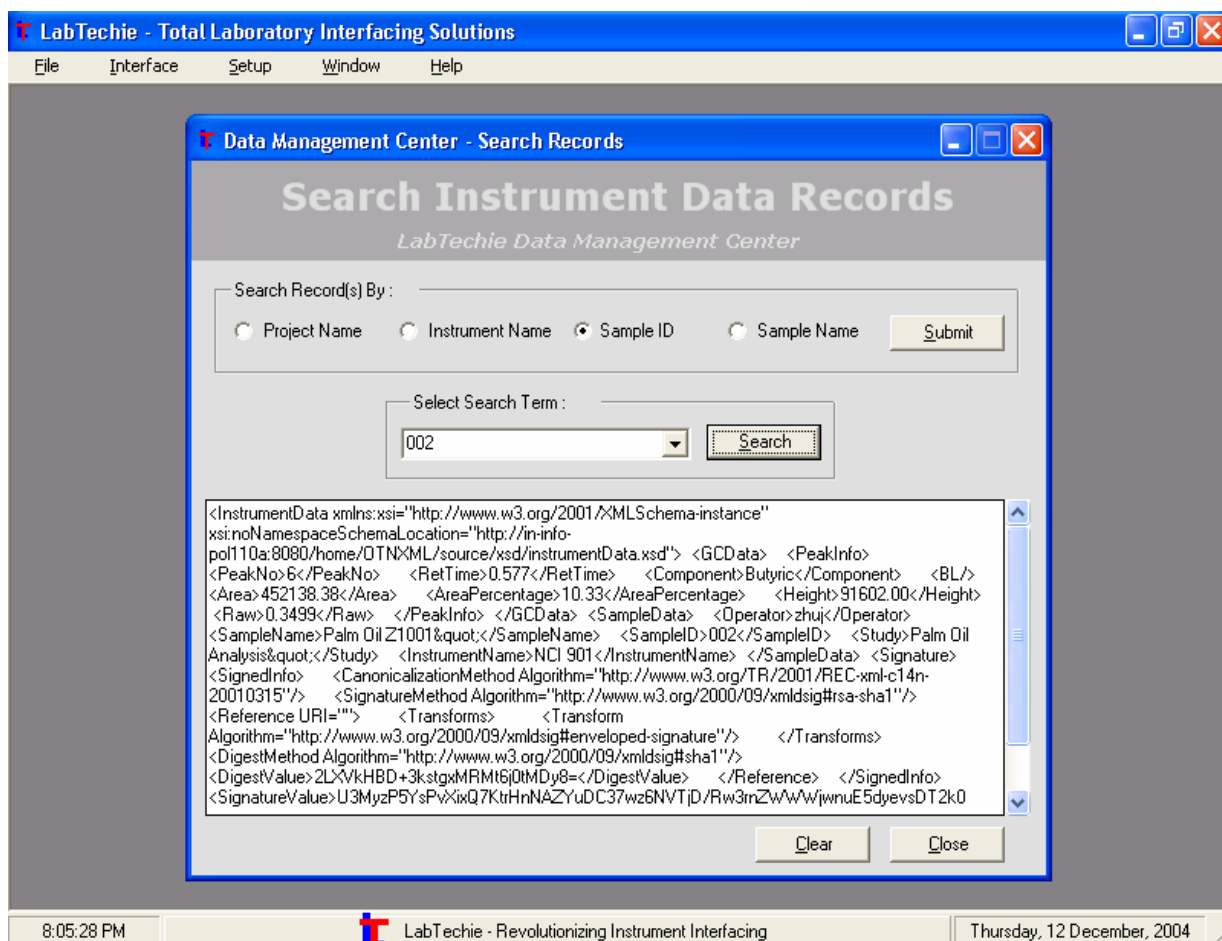


Figure 4.14: Search data by sample id

4.2. Important highlights

4.2.1. Analytical instrument data collection

The analytical instrument is customized to generate reports that contain instrument or vendor specific format of raw data, which could be comma-delimited, tab-delimited or other delimited ASCII files. Figure 4.15 shows an example of a comma-delimited ASCII file.

```

File Edit Format View Help
=====
"Software Version:","6.2.1.0.104:0104","date:","11/19/2004","5:24:11 PM"
"Reprocess Number:","in-info-liapp: 179"
"Operator:","zhuj","Sample Name:","Palm oil Z1001"
"Sample Number:","001","Study:","Palm Oil Analysis"
"Autosampler:","NONE","Rack/Vial:","0,0"
"Instrument Name:","NCI 901","Channel:","A"
"Interface Serial #:","4118280001","A/D mv Range:","1000"
"Delay Time:","0.00","min","End Time:","3.99","min"
"Sampling Rate:","1.5150","pts/s"
"Sample Volume:","1.000000","ul","Area Reject:","0.000000"
"Sample Amount:","1.0000","Dilution Factor:","1.00"
"Data Acquisition Time:","11/19/2004","5:19:59 PM","Cycle:","1"
"Raw Data File:","\\in-info-liapp\TCData\zhuj\Raw\data001-20041119-172009.raw"
"Result File:","\\in-info-liapp\TCData\zhuj\Raw\data001-20041119-172410.rst"
"Inst Method:","\\in-info-liapp\TCData\zhuj\methods\palmoil2004102801","from","\\in-info-liapp\TCData"
"Proc Method:","\\in-info-liapp\TCData\zhuj\methods\palmoil2004102801","from","\\in-info-liapp\TCData"
"Calib Method:","\\in-info-liapp\TCData\zhuj\methods\palmoil2004102801","from","\\in-info-liapp\TCData"
"Report Format File:","\\in-info-liapp\TCData\zhuj\reports\nci901palmoil20041028.rpt"
"Sequence File:","\\in-info-liapp\TCData\zhuj\Sequences\Palmoil2004102901.seq"
=====
""
""
"Analysis Report"
"Peak","Component","Time","Area","Height","Area","Norm. Area","Cal.","Volt","BL","Raw","Adjusted"
"#","Name","[min]","[uv*sec]","[uv]","[%]","[%]","Range","Range","","Amount","Amount"
-----
1,"",0.082,844.82,156.44,0.02,0.02,"",,"",,"BV",0.0008,0.0008
2,"",0.166,868.41,157.28,0.02,0.02,"",,"",,"VV",0.0009,0.0009
3,"",0.298,349918.88,128235.44,8.00,8.00,"",,"",,"VV",0.3499,0.3499
4,"",0.375,361715.48,131719.91,8.27,8.27,"",,"",,"VV",0.3617,0.3617
5,"",0.494,394723.79,119756.50,9.02,9.02,"",,"",,"VV",0.3947,0.3947
6,"Fructose",0.577,452138.38,91602.00,10.33,10.33,"",,"",,"VV",0.4521,0.4521
7,"",0.976,80892.31,13760.99,1.85,1.85,"",,"",,"VB",0.0809,0.0809
8,"",1.286,526730.10,120334.78,12.04,12.04,"",,"",,"BV",0.5267,0.5267
9,"Sucrose",1.404,380050.51,75884.31,8.69,8.69,"",,"",,"VV",0.3801,0.3801
10,"",1.614,174014.77,25474.96,3.98,3.98,"",,"",,"VB",0.1740,0.1740
11,"",2.062,626133.99,64631.87,14.31,14.31,"",,"",,"BB",0.6261,0.6261
12,"",2.472,113347.31,16475.56,2.59,2.59,"",,"",,"BV",0.1133,0.1133
13,"",2.696,124321.25,15629.58,2.84,2.84,"",,"",,"VV",0.1243,0.1243
14,"",2.937,202488.58,24641.39,4.63,4.63,"",,"",,"VV",0.2025,0.2025
15,"Glucose",3.256,587730.31,52281.63,13.43,13.43,"",,"",,"VB",0.5877,0.5877
-----
""
""
4175619 00 990747 62 100 00 100 00 "" "" "" A 2750 A 2750

```

Figure 4.15: A comma-delimited ACSII instrument data file

The instrument has been configured to store the ASCII delimited raw data files in a required location, which is a specific folder on the PC that operates the instrument.

The middle layer is able to access the folder that stores the raw data files through a file sharing system which is setup through the networking. The middle layer scans the directory to find the instrument data files and copy them to a temporary folder on that PC.

4.2.2. Data file parsing

4.2.2.1. Convert the ASCII delimited file to XML file

The ASCII delimited data files are converted to XML format (Figure 4.16) of file in the middle layer [12] [13] [14].

```

<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<!-- Original source: ..\data\ncidata001.tx0 -->
- <GCData>
- <SampleInfo>
- <Tag>=====
</SampleInfo>
<SampleInfo />
- <SampleInfo>
<Tag>Software Version:</Tag>
<Content>6.2.1.0.104:0104</Content>
</SampleInfo>
- <SampleInfo>
<Tag>Date:</Tag>
<Content>11/19/2004</Content>
<Tag>5:24:11 PM</Tag>
</SampleInfo>
<SampleInfo />
- <SampleInfo>
<Tag>Reprocess Number:</Tag>
<Content>in-info-liapp: 179</Content>
</SampleInfo>
- <SampleInfo>
<Tag />
<Content />
</SampleInfo>
<SampleInfo />
- <SampleInfo>
<Tag>Operator:</Tag>
<Content>zhuji</Content>
</SampleInfo>
- <SampleInfo>
<Tag>Sample Name:</Tag>
<Content>Palm Oil Z1001</Content>
</SampleInfo>
<SampleInfo />
- <SampleInfo>
<Tag>Sample Number:</Tag>
<Content>001</Content>
</SampleInfo>
- <SampleInfo>
<Tag>Study:</Tag>
<Content>Palm Oil Analysis</Content>
</SampleInfo>
<SampleInfo />
- <SampleInfo>
<Tag>AutoSampler:</Tag>
<Content>NONE</Content>
</SampleInfo>
- <SampleInfo>
<Tag>Rack/Vial:</Tag>
<Content>0</Content>
<Tag>0</Tag>
</SampleInfo>
</SampleInfo>
- <SampleInfo />
<Tag>1</Tag>
<Content />
</SampleInfo>
- <SampleInfo>
<Tag>0.082</Tag>
<Content>844.82</Content>
</SampleInfo>
- <SampleInfo>
<Tag>156.44</Tag>
<Content>0.02</Content>
</SampleInfo>
- <SampleInfo>
<Tag>0.02</Tag>
<Content />
</SampleInfo>
- <SampleInfo>
<Tag />
<Content>BV</Content>
</SampleInfo>
- <SampleInfo>
<Tag>0.0008</Tag>
<Content>0.0008</Content>
</SampleInfo>
<SampleInfo />
- <SampleInfo>
<Tag>2</Tag>
<Content />
</SampleInfo>
- <SampleInfo>
<Tag>0.166</Tag>
<Content>868.41</Content>
</SampleInfo>
- <SampleInfo>
<Tag>157.28</Tag>
<Content>0.02</Content>
</SampleInfo>
- <SampleInfo>
<Tag>0.02</Tag>
<Content />
</SampleInfo>
- <SampleInfo>
<Tag />
<Content>VV</Content>
</SampleInfo>
- <SampleInfo>
<Tag>0.0009</Tag>
<Content>0.0009</Content>
</SampleInfo>
<SampleInfo />
- <SampleInfo>
<Tag>3</Tag>
<Content />

```

Figure 4.16: An XML format of original instrument data file

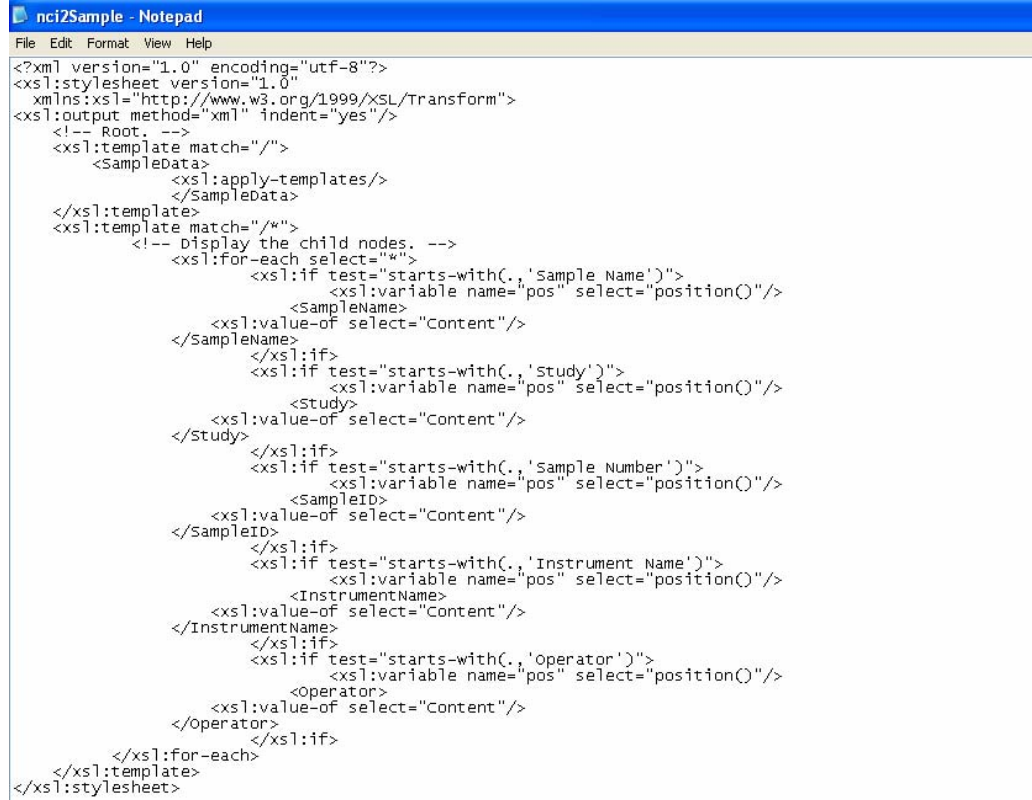
4.2.2.2. Transform XML data file with XSLT

Extensible Stylesheet Language Transformation (XSLT) is used to transform the data from one XML file containing original data to two separate

XML files – sample XML file and raw data XML file [15]. By breaking the instrument data into two parts, two relational objects, sample and raw data, were created. These two objects are related to each other through the unique ID that represents the instrument data, which is the sample ID in most cases. Thus, queries can be built in the middle layer to retrieve data and generate information after the data is stored into Oracle DBMS.

1) Sample information XSLT file

This XSLT file (Figure 4.17) is created to extract the sample information from the instrument data and form an XML file that only contains sample information (Figure 4.18)

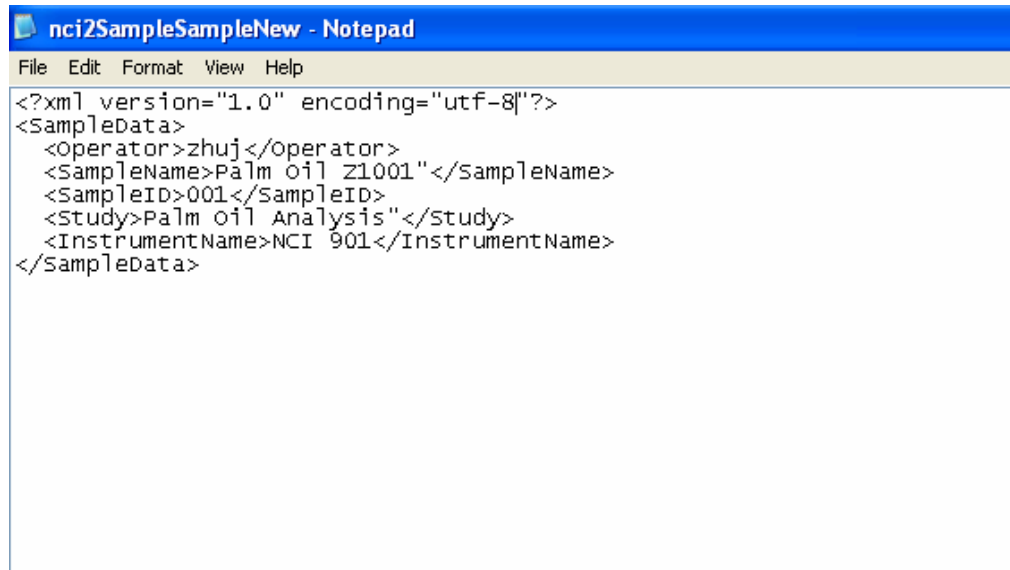


```

<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" indent="yes"/>
<!-- Root. -->
  <xsl:template match="/">
    <SampleData>
      <xsl:apply-templates/>
    </SampleData>
  </xsl:template>
  <xsl:template match="/">
    <!-- Display the child nodes. -->
    <xsl:for-each select="*">
      <xsl:if test="starts-with(., 'Sample Name')">
        <xsl:variable name="pos" select="position()"/>
        <SampleName>
          <xsl:value-of select="Content"/>
        </SampleName>
      </xsl:if>
      <xsl:if test="starts-with(., 'Study')">
        <xsl:variable name="pos" select="position()"/>
        <Study>
          <xsl:value-of select="Content"/>
        </Study>
      </xsl:if>
      <xsl:if test="starts-with(., 'Sample Number')">
        <xsl:variable name="pos" select="position()"/>
        <SampleID>
          <xsl:value-of select="Content"/>
        </SampleID>
      </xsl:if>
      <xsl:if test="starts-with(., 'Instrument Name')">
        <xsl:variable name="pos" select="position()"/>
        <InstrumentName>
          <xsl:value-of select="Content"/>
        </InstrumentName>
      </xsl:if>
      <xsl:if test="starts-with(., 'Operator')">
        <xsl:variable name="pos" select="position()"/>
        <Operator>
          <xsl:value-of select="Content"/>
        </Operator>
      </xsl:if>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>

```

Figure 4.17: An XSLT file to parse instrument sample data



```

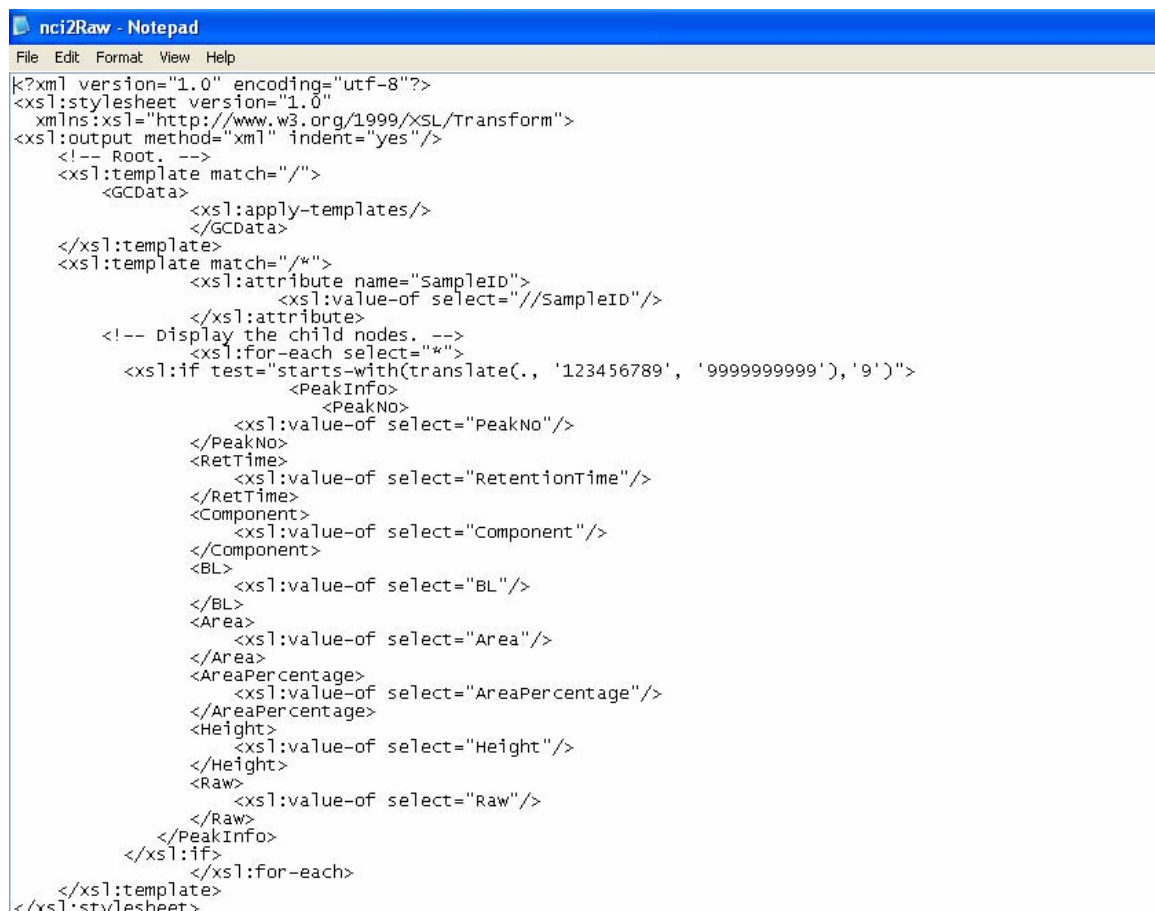
<?xml version="1.0" encoding="utf-8"?>
<SampleData>
  <Operator>zhuji</Operator>
  <SampleName>Palm Oil Z1001</SampleName>
  <SampleID>001</SampleID>
  <Study>Palm Oil Analysis</Study>
  <InstrumentName>NCI 901</InstrumentName>
</SampleData>

```

Figure 4.18: An XML format of instrument sample data file

2) Raw data information XSLT file

Figure 4.19 shows one raw data information XSLT file created to extract the raw data information from the original instrument data and to form an XML file that only contains raw data information (Figure 4.20)



```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" indent="yes"/>
  <!-- Root. -->
  <xsl:template match="/">
    <GCData>
      <xsl:apply-templates/>
    </GCData>
  </xsl:template>
  <xsl:template match="/*">
    <xsl:attribute name="SampleID">
      <xsl:value-of select="//SampleID"/>
    </xsl:attribute>
    <!-- Display the child nodes. -->
    <xsl:for-each select="/*">
      <xsl:if test="starts-with(translate(., '123456789', '999999999'), '9')">
        <PeakInfo>
          <PeakNo>
            <xsl:value-of select="PeakNo"/>
          </PeakNo>
          <RetTime>
            <xsl:value-of select="RetentionTime"/>
          </RetTime>
          <Component>
            <xsl:value-of select="Component"/>
          </Component>
          <BL>
            <xsl:value-of select="BL"/>
          </BL>
          <Area>
            <xsl:value-of select="Area"/>
          </Area>
          <AreaPercentage>
            <xsl:value-of select="AreaPercentage"/>
          </AreaPercentage>
          <Height>
            <xsl:value-of select="Height"/>
          </Height>
          <Raw>
            <xsl:value-of select="Raw"/>
          </Raw>
        </PeakInfo>
      </xsl:if>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

Figure 4.19: An XSLT file to parse instrument raw data file

```

File Edit Format View Help
<?xml version="1.0" encoding="utf-8"?>
<GCData SampleID="001">
  <PeakInfo>
    <PeakNo>1</PeakNo>
    <RetTime>0.082</RetTime>
    <Component></Component>
    <BL>BV</BL>
    <Area>844.82</Area>
    <AreaPercentage>0.02</AreaPercentage>
    <Height>156.44</Height>
    <Raw>0.0008</Raw>
  </PeakInfo>
  <PeakInfo>
    <PeakNo>2</PeakNo>
    <RetTime>0.166</RetTime>
    <Component></Component>
    <BL>VV</BL>
    <Area>868.41</Area>
    <AreaPercentage>0.02</AreaPercentage>
    <Height>157.28</Height>
    <Raw>0.0009</Raw>
  </PeakInfo>
  <PeakInfo>
    <PeakNo>3</PeakNo>
    <RetTime>0.298</RetTime>
    <Component></Component>
    <BL>VV</BL>
    <Area>349918.88</Area>
    <AreaPercentage>8.00</AreaPercentage>
    <Height>128235.44</Height>
    <Raw>0.3499</Raw>
  </PeakInfo>
  <PeakInfo>
    <PeakNo>4</PeakNo>
    <RetTime>0.375</RetTime>
    <Component></Component>
    <BL>VV</BL>
    <Area>361715.48</Area>
    <AreaPercentage>8.27</AreaPercentage>
    <Height>131719.91</Height>
    <Raw>0.3617</Raw>
  </PeakInfo>

```

Figure 4.20: An XML format of instrument raw data file

4.2.3. Development of a secure XML format of instrument data files

Figure 4.21 shows an XML format of instrument data file. The file is merged from instrument sample data file and raw data file. Figure 4.22 shows an XML format of instrument data file that has been signed with a digital signature.

```

File Edit Format View Help
<?xml version="1.0" encoding="utf-8"?>
<InstrumentData>
  <SampleData>
    <Operator>zhuji</Operator>
    <SampleName>Palm oil Z1001</SampleName>
    <SampleID>001</SampleID>
    <Study>Palm oil Analysis</Study>
    <InstrumentName>NCI 901</InstrumentName>
  </SampleData>
  <GCData SampleID="001">
    <PeakInfo>
      <PeakNo>1</PeakNo>
      <RetTime>0.082</RetTime>
      <Component></Component>
      <BL>BV</BL>
      <Area>844.82</Area>
      <AreaPercentage>0.02</AreaPercentage>
      <Height>156.44</Height>
      <Raw>0.0008</Raw>
    </PeakInfo>
    <PeakInfo>
      <PeakNo>2</PeakNo>
      <RetTime>0.166</RetTime>
      <Component></Component>
      <BL>VV</BL>
      <Area>868.41</Area>
      <AreaPercentage>0.02</AreaPercentage>
      <Height>157.28</Height>
      <Raw>0.0009</Raw>
    </PeakInfo>
    <PeakInfo>
      <PeakNo>3</PeakNo>
      <RetTime>0.298</RetTime>
      <Component></Component>
      <BL>VV</BL>
      <Area>349918.88</Area>
      <AreaPercentage>8.00</AreaPercentage>
      <Height>128235.44</Height>
      <Raw>0.3499</Raw>
    </PeakInfo>
    <PeakInfo>
      <PeakNo>4</PeakNo>
      <RetTime>0.375</RetTime>

```

Figure 4.21: An XML format of instrument raw data file

```

File Edit Format View Help
<InstrumentData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLo
  <GCData>
    <PeakInfo>
      ...
      <PeakNo>6</PeakNo>
      <RetTime>0.577</RetTime>
      <Component>Butyric</Component>
      <BL/>
      <Area>452138.38</Area>
      <AreaPercentage>10.33</AreaPercentage>
      <Height>91602.00</Height>
      <Raw>0.3499</Raw>
    </PeakInfo>
  </GCData>
  <SampleData>
    <Operator>zhuji</Operator>
    <SampleName>Palm oil Z1001</SampleName>
    <SampleID>002</SampleID>
    <Study>Palm oil Analysis</Study>
    <InstrumentName>NCI 901</InstrumentName>
  </SampleData>
  <Signature>
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
      <Reference URI="">
        <Transforms>
          <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>2LxVxHBD+3kstgxMRMt6j0tMDy8=</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>U3MyzP5YsPvxiXQ7KtrHnNAZYUDC37wz6NVTjD/Rw3rnZwwjwnue5dyevsDT2k0
qSVn9vbSh43TFACYUF3xvw==</SignatureValue>
    <KeyInfo>
      <KeyName>rsaKey.pem</KeyName>
    </KeyInfo>
  </Signature>
</InstrumentData>

```

Figure 4.22: A secured XML format of instrument data file

4.2.4. Storage of XML format of data in the archiving database

In this project, a repository was built to store secure XML format of instrument data files using the XML DB capability in Oracle 10g DBMS Release

1. The repository not only stores the XML data files, but it also allows the user to query the repository and retrieve the instrument records for review.

4.2.4.1. Oracle XML DB

XML DB is Oracle's high-performance, native XML storage and retrieval technology available from the release of Oracle9i database server. It provides a unique ability to store and manage the structured data and handle the interchangeability between the XML and SQL metaphors [16]. Some new or improved XML DB features of Oracle10g such as deep support for XML Schema and XMLType, the XML DB Repository feature, XPath, and XQuery were used in this project. By combining the power of XML and SQL, the middle layer, backed by Oracle 10g database server, is able to deliver powerful, flexible capabilities to store and query XML content. Some of the features in the middle layer are listed below:

- Collect data and store them as nodes and fragment in an XML document
- Update nodes and fragments within an XML document
- Create indexes on specific nodes within an XML document

- Index the entire content of an XML document
- Determine whether an XML document contains a particular node

Since Oracle XML DB is compatible with Oracle DBMS, the functionalities in the XML DB package are seamlessly incorporated into Oracle database systems. It allows the XML developers to use the natural conventions or tools of XML, transparently store and retrieve their XML documents in an Oracle database. As for the database administrator, it brings the same performance and manageability from a database perspective working with XML. In addition, there is no need to run through a separate installation process to set up an XML DB package if a 9i or updated version of Oracle DBMS is used.

4.2.4.2. Storage of XML data as XMLType

Before XML data was stored to Oracle DBMS using XML DB technology, the process of selecting the storage mechanism had to be conducted [17] as shown in Figure 4.23.

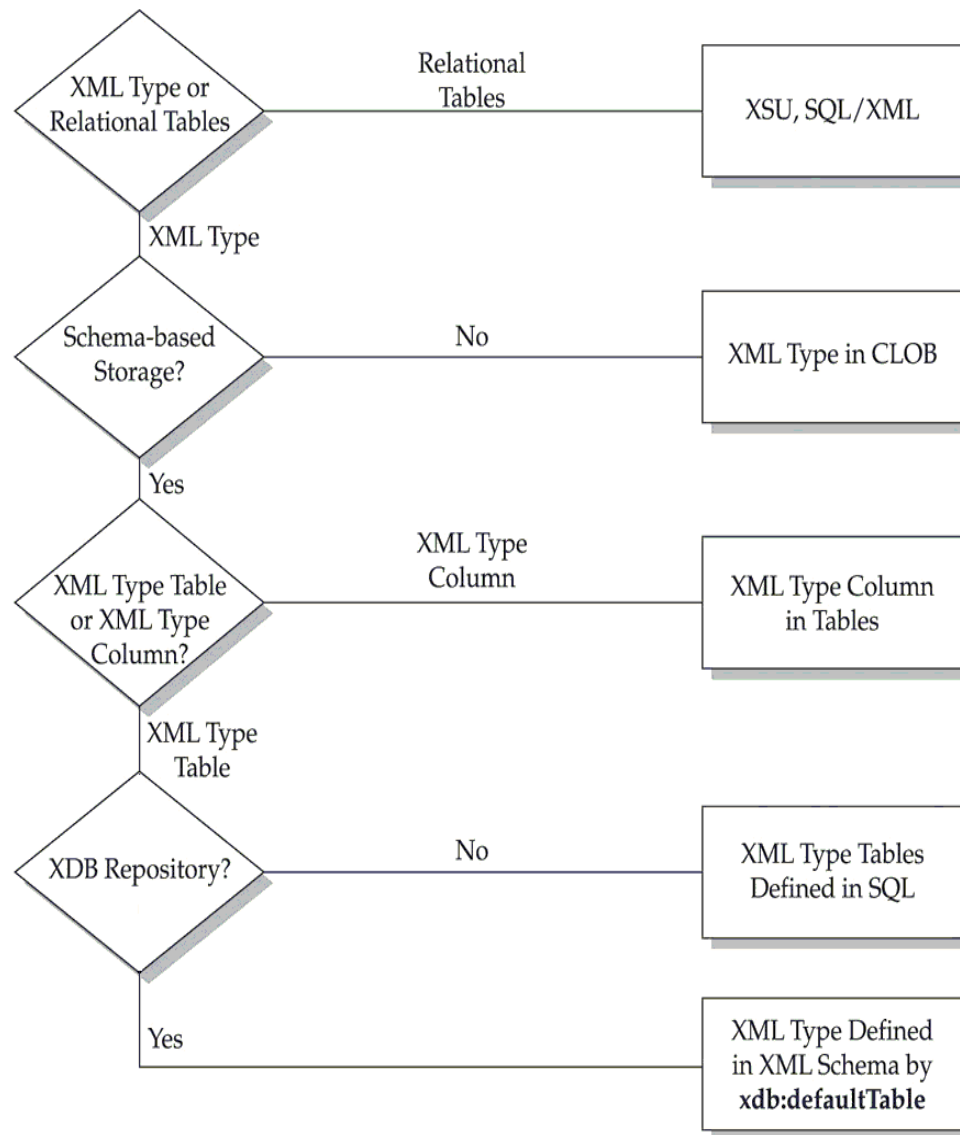


Figure 4.23: Storing XML data summary

1) XMLType or relational tables

Storage of XML data in XMLType or relational tables depends on XML data format and Document Object Model (DOM) fidelity requirement on XML content.

XML document can be classified into two types of format: data-centric and document-centric format [18]. The differences between these two formats are listed below:

- Data-centric XML document
 - A regular structure is in this type of XML document
 - The document has little or no mixing of content.
 - DOM fidelity for the document is not required to be preserved.
- Document-centric XML document
 - A less regular or irregular data structure.
 - The content has some degree of mixing.
 - Many queries are required on XML document.
 - DOM fidelity needs to be preserved.

Preserving Document Object Model (DOM) fidelity of XML document means that the in-memory representation in DOM of the original document was identical after storage and retrieval from database. This includes maintaining the order in which elements appear within a collection and within a document, as well as storing and retrieving out-of-band data such as comments, processing instructions, and mixed text. Providing DOM fidelity means that the

loss of information is not occurred no matter which database system is used to store and manage XML documents.

If XML document is data-centric, relational tables are chosen to store the data. If it is document-centric, native XMLType is used to stored XML document. By implementing XMLType, XML document is stored into Oracle DBMS as an XML object, which not only maintains the DOM fidelity of XML documents, but allows users to query the database using XPath, XQuery and other native XML technologies.

For the secure XML instrument document in the middle layer developed in this research, it is defined as a document-centric format (Figure 4.24). The reasons why it is defined as document-centric format are as follows:

- Since instrument data format varied among different vendors and even different models from the same vendor, there is less regular data structure for secured XML document.
- In the high level hierarchy of the secure XML document, there are two regular objects which are the sample information and raw data information. However, below these two objects on the hierarchy, there is no certainty of how the data will be structured.

- The other elements in secured XML documents (such as signatures, digest data) add more complexity of the data structures and mixing content.
- Since the document is signed with the digital signature, it is required to maintain DOM fidelity so that the secured document can be correctly validated when needed.

2) XML Schema-based storage or non-Schema based storage

As indicated in Figure 4.24, there are two storage options when XML is stored into XMLType table.

- Non XML Schema-based XMLType
 - XML data is stored as CLOB by default.
 - It is optimum for DTD-based document.
- XML Schema-based XMLType
 - XML documents are shredded and stored as a set of SQL objects.
 - It is optimum for fine-grained data queries and retrievals.
 - It can preserve DOM fidelity.
 - It can perform XML instance validation according to the XML Schema.

```

File Edit Format View Help
<InstrumentData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLo
  <GCData>
    <PeakInfo>
      ...
      <PeakNo>6</PeakNo>
      <RetTime>0.577</RetTime>
      <Component>Butyric</Component>
      <BL/>
      <Area>452138.38</Area>
      <AreaPercentage>10.33</AreaPercentage>
      <Height>91602.00</Height>
      <Raw>0.3499</Raw>
      ...
    </PeakInfo>
  </GCData>
  <SampleData>
    <Operator>zhuji</Operator>
    <SampleName>Palm Oil Z1001</SampleName>
    <SampleID>002</SampleID>
    <Study>Palm Oil Analysis</Study>
    <InstrumentName>NCI 901</InstrumentName>
  </SampleData>
  <Signature>
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
      <Reference URI="">
        <Transforms>
          <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>2LXvkHBD+3kstgxMRMt6j0tMDy8=</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>U3MyzP5YsPvXiQ7KtrHnNAZYUDC37wz6NVTjD/Rw3rnZwwjwnue5dyevsDT2k0
qSVn9vb5h43TFACYUF3xvw==</SignatureValue>
    <KeyName>rsaKey.pem</KeyName>
  </Signature>
</InstrumentData>

```

Figure 4.24: A secured XML format of instrument data file

Since the middle layer allows the user to review and query the XML format of instrument documents, well-defined data queries need to be built into the application. Thus, XML Schema-based XMLType was chosen to store XML document as SQL objects. The front-end can build queries in SQL to retrieve the data by this technique. Again, only the Schema-based XMLType is able to preserve DOM fidelity of the secure XML documents.

3) XMLType table or XMLType column

XML documents can be stored in the database in two different forms. One is XML Type table and the other is XML Type column table. If the XML document is created as table, there are no more columns can be added to the table. If XML document is created as an XMLType column, other columns could be added to the same table. However, there is no difference between these two options in terms of functionality. XMLType was eventually used in the project as it gives the user an option to add other columns if needed.

4) XML DB Repository

The repository feature is the most important reason that XML DB was used to archive the secured XML documents. With secure XML documents stored in XML DB repository as XMLType, it ensures that the queries could be built in the middle layer interface using XPath and XQuery to provide users with the options to query the instrument data [19].

4.2.4.3. XML DB Schema creation and generation

1) Create XML Schema

XML Schema describes the structure of a set of XML documents. In this project, the XML Schema is used to describe the structure of data generated from analytical instruments. The schema is thus created

(Figure 4.25) based on the secure XML format of instrument data file generated from preceding steps, as shown in Figure 4.24.

The W3C XML Schema generation conventions need to be followed when creating the schema. For example, we need to define the namespace of the schema. Then, we need to pay attention to the data type of the schema required by Oracle XML DB. The data type refers to as simpletype or complextype definition. Complextype is used to describe hierarchy of an object. In Figure 4.25, the instrument data root element includes three elements. Therefore, the instrument data element is created as complextype named instrumentdatatype. Again, since gcdata element also includes some elements, it will also be created as complextype. However, since peakno element doesn't include any other elements, it is created as simpletype.

In order to enable the query capability of XML DB, schemas must be annotated. These annotations control how XML documents are stored in the database. The attributes for the annotations are in different namespace than the Schema attributes. The most used annotations are defaultTable, SQLName, SQLType, SQLCollType, maintainDOM, storeVarraysAsTable.

```

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xdb="http://xmlns.oracle.com/xdb" version="1.0">
  <xs:element name="InstrumentData" type="InstrumentDataType" xdb:defaultTable="INSTRUMENTDATA"/>
  <xs:complexType name="InstrumentDataType" xdb:SQLType="XDBPO_TYPE">
    <xs:sequence>
      <xs:element name="GCData" type="GCDataType" xdb:SQLName="GCData"/>
      <xs:element name="SampleData" type="SampleDataType" xdb:SQLName="SAMPLEDATA"/>
      <xs:element name="Signature" type="SignatureType" xdb:SQLName="SIGNATURE"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="GCDataType" xdb:SQLType="XDBPO_GCData_TYPE">
    <xs:sequence>
      <xs:element name="PeakInfo" maxOccurs="4" xdb:SQLName="PEAKINFO" xdb:SQLCollType="XDBPO_PE"
        <xs:complexType xdb:SQLType="XDBPO_PEAKINFO_TYPE">
          <xs:sequence>
            <xs:element ref="PeakNo" minOccurs="0"/>
            <xs:element ref="RetTime" minOccurs="0"/>
            <xs:element ref="Component" minOccurs="0"/>
            <xs:element ref="BL" minOccurs="0"/>
            <xs:element ref="Area" minOccurs="0"/>
            <xs:element ref="AreaPercentage" minOccurs="0"/>
            <xs:element ref="Height" minOccurs="0"/>
            <xs:element ref="Raw" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="PeakNo" xdb:SQLName="PEAKNO" xdb:SQLType="VARCHAR2" xdb:defaultTable="">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:minLength value="0"/>
        <xs:maxLength value="10"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="RetTime" xdb:SQLName="RETTIME" xdb:SQLType="VARCHAR2" xdb:defaultTable="">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:minLength value="0"/>
        <xs:maxLength value="24"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="Component" xdb:SQLName="COMPONENT" xdb:SQLType="VARCHAR2" xdb:defaultTable="">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:minLength value="0"/>
        <xs:maxLength value="256"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>

```

Figure 4.25: Complex type and annotations used in schema definition

2) Register XML schema in Oracle database

The registration process requires a single command using the Oracle XML DB `dbms_xmlschema.registerSchema ()` function [20], as shown below:

```
begin
dbms_xmlschema.registerSchema (
    schemaurl=>'http://in-info-
poll10a:8080/home/OTNXML/source/xsd/instrumentData.xsd',
    schemadoc=>bfilename('OTNXML_DIR','instrumentData.xsd'),
    local=>TRUE,
    gentypes=>TRUE,
    genbean=>FALSE,
    force=>FALSE,
    owner=>'OTNXML',
    csid=>nls_charset_id('AL32UTF8')
);
end;
/
```

Once the command executes, Oracle XML DB will automatically shred XML documents conforming to this schema into relational tables.

4.2.4.4. Loading secure XML instrument files

A validation process against the schema registered in previous step is performed when loading secure XML documents to Oracle XML DB. The statement could look similar as follows:

```
insert into InstrumentData values  
  
( XMLType  
  
  ( bfilename('OTNXML_DIR', nci90120041118001.xml'),  
  
    nls_charset_id('AL32UTF8') )  
  
);
```

If the XML document does not conform to the schema, an error message is generated to indicate where the problem comes from. The users can go back and modify either the schema or the XML document.

4.2.4.5. Querying XML instrument data from database

Although Oracle XML DB is fundamentally based on Oracle Database relational architecture to make the familiar SQL paradigm continue to work well, only the SQL with some functional extensions is able to query XML documents which have the hierarchical nature.

XPath, a query language designed specifically for XML is used in this project. XPath is recommended as a query language for XML document by W3C. Figure 4.26 shows an example of using XPath to query the secure XML instrument document repository. The ordinary SQL is mixed with the Oracle

XML DB function existsNode() and an XML XPath statement (/InstrumentData/SampleData [SampleID = "002"]') to determine if any of the XML instrument data match a specific sample id "002".

These XPath notations and SQL functions existsNode() and extract() are part of a set of extensions to SQL that Oracle has implemented in Oracle10g Database to the emerging SQL XML specification. The goal of SQL XML is to enable SQL to query XML and also to generate XML from SQL.

In this project, users are provided with search criteria such as sample id, sample name, instrument name and project/study name. And the result returned from query is a complete secure XML instrument document.

```
SQL> select extractvalue(object_value, '/InstrumentData/SampleData/SampleID') SampleID,
2  extractvalue(object_value, '/InstrumentData/Signature/KeyInfo/KeyName') KeyName,
3  extractvalue(object_value, '/InstrumentData/SampleData/SampleName') SampleName
4  from InstrumentData
5  where existsNode(object_value, '/InstrumentData/SampleData[SampleID = "002"]') = 1;
```

SAMPLEID	KEYNAME
002	rsakey.pem
Palm Oil Z1001"	

Figure 4.26: Querying XML DB using XPath

4.3. Performance

4.3.1. Parsing and processing XML files

Processing of XML files is performed by using DOM. Although powerful and flexible, DOM is memory-intensive because the entire XML document has to be loaded into an in-memory representation. To reduce the memory overhead associated with DOM, Oracle XML DB automatically uses an optimized in-memory structure called an XML Object (XOB) [21]. In our project, we have one object called instrument data which handles both sample information and raw data information.

An XOB is much smaller than the equivalent DOM. It does not duplicate information such as tag names and node types that can easily be obtained from the associated XML Schema. The use of the XOB is transparent to the application developer. It is accessible through the XMLType data type and the C, PL/SQL, and Java application-programming interfaces (APIs). This means, once having defined the instrument data XOB for each instrument, the developer does not have to worry about creating those objects in the database. The database administrator will take care of the creation and management of those objects. In our project administrator uses the PL/SQL to create instrument data XOB as XMLType and load the data to Oracle DBMS.

The XOB is also able to reduce the amount of memory required to work with an XML document via a feature called the Lazily-Loaded Virtual DOM [21]. This feature defers the loading of the in-memory representation of the XML document nodes that are part of sub-elements or a collection until some methods attempt to operate on a node within that object.

From a storage perspective, when XML documents are stored in Oracle XML DB using the structured storage option, the shredding process discards all of the tags and white space contained in the document. If the document has a high ratio of tags to data, the space savings can be significant. This doesn't mean that the information contained in the tags is lost. When a document is extracted, or when node-based operations such as XPath evaluations take place, Oracle XML DB uses the information contained in the associated XML Schema to dynamically reconstruct any necessary tag information.

4.4. Summary

In this project, a middle layer was developed to retrieve data from analytical instruments, convert data from ASCII delimited format to XML format. Digital signatures can be signed on the XML format of instrument data, which results in the creation of secure XML format of instrument data.

On the interface of middle layer, user is able to review the data either in XML format or spreadsheet format. Then user can start to store data to LIMS database. The middle layer is also able to archive the instrument raw data in an Oracle DBMS during the process of data storage.

The data stored in the archiving database is in secure XML format. A schema was created to validate the instrument document when the document is stored into database. The middle layer also provides the user with options to query the archive database for the instrument data.

5. CONCLUSION

It is possible to build a generic middle layer, however, it is not trivial task. The middle layer built in this project is able to perform the proposed tasks including reading data from the instruments, reviewing and storing data to the LIMS database. In addition, the interface is able to archive the data to Oracle10g interface. However, the middle layer is only working well with the instruments that generate similar format of data. It is mostly due to the difficulties of making a generic transformation interface that is able to convert all the different formats of instrument data files to XML format. For the middle layer to work with other types of instrument data, a hand-coded transformation style sheet has to be created for the newly added instrument. Hence, the efficiency of interfacing the instrument and LIMS is not improved compared to creating different drivers for each instrument. Another concern is that the schema needs to be modified when storing data to Oracle database for the different format of instrumental data.

5.1. Overview of significant findings

5.1.1. Parsing ASCII delimited file to XML file on Microsoft .Net platform

The middle layer is built on the Microsoft .Net platform, so does the ASCII delimited file to XML file parser that is used to parse the instrumental data ASCII delimited format into XML format.

The instrumental data file could be comma-delimited, tab-delimited or other delimited formats. In the project, the parser designed is generic enough to

handle different types of delimited ASCII files, which makes it possible to build a generic interface that converts all instrumental data formats to XML format and then store the data in LIMS database.

5.1.2. Storing of secure XML data in Oracle 10g XML DB

XML document is orders of magnitude larger in the string format than equivalent binary format. The cost of processing string-based data over native database formats may bring concerns to many administrators. Fortunately, both storage size and processing cost issues have been addressed in Oracle XML DB [22]. This is a major reason why using of structured XMLType tables in Oracle XML DB is preferred to store the secure XML instrument data.

Because of the instrument data validation and querying functionality, a well-designed XML schema is the first step when developing the middle layer. It is necessary to spend more time to design and refine the XML schemas than create applications to communicate with the database. Therefore, the structure and content of the XML instance documents are able to achieve a natural data granularity and grouping that could better fit the analytical data main access and usage patterns gathered from the instrument users.

Overall, in spite of various ramp-up issues with different versions of Oracle XML DB that have been tested in this project, XML DB is still preferred because it enabled us to deliver a working application with less complexity, more flexibility, and better performance than any file sharing system on the secured

network as the storage device that have been previously tested. For example, one column XMLType table is able to store all the xml data. In summary, Oracle XML DB enabled us to use XML effectively as the core technology for designing and developing the middle layer.

The transparent duality between XML metaphors and SQL metaphors is impressive when evaluating and using Oracle XML DB, Through the XPath support, XML remains transparent. It seems that the documents themselves were on a local file system. For the purpose of querying data from Microsoft .Net platform, all the power of SQL is available to work with XML documents. Thus, we are able to offer users options to query data with different criteria.

5.2. Consideration of findings in context of current knowledge

Developing an application on Microsoft .Net platform along with XML related technologies is gaining popularity in the scientific software development area. However, most of scientific data are still generated in the proprietary format with an option of being converted to ASCII delimited format. Therefore, the creation of generic ASCII delimited file to XML file parser is will play a significant role in developing an XML powered laboratory automation application on the Microsoft .Net platform.

Originally, XML was designed as a document format without considering storage and system integration requirements. For example, in the beginning, XML was used as a tool to transport documents in this project aimed to integrate analytical

instruments and LIMS; particularly, in transporting data from instruments to either archiving database or production database (LIMS database). XML was able to handle the problem that cannot be solved using ASCII data format. Thus, ASCII to XML parser plays a critical role in making this data transportation doable. At a time when RS232 is still the main mode of transporting instrument data, the creation of ASCII to XML parser is magnificent in helping instrument interfacing technologies to move forward.

Archiving instrument data in the Oracle10g DBMS helps us to better understand the ability of Relational Database Management System (RDBMS) in supporting XML parsing and processing. Particularly, the fast-improving proprietary database systems, such as XML DB from Oracle, are providing more comprehensive functionalities to handle XML documents. In return, it facilitates the use of XML in building the content management applications in scientific data management fields. Also, with the help of ASCII to XML parsing technologies, we have the abilities to store the instrument data in a secure repository such as RDBMS.

6. DISCUSSION

6.1. Limitations of the study

6.1.1. Limitation of XML parsing and transforming

When parsing and transforming the XML data, an Extensive Stylesheet Language Transformation (XSLT) file is required for each instrument that has a

different schema for data. This is because each XSLT file is built upon the schema of the instrument data. There is not a generic XSLT file that is able to parse and transform the XML data from any instrument. Thus, one has to either create XSLT files for all the different instrument data or develop a comprehensive schema that is able to address the characteristics of data from any instrument. Thus, for this project, additional XSLT files have to be created for each new format of data.

6.1.2. Limitation of knowledge on XML storage

The secure XML instrument documents would be stored in the file systems on the secured network as stated in the original proposal. However, it turned out the file system did not scale up very well to handle situations when hundreds or thousands of XML documents have to be managed. In the meanwhile, the file system did not offer good tools to perform search queries and the search queries could not directly access the XML data model. Instead, it only queried the XML as a plain text file ignoring the structure of the document, which means there is no relationship between instrument files. Particularly, after we parsed the original instrument file into two separate files – sample information file and raw data information file, it was impossible to query those two files in the file system if we wanted to merge them and signed the digital signatures.

Therefore, Oracle DBMS XML DB was then used to store the XML documents. Oracle9i version of XML DB was first tested. However, because of the poor XML functionality support provided by Oracle9i version, the XML document against the XML schema of instrument data were still not able to be

validated resulting in the poor performance of database querying. Oracle10g DBMS was then used to archive database. Compared to Oracle9i version of XML DB, Oracle10g DBMS performs very well on the tasks such as instrument XML data validation and data querying.

6.1.3. Lack of dedicated computer systems

Two computer systems were loaded with two versions of Oracle DBSM server, Microsoft IIS 5.0 server and one other application server. Different systems running on the same computer system not only affects the performance of each system but prevents some systems from behaving properly.

6.2. Recommendations for further research

6.2.1. Design and manage XML format of instrument data using a popular interchange format

For the middle layer developed in the project, a schema for one instrument has to be created for integrating this instrument into the system every time. Although instrument groups have been created, such as GCMS and LCMS, schemas at group level have not been set up yet. If a group level of schema is available, we could just inherit from the group level schema when creation a new instrument instance is necessary. Additionally, the instrument schemas, created

for the middle layer, are very much lack of scalability by far. Therefore, for the future improvement of this application, to create group level schema using a widely recognized analytical data interchange format standard (e.g. Analytical Instrument Markup Language (AnIML), Generalized Analytical Markup Language (GAML)) is recommended.

Take the Analytical Information Markup Language (AnIML) as an example, AnIML is being developed by ASTM subcommittee E13.15 on Analytical Data Management as a "web-aware" mechanism for instrument-to-instrument, instrument-to-application, and application-to-application data interchange and archiving [23]. AnIML is based partially on the NIST's SpectroML markup language for UV-VIS spectroscopy data and Thermo Electron's Generalized Analytical Markup Language (GAML) and borrows mainly from older interchange standards such as IUPAC, JCAMP-DX and ASTM, ANDI, from existing data dictionaries, and from other relevant markup language efforts. AnIML is comprised of two major parts: the AnIML Core and Technique Layers. The structure of AnIML Core is described by an XML (Extensible Markup Language) schema, provides ways to organize and represent arbitrary analytical data. AnIML Technique Layers formally define the structure of data and metadata for specific analytical techniques and take the form of XML instance documents such as the secure XML analytical instrument documents used in the middle layer. The Technique Schema provides the meta-representation for the Technique Layers in AnIML, which are extensible to permit vendor,

enterprise, and/or user extensions to the data representation. In the middle layer designed in this project, there is a project setting function into where users can add instrument information such as vendor and owner. If the AnIML techniques could be implemented, it will not only help to manage the instrument group level of schemas, but incorporate other instruments and projects related to metadata into the system.

There are some software tools available and free to use. The time spent on design schemas in this project could be enormously reduced if we took advantage of these software tools. For example, one software tool (the AnIML Validator) [24] has been created to check different formats of instrument data for completeness and proper syntax. Another software tool (the AnIML Technique Creator) [24] allows analytical experts who lack XML expertise to create Technique Layer instance documents. Accordingly, it is easier for application customers to take over the job of adding new instrument to the middle layer. Adherence to the AnIML standard permits the creation of generic data viewers that can function inside XML-aware programs, such as Microsoft's Internet Explorer. With such a viewer, a user should be able to look at result data from any analytical instruments in AnIML format.

REFERENCES

- [1] Smith, T. (2000, July 18). Information Management System for Molecular Biology. Retrieved March 2, 2005, from <http://www.geospiza.com/rc/white-papers-research.htm>
- [2] Perry, D. (2004). Laboratory Information Management Systems, guess lecture for Indiana University School of Informatics, course number C571
- [3] Liscouski, J. (1995). Laboratory and Scientific Computing: a Strategic Approach. New York: John Wiley and Sons, Inc.
- [4] McDowall, R.D. (2004, February 27). Chromatography Data System I: The Fundamentals. Retrieved February 5, 2005, from <http://www.21cfrpart11.com/files/library/compliance/>
- [5] Valle, M. Scientific Data Management. Retrieved February 5, 2005, from <http://www.cscs.ch/~mvalle/sdm/index.html>
- [6] An introduction to LIMS. Retrieved October 18, 2004, from <http://www.limsources.com/intro.html>
- [7] 21 Code of Federal Regulations (21 CFR Part 11) Electronic Records; Electronic Signatures Final Rule. (1997, March 20). The Federal Register. Retrieved February 27, 2004, from http://www.fda.gov/ora/compliance_ref/part11/frs/background/11cfr-fr.htm
- [8] Hirsh, F. (2002, November 28). Getting Started with XML Security. Retrieved March 2, 2004, from <http://www.sitepoint.com/article/getting-started-xml-security>
- [9] Eastlake III, D. E., Niles, K. (2003). Secure XML. Boston: Addison-Wesley.

- [10] LabVantage Products: Sapphire LIMS. Retrieved October 5, 2004, from <http://www.labvantage.com/products/sapphire/index.html>
- [11] About Labware. Retrieved October 5, 2004, from <http://www.labware.com/lwwweb.nsf/lp/en01>
- [12] XML and Relational DB. XMLEverywhere.com (2001, July 1). XML Persistency. Retrieved October 10, 2004, from <http://www.xmleverywhere.com/WhitePapers/persistence.htm>
- [13] XMLEverywhere.com (2002, May 23). Internet Data Interchange Retrieved October 10, 2004, from <http://www.xmleverywhere.com/WhitePapers/dataInterchange.htm>
- [14] XMLEverywhere.com (2002, January 29). Retrieved October 10, 2004, from <http://www.xmleverywhere.com/WhitePapers/xml-acceptor.html>
- [15] Miloslav, N., Jirat, J. XPath Tutorial. Retrieved October, 2004 from <http://www.zvon.org/xxl/XPathTutorial/General/examples.html>
- [16] Bourett, R. (2001, June). XML and Databases. Retrieved January 15, 2005, from <http://www.rpbouret.com/xml/XMLAndDatabases.htm>
- [17] Osborn, S. Oracle XML Database Lectures. Retrieved January 20, 2005 from <http://www.csd.uwo.ca/courses/CS853b/>
- [18] Obasanjo, D. (2001). An Exploration of XML in Database Management System. Retrieved January 15, 2005, from <http://www.25hoursaday.com/StoringAndQueryingXML.html>
- [19] Clark, J., Steve D. (1999, November 16). XPath: XML Path Language (Version 1.0). Retrieved February 23, 2005, from <http://www.w3.org/TR/1999/REC-xpath-19991116>
- [20] Oracle9i XML Database Developer's Guide – Oracle9i XML DB. Retrieved February, 2005 from

http://download-west.oracle.com/docs/cd/B10501_01/appdev.920/a96620/toc.htm

[21] Lehmann, M. (2003, March). From XML to Storage and Back. Retrieved February 24, 2005, from <http://www.oracle.com/technology/oramag/oracle/03-mar/o23xml.html>

[22] Oracle Technology Network. (2005, March). Mastering XML Generation – Oracle10g XML DB Release 2. Retrieved March 12, 2005, from http://www.oracle.com/technology/tech/xml/xmlldb/Current/TWP_Mastering_XML_Generation.pdf

[23] Analytical Information Markup Language (AnIML). Retrieved March, 2005, from <http://animl.sourceforge.net/>

[24] ASTM Committee E13.15 (2004, July 8). Minutes of AnIML Working Group of ASTM Committee E13.15. Retrieved March, 2005, from http://animl.sourceforge.net/E13.15_Minutes%2020040707.doc

Jianyong Zhu

EDUCATION

08/2003–06/2005 Indiana University, Indianapolis, IN

M.S., Chemical Informatics, specializing in Laboratory Informatics

01/2001–08/2002 Franklin University, Columbus, OH

B.S., Computer Science

09/1992–08/1996 Qingdao Institute of Chemical Technology, Qingdao, China

B.S., Fine Chemical Engineering

Minor, Computer Science

PROFESSIONAL EXPERIENCE

01/2004–05/2005 **AIT Laboratories**, Indianapolis, IN

Information Technology Intern

- Conducted project integrating NuGenesis Scientific Data Management System (SDMS), ChemWare Horizon Laboratory Information Management System (LIMS) and analytical instruments.
- Conducted the projects archiving ISO 9000 and regulatory compliant documents using NuGenesis Scientific Data Management System (SDMS)

- Involved in Chemware Horizon LIMS routine work, such as system trouble shooting, data mining and reporting by writing PL/SQL scripts and customizing Brio Report
- Involved in the installation of Labtronics LimsLink to automate data entry from instruments to ChemWare Horizon LIMS
- Involved in I.T. routine trouble shooting, such as Windows 2000 Server Active Directory configuration, Oracle database optimization, hardware, software and networking related work

01/2004–present **Indiana University, School of Informatics**, Indianapolis, IN

Research Assistant

- Conducted research on the area of integrating varied laboratory applications, for example, LIMS (LabWare LIMS and LabVantage Sapphire LIMS), SDMS (NuGenesis), ECMS (CyberLab), Data Mining and Visualization Tools (SpotFire), CDS (Perkin Elmer TotalChrom), Laboratory Electronic Notebook (LabTrack)

09/2001–08/2003 **Franklin University**, Columbus, OH

Tutor

- Tutored most of the core courses in computer science major while attending Franklin University, including C++, Java, Database Design, Client/Server Programming with Java, Web Design with MS FrontPage and Adobe Photoshop, Visual Basic, Computer Networks, Software Engineering, etc.
- Taught learning sessions such as Data Structure, Algorithm Analysis, etc.

08/1998–09/2000 **Rhein Chemie (Qingdao) Ltd., a Bayer company**, Qingdao, China

Laboratory Supervisor

- Managed the QC laboratory
- Responsible for purchasing and installing instruments for the laboratory
- Responsible for recruiting and training new employees
- Developed new products for the Asia market along with experts of marketing and R&D departments from the headquarters in Germany
- Developed a database using MS Access to store analytical methods and testing result

02/1998–06/1998 **Rhein Chemie Rheinau GmbH**, Mannheim, Germany

Trainee

- Trained in the QC and R&D laboratories in preparation for building the QC laboratory in Qingdao.
- Conducted analysis of the raw material collected from the suppliers in Asia
- Conducted analysis of the products collected from the competitors in Asia

09/1997–07/1998 **Rhein Chemie (Qingdao) Ltd., a Bayer company**, Qingdao, China

Manufacturing Manager Assistant

- Responsible for building the laboratory for the new Qingdao division of Rhein Chemie Ltd. along with experts from Germany
 - Designed the QC laboratory
 - Responsible for purchasing the equipment of QC laboratory

- Compiled the analysis methods and prepared the instruments purchasing requisitions
- Involved in visiting and evaluating potential raw material suppliers
- Accompanied guests from Germany to visit departments of local government as an interpreter for the issues of tax, finance, working permission of foreign experts, etc.

09/1996–08/1997 **Red Star Chemical (Group) Co.**, Qingdao, China

Trainee

- Trained in the workshops and laboratories for the placement in a certain field

02/1996–04/1996 **SINOPEC Qilu Petrochemical Co., Ltd.**, Shandong, China

Intern

- Interned in the workshops of oil refinement department and fine chemical department.

01/1996–02/1996 **Qingdao Soda Co. Ltd.**, Qingdao, China

Intern

- Got familiar with the diaphragm and ion membrane electrolysis process of caustic soda, chloride, and hydrogen.

VOLUNTEERING EXPERIENCE

06/2003–Present **American Chemical Society Columbus Section**, OH

Website Designer/Updater

- Designed and updated the ACS Columbus section's website

04/2003–07/2003 **The Ohio State University**, Columbus, OH

Website Designer

- Designed a website for Professor Linda K. Weavers of the Department of Civil and Environmental Engineering and Geodetic Science

HONORS AND AWARDS

01/2004–05/2005 Presidential Internship, American Institute of Toxicology (AIT) Laboratories

06/12/2004-06/16/2004 Association for Laboratory Automation (ALA) Academic Travel Award for ALA LabFusion 2004 Conference, Boston, MA

09/2003–05/2004 Daylight Innovation in Chemical Informatics Fellowship, Indiana University

01/2001–08/2002 Franklin University President Honored Student Lists, Franklin University

PROFESSIONAL SOCIETIES

American Chemical Society, student member

Association for Laboratory Automation, student member