

2006

## Establishing A Web-Based Integration Module In .Net And Labview Environment

Yuqiu You  
*Indiana State University*

Follow this and additional works at: <https://scholars.indianastate.edu/etds>

---

### Recommended Citation

You, Yuqiu, "Establishing A Web-Based Integration Module In .Net And Labview Environment" (2006). *Full List of Electronic Theses and Dissertations*. 989.  
<https://scholars.indianastate.edu/etds/989>

This Dissertation is brought to you for free and open access by Sycamore Scholars. It has been accepted for inclusion in Full List of Electronic Theses and Dissertations by an authorized administrator of Sycamore Scholars. For more information, please contact [dana.swinford@indstate.edu](mailto:dana.swinford@indstate.edu).

## VITA

### EDUCATION

- MS                      Industrial Education and Technology  
Morehead State University, Morehead, KY  
May 2002
- BE                      Automation Engineering  
Huazhong University of Science and Technology, P.R.China  
July 1995

### PROFESSIONAL EXPERIENCE

Assistant Professor    Department of Industrial Engineering and Technology, Morehead State University, Morehead, KY. August 2005 to Present.

Research Assistant    Department of Electronics and Computer Technology, Indiana State University, Terre Haute, IN. January 2004 to present.

Graduate Assistant    Department of Manufacturing Systems, North Carolina A&T State University, Greensboro, NC. January 2002 to May 2003.

Graduate Assistant    Department of Industrial Education & Technology, Morehead State University, Morehead, KY. January 2000 to December 2001.

Engineer                Tianshi Air-conditioning Technology & Engineering Inc.  
Chengdu, China. August 1995 to December 1999.

### HONORS

- First Place Prize for Research Presentation Competition in NAIT, 2002
- Future Venture Grant from NC A&T , 2002

### PUBLICATIONS

- *Implementing LabVIEW in establishing remote control laboratory*, NAIT Selected Papers, 2004.
- *Remote logix control*, NAIT Selected Papers, 2004.



ESTABLISHING A WEB-BASED INTEGRATION MODULE IN .NET  
AND LABVIEW ENVIRONMENT

---

A Dissertation  
Presented to  
The School of Graduate Studies  
College of Technology  
Indiana State University  
Terre Haute, Indiana

---

In Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy

---

By  
Yuqiu You  
May 2006

UMI Number: 3220264

### INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

**UMI<sup>®</sup>**

---

UMI Microform 3220264

Copyright 2006 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company  
300 North Zeeb Road  
P.O. Box 1346  
Ann Arbor, MI 48106-1346

School of Graduate Studies  
Indiana State University  
Terre Haute, Indiana

CERTIFICATE OF APPROVAL

DOCTORAL DISSERTATION

This is to certify that the Doctoral Dissertation of

Yuqiu You

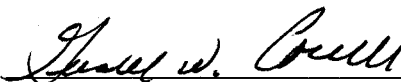
entitled

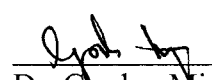
Establishing a Web-based Integration Module in .NET  
and LabVIEW Environment

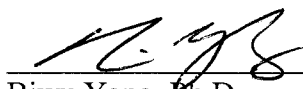
has been approved by the Examining Committee for the dissertation requirement for the


Doctor of Philosophy

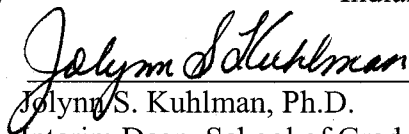
in Technology Management (Manufacturing Systems)  
May 2006

 3/22/06  
Gerald W. Cockrell, Ed.D. Date  
CHAIR  
Indiana State University

 3/23/06  
Dr. Gordon Minty, Ph.D. Date  
Member  
Indiana State University

 3/16/06  
Biwu Yang, Ph.D. Date  
Member  
East Carolina University

 3/22/06  
Ming Zhou, Ph.D. Date  
Member  
Indiana State University

 4/12/06  
Jolynn S. Kuhlman, Ph.D. Date  
Interim Dean, School of Graduate Studies  
Indiana State University

## ABSTRACT

The top problems faced by manufacturing enterprises to implement system integration solutions are confusing solutions and terminology, the lack of understanding of cross-domain technologies, and the lack of business justification. In this study, a generalized web-based partial module was established to interface between manufacturing control functions and higher management level functions in a manufacturing enterprise. It is composed of three parts, a LabVIEW-based data collector, a system server, and a web-based interface. The data collector was constructed as an open source system module for data collection from LabVIEW-based control applications. It can be integrated into LabVIEW VIs without requiring extra system resource from the control server. The web interface and data structure in the module are designed by using the terminology and methodologies from the Generalized Enterprise Reference Architecture and Methodology (GERAM) and ISA S95.

To evaluate the efficiency of the data collector, a queuing network model is established to analyze the effect of system change. The resource measurements are sampled from the same control server to analyze the effect of the data collector on the system resource. A statistical method, one-way ANOVA, is applied to evaluate the effect on the system. And SPSS statistical software is used for the statistical analysis.

## ACKNOWLEDGMENTS

The author wishes to acknowledge all those individuals who assisted in the completion of this dissertation. Their support and guidance toward successful completion of this dissertation is highly appreciated.

Sincere acknowledgement is expressed to Dr. William James, who passed away two months before the completion of this dissertation. As the advisor for my first two years in this program and my dissertation committee member, he helped and encouraged me all the time during this process. His useful and effective assistance was always a positive help toward finishing the dissertation.

Also, this dissertation could not have been finished without Dr. Gerald Cockrell, who not only served as my advisor but also encouraged and challenged me in the last two and half years in the academic program. He patiently guided me through the dissertation process, never accepting less than my best efforts.

Special thanks are extended to Dr. Gordon Minty, Dr. Biwu Yang, and Dr. Ming Zhou for their willingness to lend their educational and research expertise through constructive criticism and guidance during the completion of this dissertation.



## TABLE OF CONTENTS

	Page
ACKNOWLEDGMENT.....	iv
LIST OF TABLES.....	viii
LIST OF FIGURES.....	ix
 Chapter	
1. INTRODUCTION.....	1
General Area of Concern.....	1
Purpose of the Study.....	3
Significance of the Study.....	5
Definition of Terms.....	9
2. REVIEW OF LITERATURE.....	12
Historical Background.....	12
Architectural Models for Enterprise Integration.....	17
ISA S95 International Standards for Integration.....	20
.NET Technologies for Module Development.....	22
LabVIEW for the Development of Control System.....	23
Relational Database and Data Access.....	26
3. METHODOLOGY.....	30
Restatement of the Study Objectives.....	30

Overview of the Integration Module.....	31
<i>System Architecture</i> .....	31
<i>Interface Features</i> .....	33
Construction of the Data Collector.....	36
<i>LabVIEW Environment and Database</i> .....	36
<i>Database Concept</i> .....	37
<i>Components of the Data Collector</i> .....	37
<i>Integration of the Data Collector Module</i> .....	45
<i>Communication with the Database</i> .....	57
The Database Server and the Web-based Interface.....	64
<i>Introduction</i> .....	64
<i>Procedure Logic</i> .....	65
<i>Coding Structure</i> .....	67
<i>Web-based LabVIEW Control Panel</i> .....	69
Implementing Security.....	75
<i>Security Methods for the Three Components</i> .....	75
<i>Forms Authentication for the Web-based Interface</i> .....	76
Testing and Analysis.....	81
<i>Introduction</i> .....	81
<i>Queuing Network Modeling</i> .....	81
<i>Statistical Technique Used</i> .....	82
<i>Hypotheses</i> .....	84
<i>Assumptions and Limitations</i> .....	85

<i>The Experimental Design</i> .....	85
<i>Set the Type I Error Rate</i> .....	86
<i>The Population and Sample</i> .....	87
<i>Descriptive Statistics</i> .....	88
<i>Output Tables of the Statistical Test</i> .....	95
<i>Conclusion</i> .....	99
4. FINDINGS AND RECOMMENDATIONS.....	100
Introduction.....	100
Modular and Structured System.....	100
<i>The Preference on Modular and Structured Systems</i> .....	100
<i>The Template of a Structured System</i> .....	102
<i>The Personnel Involvement</i> .....	104
The Potential Development and Enhancement of the System.....	106
Recommendations for Future Study.....	108
REFERENCES.....	111

## LIST OF TABLES

Table	Page
1. SQL Commands Used by the Data Collector .....	40
2. Other ADO Components.....	45
3. ADO Connection SubVIs .....	47
4. ADO Command SubVIs .....	48
5. ADO Recordset SubVIs.....	50
6. I/O Addressing of the Control Process .....	71
7. Descriptive Statistics of CPU Usages in Each System .....	89
8. Descriptive Statistics of CPU Usages in Both Systems.....	91
9. Descriptive Statistics of the Number of Threads in Each System .....	92
10. Descriptive Statistics of the Number of Threads in Both Systems.....	94
11. Descriptive Statistics of CPU Usages for ANOVA Test.....	95
12. Homogeneity Test on CPU Usage Variable .....	95
13. ANOVA Test on CPU Usage Variable.....	96
14. Descriptive Statistics of Thread Numbers for ANOVA Test .....	97
15. Homogeneity Test on Thread Number Variable.....	97
16. ANOVA Test on Thread Number Variable .....	98

## LIST OF FIGURES

Figure	Page
1. Serial device network.....	13
2. Distributed device networks .....	14
3. Integration of production management systems .....	15
4. System architecture.....	31
5. The main interface window .....	33
6. Measurement data table .....	34
7. Device status data table.....	34
8. LabVIEW real-time control panel .....	35
9. Communication path between LabVIEW and a database.....	38
10. Communication path between ADO and SQL server.....	42
11. ADO object structure .....	44
12. SQL Execute VI.....	51
13. Block diagram of the SQL Execute VI.....	52
14. Automation Open Function VI .....	53
15. Invoke Node function .....	54
16. The Property Node function.....	56
17. Commands for SQL server security mode.....	59
18. Local system service window .....	59

19. SQL server authentication window.....	60
20. Data pane of the database interface .....	61
21. Design view of the database table.....	61
22. Block diagram of the database VIs .....	62
23. SQL statement format.....	63
24. DeviceStatus data table .....	64
25. Procedure logic for DataGrid control .....	65
26. DataSet for DevStatus table .....	67
27. Structure of VB coding .....	67
28. Example of event coding .....	68
29. Data binding procedure.....	69
30. Components of the wet process trainer .....	69
31. Process control diagram.....	73
32. Mode controls and digital indicators.....	74
33. Emergency Stop and Reset buttons.....	75
34. Web.config.....	78
35. The server-side login form.....	79
36. Coding example for authentication and authorization .....	80
37. Coding for interface security .....	81
38. The Task Manager window .....	88
39. Histogram of CPU Usages on the existing system .....	90
40. Histogram of CPU Usages on the modified system .....	90
41. Histogram of CPU Usages on both systems .....	91

42. Histogram of thread numbers on the existing system.....	93
43. Histogram of thread numbers on the modified system .....	93
44. Histogram of thread numbers on both systems.....	94

## Chapter 1

### INTRODUCTION

#### General Area of Concern

In today's new manufacturing environment, manufacturing enterprises are facing rapidly changing situations. To be competitive, enterprises must adapt to this change and evolve to be reactive so that changes become natural dynamic states rather than something forced onto the enterprise. This evolution requirement necessitates the need for enterprise integration with an increasing emphasis on agility. MEL (Manufacturing Engineering Laboratory) of NIST (National Institute of Standards and Technology), an agent for change in the fast-paced world of manufacturing, defined enterprise integration as providing the right information, at the right place, at the right time, and updating the information in real time to reflect the actual state of the enterprise operation (MEL, 1999).

Enterprise integration has been discussed since the early days of computers in industry in general and in the manufacturing industry particularly (MEL, 1999). In spite of the different understanding of the scope of enterprise integration from that time, the eventual vision for it is to be a tool for the enterprise operation supporting day-to-day decision making across the entire operation. This tool links decision makers on all organizational levels to relevant and real time information across the organizational boundaries. The implementation of enterprise integration requires explicit knowledge of both the information needed and created by the different activities in the enterprise



operation; requires information sharing systems and integration platforms capable of handling information transaction across heterogeneous environments; and also requires the up-date of the operational data as well as adapting to environmental changes.

However, MEL indicated the top problem associated with the applications of enterprise integration in its report on issues in enterprise integration in 1997. The problem is that the application of existing enterprise integration technologies has been hampered by the lack of business justification, the plethora of conflicting solutions and terminology, and by an insufficient understanding of the technology by the end users. Also, a recent survey in 2003, conducted by ARC Advisory Group, found that one of the top problems with IT applications in manufacturing enterprise integration is related to confusing solutions and terminology and the lack of understanding of cross-domain technologies (ARC, 2003). This especially inhibits, or at least delays, the use of relevant methods and tools in small-to-medium-sized enterprises.

In this study, a web-based partial module for implementation of enterprise integration is established to interface between manufacturing control functions and higher management level functions in a manufacturing enterprise. This web-based module provides a generalized model for integration applications of manufacturing enterprises on enterprise-control system integration. It is composed of three parts, ASP.NET web forms, LabVIEW control applications, and a dynamic database. The mechanism for retrieving, storing, and publishing real-time data among these three parts is the core method for building the module. The method solves the problem of communication between different applications and languages, and provides a way of getting real-time data from LabVIEW applications and publishing to web services. The implementation of this enterprise

module provides a template for enterprise-wide web applications to communicate with LabVIEW interfaced control and monitor processes in real time.

The web interface and data structure in the module are designed by using the terminology and methodologies from the Generalized Enterprise Reference Architecture and Methodology (GERAM) and ISA S95. GERAM is a generalized architecture model for enterprise integration methodologies, which is designed and published by IFIP-IFAC (International Federation for Information Processing-International Federation of Automatic Control) Task Force. ISA S95, published by ISA (the Instrumentation, Systems, and Automation Society), is a standard used to define the enterprise-control system integration. Therefore, this web-based enterprise module provides a simplified interfacing solution for the process segment and production performance monitoring and control in manufacturing enterprises, especially small-to-medium-sized manufacturing enterprises. It is easy to understand by people on different levels of a manufacturing enterprise. It is small in scale, but its module function adds flexibility, compatibility, and extendibility for future development.

In this study, a queuing network model is established to analyze the effect of system change. A statistical method, one-way ANOVA, is applied to evaluate the effect on the system. And SPSS statistical software is used for the calculations.

### Purpose of the Study

The purpose of this study is to establish a generalized partial enterprise module for enterprise-control system integration in manufacturing enterprises based on the development of a real-time data communication mechanism among ASP web forms, LabVIEW control applications, and a dynamic database, and evaluate the efficiency of

the module by a statistical study based on the queuing network modeling. It is a web-based enterprise module that provides different interfaces for end users in the manufacturing enterprise based on their different roles over the Internet. Virtual remote panels for process control and monitoring, real-time data retrieving, data analysis, and system maintenance are available from the web-based interfaces. System security is executed by implementing ASP form authentication and authorization. End users are required to provide user credentials to logon to the system, and their accesses are managed by a role-based authorization method. The dynamic database is built in the Microsoft SQL Server 2000 Desktop Engine (MSDE), a database server. Through the real-time data communication mechanism, the database is able to update data according to changes in LabVIEW-based control and monitoring segments in real time, and ASP web forms will display these changes to end users on the web-based interfaces.

This real-time data management solution on LabVIEW-based control and monitor processes provides a template to implement enterprise-control integration on LabVIEW-based control and monitoring segments in manufacturing enterprises. It integrates LabVIEW-based control and monitoring segments with enterprise-wide web applications and provides real-time data from manufacturing processes on factory floor for decision support and operation monitoring in higher enterprise levels. Therefore, a timely data communication between web-based interfaces, the dynamic database, and LabVIEW-based control and monitoring segments, is established to support real-time data access by end users.

The design of this enterprise module uses GERAM as a reference model, and applies terminologies and data attributes in the ISA S95 standard. This design method

standardizes terms, data structure and attributes in this enterprise module, thus it is easy for manufacturing people to understand and implement.

In order to evaluate the efficiency of the module, a single service model is established based on the queuing network modeling. The service represents the system resource (CPU and processor), and the customer represents the transactions processed in LabVIEW application. The existing system is the system running a LabVIEW control application without a data collector. The modified system is the system running a LabVIEW control application with a data collector integrated. A statistical technique is required to provide answers to the question – is there a significant difference on the CPU usage and the number of threads between the existing system and the modified system.

#### Significance of the Study

The management of complex value chains in manufacturing enterprises requires increased integration of disparate plant control systems and other computerized enterprise processes (Mick, 2003). Using Internet technology to gain more effective integration is a central unifying theme for the 21st century manufacturing enterprise (Worthington & Boyes, 2002). However, the state of enterprise integration becomes rather confusing. On the one hand, the need for enterprise integration solutions is intensified by the competitive environment and market expectations. On the other, the solutions seem to compete with one another, focus on particular issues, use conflicting terminology and do not provide any clues on their relations to solutions on other issues. This dilemma is even more obvious on the interfacing between manufacturing control functions and other enterprise functions for manufacturing enterprises (Williams, 1998).

SAP and Lighthammer are software vendors that provide solution packages in enterprise-control system integration for manufacturing enterprises. SAP created SAP NetWeaver and SAP .NET Connector to extend its strong business solutions into the manufacturing plant floor. Lighthammer uses Microsoft .NET as a platform for developing and deploying XML-based web services. Its Collaborative Manufacturing Service (CMS) brings web services technologies to manufacturing control integration. At the same time, automation manufacturers are trying to integrate web functions into their automation systems to reach the computerized business processes. For instance, SIMENS developed SIMATIC IT to help their customers reach the computerized business processes from SIMENS automation products. Rockwell provided RSLinx and RSView for web-based remote control and interface design based on Rockwell logix controller systems. The solution packages provided by these companies have common characteristics: (1) focusing on their own technology background; (2) being closely related to their former products; and (3) creating high initial investment. For small-to-medium-sized manufacturing enterprises, it is hard to get business justification for implementing these solutions. As indicated by MEL, there exists the need for modular, more flexible, more compatible and economic solutions to implement enterprise-control integration in manufacturing enterprises.

Today many issues exist in enterprise integration. The fragmentation of current research activities leads to multiple sub-solutions with many overlaps and even more contradictions. This prevents potential users from employing the research results on a sufficient scale in their day-to-day operation and in turn reduces the interest of IT vendors to invest in the necessary support technology for enterprise integration (MEL, 1999). For

example, some solutions are discussed for dealing with particular communication problems in control-system interfacing (Schneider, 2000). Some studies discussed the details of the different types of Ethernet standards and future development. Some studies discussed the advantages and disadvantages of using Ethernet for manufacturing applications, the various hardware and software available, and the role of web in process control applications (Wojcik, 2000). Most studies concentrated on one specific application, such as the development of a web-based HMI (Taccolini, 2001), and the application for a remote monitoring (Woods, 2003). A recent study on developing a multi-agent system for enterprise integration provided a solution called a multi-agent system to integrate different legacy business applications, like Capacity Analysis (CA) and Enterprise Resource Planning (ERP) (Peng, Finin, Labrou, Chu, Long, Tolone, and Boughannam, 2002). As shown above, most studies in this field either provide specific technical solutions to improve the performance of an integration application, or develop a new application for a specific situation. However, none of these studies resulted in a generalized partial enterprise module for implementation of enterprise-control system integration.

Data access technology is the core technology that determines the efficiency and feasibility of applications for enterprise-control system integration. The hard part of managing real-time data has always been trying to make a computerized database to communicate with plant-floor controllers in a timely manner. Since the late 1980s, database server vendors have published quite a few versions of database servers and provided different database access technologies for their servers, like Microsoft and Oracle (Foggon & Maharry, 2004). From the Microsoft DataBase (MDB), the simplest

relational database, to MySQL, the best open source relational database server, Microsoft has published several versions of database servers. Data access technologies also have evolved from Open DataBase Connectivity (ODBC) to the latest ADO.NET. Database server vendors like Microsoft have been trying to make database access as easy and as painless as possible for anyone who needs that facility. But it seems that these technologies have been heavily applied in page designs and e-commerce related applications. It is difficult for factory-floor controllers to access databases directly. While LabVIEW provides virtual interfaces for those controllers, there is no direct application for a database to communicate with LabVIEW interfaces for real-time data retrieving. The development of a dynamic database that is capable of communicating with LabVIEW-based control and monitoring segments on the factory floor is the requirement of implementing enterprise-control system integration.

The web-based partial enterprise module established in this study is an integrated web service platform established in the .NET and LabVIEW environment. It is a service-based architecture to provide interactive web-based interfaces for people on different levels of a manufacturing enterprise who are searching for real-time information from the factory floor. The module functions as a bridge between factory-floor controllers and higher level enterprise functions for real-time data communication. It can be implemented in real world industry as a reference integrating infrastructure for enterprise-control system integration, particularly applicable for small-to-medium-sized manufacturing enterprises. It also can work as an initialized web-based platform for system integration simulations in academic laboratories. It is a modular system, and is extensible to accommodate various web-based controllers and instrumentations.

## Definition of Terms

*ADO*: Microsoft ® ActiveX ® Data Object is used to enable client applications to access and manipulate data from a variety of sources through an OLE DB provider. ADO supports key features for building client/server and Web-based applications.

*ASP*: Active Server Page is a Microsoft web technology for creating dynamic Web applications. It enables HTML pages to be dynamic and interactive by embedding scripts. This technology will be used to build the performance management system in this study.

*ActiveX*: A set of technologies that enables software components to interact with one another in a networked environment, regardless of the language in which the components were created. Currently, ActiveX is used primarily to develop interactive content for the World Wide Web. ActiveX controls can be embedded in Web pages to produce animation and other multimedia effects, interactive objects, and sophisticated applications. In this study, ActiveX will be utilized to interact between LabVIEW VIs and web applications.

*Enterprise Integration*: ISA S95 defines enterprise integration as the coordination of the operation of all elements of the enterprise working together in order to achieve the optimal fulfillment of the mission of that enterprise as defined by enterprise management (Williams, 1998). A more understandable definition by MEL is providing the right information, at the right place, at the right time, and updating the information in real time to reflect the actual state of the enterprise operation (MEL, 1999).

*Enterprise-control System Integration*: A part of the whole enterprise integration, which defines the interface between plant control systems and other enterprise systems, such as business process systems (Williams, 1998).



*GERAM*: Generalized Enterprise Reference Architecture and Methodology is about those methods, models and tools which are needed to build and maintain the integrated enterprise (MEL, 1999).

*ISA S95*: The international standard for the integration of enterprise and control systems. S95 consists of models and terminology which can be used to determine which information has to be exchanged between systems for sales, finance and logistics and systems for production, maintenance and quality.

*LabVIEW*: The graphical development environment for creating flexible and scalable test, measurement, and control applications rapidly. It will be used to develop VIs that interface with real-world signals, analyze data for meaningful information, and share results and applications in this study.

*.NET*: This term denotes the Microsoft platform that can be used to develop applications to connect information, people, systems, and devices based on web service architectures. Microsoft .NET technology will be used to develop web forms and XML web services for the performance management system in this study.

*Relational Database*: The concept of relational databases was first described by Edgar Frank Codd (almost exclusively referenced as E. F. Codd in technical literature) in the IBM research report RJ599, dated August 19th, 1969. Basically, a relational database consists of a set of tables, where each table is a set of records. A record in turn is a set of fields and each field is a pair field-name/field-value. All records in a particular table have the same number of fields with the same field-names. All information in the database should be represented in one and only one way -- as values in a table. Each and every

datum (atomic value) is guaranteed to be logically accessible by resorting to a combination of table name, primary key value, and column name.

*VIs:* Virtual instruments. In this study, VIs represent the graphical user interfaces programmed in LabVIEW environment for the purpose of motion control and process control.

*Web Forms:* A Web Forms page presents information to the user in any browser or client device and implements application logic using server-side code. Web forms are designed by using Microsoft ASP.NET technology in which code that runs on the server dynamically generates Web page output to the browser or client device. In this study, web forms are the main elements for constructing the performance management system.

*XML Web Service:* Extensible Markup Language (XML) is a simple, very flexible text format code. It is applied in the exchange of a wide variety of data on the Web and elsewhere. An XML Web service is a programmable entity that provides a particular element of functionality, such as application logic, and is accessible to any number of potentially disparate systems using ubiquitous Internet standards, such as XML and HTTP. An XML Web service can be used internally by a single application or exposed externally over the Internet for use by any number of applications. In this study, XML web service is the key element for realizing real-time performance management.

## Chapter 2

### REVIEW OF LITERATURE

#### Historical Background

Enterprise integration is the re-engineering of business processes and information systems to improve teamwork and co-organizational boundaries, thereby increasing the effectiveness of the whole enterprise (Francois, 1996). It has been discussed since the early days of industrial computers in industry in general and in the manufacturing industry in particular. The integration in manufacturing industry has been explained as the operation integration and support of communication in manufacturing by means of information technology. Until today, there are four phases in the evolution of the integration applications in manufacturing systems: (1) serial device networks on the plant floor; (2) distributed device networks including serial and Ethernet networking; (3) integration of production management system with automation system; and (4) the current challenge of real-time performance management (Mick, 2003).

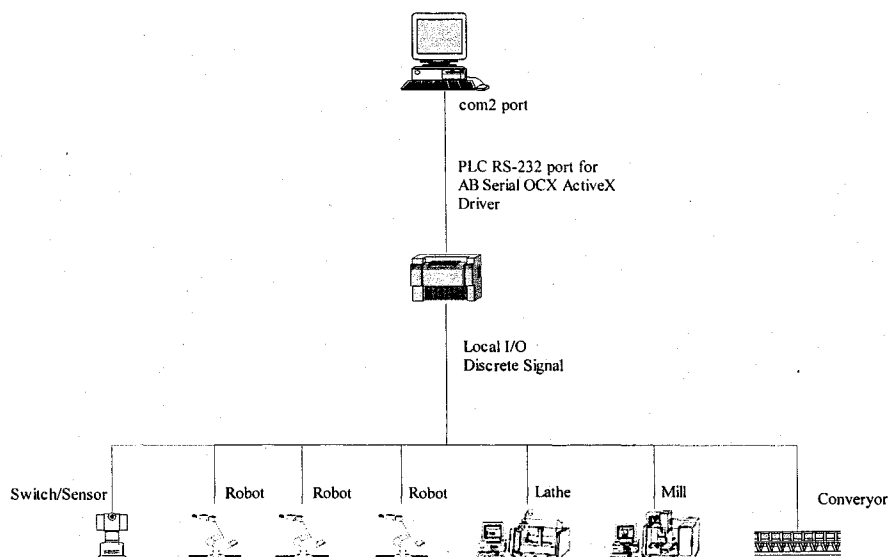
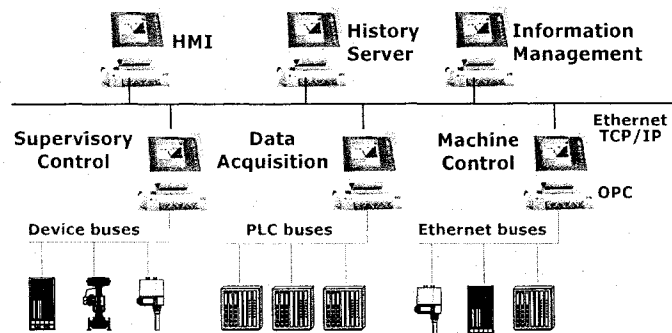


Figure 1. Serial device network

As shown in Figure 1, computers were initially used in plant floor applications as production device configuration tools and for providing diverse operator interfaces. The system was integrated commonly through serial connection (RS232 or RS485) and was quite suitable for PLC (Programmable Logical Controller) programming, device configuration, and some operator interface applications. The serial device networking system had limited capabilities. However, the software running on this system for trending, archiving, sequencing, and PC-based control, demonstrated the demand for built-in networking. This phase is the integration of plant-floor controllers with HMI.

Distributed systems, the second phase for integration of manufacturing systems, are the integration of plant-floor device control with networked computers for plant management. Distributed systems are extended from the principle of Distributed I/O. The concept of Distributed I/O is an industry standard in the data acquisition and control industries. Distributed I/O is essentially the placement of intelligent devices that have the

ability to make decisions based upon environmental conditions. By using robust networking technology to position intelligent, distributed I/O devices closer to the sensor, process, or unit under test, significant cost savings and improved performance can be obtained. Distributed I/O systems include special capabilities to improve reliability, onboard diagnostics, and maintenance to maximize system uptime. Intelligent distributed systems can help to implement embedded, deterministic control systems. Some practical applications of distributed I/O are assembly line control, remote control of construction equipment, military excursions, hazardous material situations, factory monitoring and control, and Heating Ventilation and Air Conditioning (HVAC) systems.



#### Distributed Systems

Figure 2. Distributed device networks

(Courtesy of Microsoft Corporation)

Distributed systems were enabled by the advancement of control devices and industrial device level busses which evolved from serial media to Ethernet control networks. The distributed systems managed all aspects of automation systems and field devices, including programming, configuration, and operation. They collected and

managed growing amounts of historical data in support of production operators, supervisors, and engineers. The typical systems are Windows NT-based system (Microsoft) and NI FieldPoint system (National Instrument). It becomes clear that standard software interfaces to devices and information are required. Advanced control functions such as on-line optimization, statistical quality control, and PC-based control need to be economically feasible.

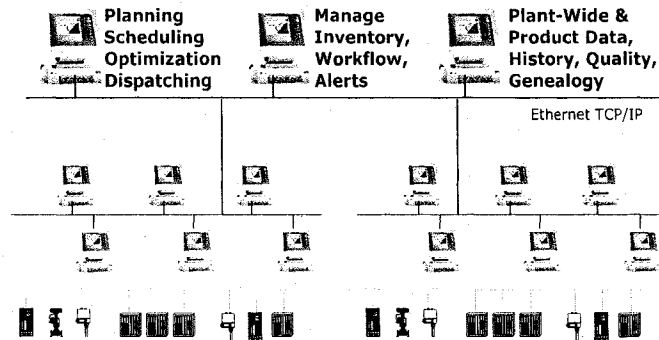


Figure 3. Integration of production management system  
(Courtesy of Microsoft Corporation)

The integrated systems in Figure 3 went beyond the automation systems to include production management systems, including such capabilities as detailed planning and scheduling, batch management, plant-wide historical data management, and quality data management. This third phase of integration in manufacturing systems built hierarchical LAN (Local Area Network) or WAN (Wide Area Network) networks for plant-wide information management. The integration of production management systems requires

considerably more IT infrastructure than just automation systems. Typically, Microsoft's relational database, SQL Server, is required for data management needs, MSMQ for messaging, BizTalk for process-centric integration, and COM+.

The current challenge for the integration in manufacturing systems is to build integrated systems that are able to detect and react to events in real time and provide near immediate feedback on how the enterprise is doing against dynamic targets (Mick, 2003). This accentuates the demand on information systems to reliably bring together information from production and business systems, and place it in a context where people at any level can make better decisions faster. Dynamic, standards-based architectures are required for realizing the potential of such a Real-time Performance Management (RPM). Faster integration and flexibility are being pursued for enterprise-wide service-based architectures.

Web-based applications, which have been used in business systems, are being applied in manufacturing systems to achieve the manufacturing system integration over the Internet. Web forms, web services, and Extensible Markup Language (XML), are critical tools for implementing web-based system integration. Because of the data variety in manufacturing systems, database and data access technologies are especially important for the efficiency and feasibility of manufacturing systems integration. Opportunities for manufacturing systems are to implement re-useable services and rapid application development tools to build agility and track enterprise objectives closely; to use service based process engines to implement processes that cross business and manufacturing functions; and to develop manufacturing metrics and intelligence using standard Web forms and services that consistently access historical information and status across

diverse systems. Also manufacturing systems can store and manage persistent data for product models, processes and configurations in XML, minimizing the loss of intellectual property, and use service registries to manage and define interfaces and deploy applications with the most appropriate communication methods.

The future challenge will be to standardize the implementation to achieve interoperability and easy integration. Security problems and requirements will be met, opening the door for higher levels of real-time visibility and more dynamic enterprise wide platforms.

### Architecture Models for Enterprise Integration

Enterprise integration attempts to create a complete information processing system to serve both functional requirements and the corporate business objectives of enterprises in real time. In order to enable consistent modeling of the enterprise integration, the modeling process has to be guided and supported by reference architecture, a methodology and IT based tools. Previous researches have produced several reference architectures to organize all enterprise integration knowledge and serve as a guide in enterprise integration applications, for example, CIMOSA by the AMICE Consortium, PERA by Purdue University, TOVE by the University of Toronto, GRAI and GIM by the GRAI Laboratory, and GERAM by IFAC/IFIP Task Force. These reference architectures proposed conceptual frameworks and associated methodologies to support the life-cycle states of an integrated manufacturing enterprise.

CIMOSA (Computer Integrated Manufacturing Open System Architecture) was developed for ESPRIT (European Strategic Program for Research and Development in Information Technology) by AMICE (a consortium of 30 major European vendors and



users of CIM systems. It is aimed at the development of open reference architecture for the definition, specification and implementation of CIM systems. CIMOSA defines a model-based enterprise engineering method which categorizes manufacturing operations into Generic and Specific (Partial and Particular) functions. The CIMOSA framework has 3 levels of architectural generality: the requirements modeling definition level, the design specification modeling level, and the implementation description modeling level. These levels contain all the constructs required to gather the user requirements for this system operation and to translate these requirements into a consistent system description and implementation. CIMOSA provides four different types of views: function view, information view, resource view and organization view (Francois, 1996). The Function View describes work flows. The Information View describes the Inputs and Outputs of Functions. The Resource View describes the structure of resources (humans, machines, and control and information systems). The Organization View defines authorities and responsibilities.

PERA (Purdue Enterprise Reference Architecture) developed at Purdue University focuses on separating human based functions in an enterprise from those with a manufacturing or information perspective (Weston, 1998). PERA takes two views of the enterprise, a functional view and an implementation view. The functional view consists of an information functional model and a manufacturing functional model. The implementation view consists of the information architecture and the manufacturing architecture (Francois, 1996). Both information and manufacturing streams flow throughout the two views. The information stream consists of planning, scheduling,

control, and data management functions whereas the manufacturing stream consists of physical production functions.

TOVE (Toronto Virtual Enterprise), developed at the University of Toronto, created a data model that provides a shared terminology for an enterprise that different computer systems can jointly understand and use. It defined a generic level representation that includes the representations of time, causality, activity and constraints. This generic level is defined in terms of a conceptual level based on a certain terminology (Tham, 1997).

The GRAI-GIM methodology was begun at the University of Bordeaux in the 1970's. It was designed to help define a model of an Integrated Manufacturing System in order to specify a CIM System for subsequent purchase or development. The GRAI model is a reference through which various elements of real world can be identified. The macro conceptual model is used to express one's perception and ideas on the manufacturing system which is decomposed into a decision subsystem, an information subsystem and a physical subsystem. Particularly within the decision subsystem one finds a hierarchical decision structure composed of decision centers. Decision centers are connected by a decision frame (objectives, variables, constraints and criteria for decision making). The operating system is an interface between the decision system and the physical system. The micro conceptual model is used to represent the internal elements and structure of the decision center (Williams & Li, 1998).

The IFIP/IFAC Task Force analyzed all the existing architectures and concluded that even if there were some overlaps, none of the reference architectures subsumed the others; each of them had something unique to offer. Starting from the evaluation of existing enterprise integration architectures, the IFAC/IFIP Task Force has developed an overall

definition of a generalized architecture, GERAM (Generalized Enterprise Reference Architecture and Methodology) (MEL, 1999). GERAM provides a description of all the elements recommended in enterprise integration and sets the standard for the collection of tools and methods. GERAM views enterprise models as an essential component of enterprise integration. The set of components identified in GERAM include generalized enterprise reference architecture, enterprise engineering methodology, enterprise modeling languages, partial enterprise models, generic enterprise modeling, enterprise engineering tools, enterprise modules, and enterprise operational systems (MEL, 1999).

In conclusion, GERAM provides the necessary guidance of the modeling process, and enables semantic unification of the model contents. To develop a generalized partial enterprise module in this study, GERAM is selected to be the reference architecture for modeling.

#### ISA S95 International Standards for Integration

This study establishes a partial enterprise module to interface between manufacturing control functions and other computerized enterprise functions for enterprise-control system integration. This enterprise module is a very important example of the multi-use of the same meaning in different words and has the vital need for cross-understanding and mutually acceptable standards. ISA S95 and its XML implementation B2MML provide standardized terminology and tools for data and attribute definitions for the interface constructions in this study.

ISA S95 is the international standard for the integration of enterprise and control systems developed by ISA and the World Batch Forum. It was created to solve the rising problem of misunderstanding grown between the industrial control and business process

groups, and to implement the necessary ready transfer of information in electronic form between business processes and the plant floor control systems. ISA S95 establishes common terminology for the description and understanding of enterprise, including manufacturing control functions and business process functions (Williams, 1998). It is a standard that defines the interface between manufacturing control functions and other enterprise functions, and defines information exchange between them including data models and attribute definitions.

ISA S95 consists of models and terminology which can be used to determine which information has to be exchanged between systems for sales, finance and logistics and systems for production, maintenance and quality. There are three parts of the ISA S95 standard: models and terminology; object model attributes; and activity models of manufacturing operation management. Part one consists of standard terminology and object models, which can be used to determine which information must be exchanged between plant control systems and enterprise business processes. Part two consists of attributes for every object that is defined in part one. The objects and attributes of part two can be used for the exchange of information among different systems. Part three focuses on the functions and activities of manufacturing operations management which is still under developing by ISA and World Batch Forum group.

B2MML (Business to Manufacturing Markup Language) is an XML implementation of the ANSI/ISA 95 family of standards (ISA-95) developed by World Batch Forum, known internationally as IEC/ISO 62264. B2MML consists of a set of XML schemas written using the World Wide Web Consortium's XML Schema language (XSD) that implement the data models in the ISA-95 standard. B2MML has totally defined nine

schemas that can be used respectively for information exchange regarding manufacturing equipment, maintenance, material, personnel, process segments, product definition, production capability, production performance, and production schedule (WBF, 2003). Each XML schema defines type names, user elements, user enumeration and models for information exchange.

In this study, B2MML schema will be used as an XML tool to define data and data attributes in constructing the web-based interface for the real-time performance management system.

### .NET Technologies for Module Development

All infrastructure and platform suppliers are building XML and Web services into their products, and it is clear that most business and plant floor systems will eventually be impacted. Even now, manufacturers are deploying XML widely, and Web services in limited areas (Mick, 2003). Web services will be used more heavily over the next few years, and a Service Based Architecture (SBA) is necessary as a reference framework for integration between business and production systems. As mentioned in the last section, ISA S95 has defined standard models and terminology for business and production systems to communicate. These have been defined with XML schemas by the World Batch Forum, and are being used by software suppliers with Web services.

To build a generalized partial enterprise module for integration between manufacturing control functions and other computerized enterprise functions in this study, ASP Web applications and XML Web services are the main components to construct dynamic web applications and achieve web-based information exchange between those functions. .NET technologies provide a complete set of development tools for building

ASP Web applications and XML Web services. Therefore, .NET component platforms will be used as the construction platforms for the partial enterprise module in this study. .NET is a vision and set of Microsoft software technologies for connecting information, systems, and devices. It enables a high level of software integration through the use of XML Web services—small, discrete, building-block applications that connect to one another as well as to other, larger applications over the Internet.

ASP Web applications will be compiled in ASP.NET environment, which is a part of the .NET framework. ASP.NET includes a set of controls that encapsulate common HTML user interface elements. These controls run on the Web server, however, and push their user interface as HTML to the browser. On the server, the controls expose an object-oriented programming model. It allows programmable web pages to be integrated with ASP Web applications. Programmable web pages are the important elements that realize the real-time performance management in this study.

XML Web services are applications that can receive requests and data using XML over HTTP. XML Web services are not tied to a particular component technology or object-calling convention and can therefore be accessed by any language, component model, or operating system. Also XML Web services can provide the means to access server functionality remotely by clients. These features make XML Web services perfect tools for constructing the real-time performance management system.

#### LabVIEW for the Development of Control Systems

LabVIEW, developed by National Instruments, is a graphic programming language to build virtual instruments (VIs) for control systems. The VI developed in LabVIEW environment provides an interface between a user and a control process. The main

concept of such an interface is to provide a general view of the process and facilitate full control of the operations (Beyon, 2001).

Three reasons exist for this study to choose LabVIEW-based control systems to implement enterprise-control system integration. First, LabVIEW is widely used in developing automatic control solutions in real world industries, research studies and academic laboratories. Second, in LabVIEW, the locally controlled setup can be turned into a remotely controlled one by moving the user interface away from the physical setup with the integrated web-based functions. These web-based functions not only represent the advanced technology in current plant control systems, but also provide features that will benefit the construction of the web-based enterprise integration module. Third, LabVIEW also provides advanced communication methods for the integration of LabVIEW VIs with other applications, such as ActiveX containers, File Input and Output, and .NET constructor nodes.

In this study, the method used for control system interface in LabVIEW environment is FieldPoint. FieldPoint is a proprietary method for interfacing devices to computers developed by National Instruments. But it is very similar in principle to the standard of a fieldbus interfacing method used by many process control equipment suppliers. The idea of fieldbus grew out of the problem of interfacing hundreds or thousands of sensors and actuators to PLCs and process control computers in large industrial plants. Rather than connect each sensor or actuator to a central plant computer, requiring hundreds or thousands of kilometers of wiring, the idea of fieldbus is to connect related groups of sensors and actuators to a local microcomputer that communicates with the central plant computer via an Ethernet local area network (LAN). Earlier fieldbus units used serial

interface lines for communication but the principal remained the same. The result was an enormous reduction in wiring and a corresponding increase in reliability.

For FieldPoint control in the LabVIEW environment, a virtual interface programmed by LabVIEW graphical language provides a control panel for users to interact with the control process through FieldPoint Ethernet communication and the communication between the FP controller and I/O Modules. The communication between the FP controller and I/O Modules is similar among different types of network modules. Each I/O module cycles through its internal routine of sampling all channels, digitizing the values and updating the values on the module channel registers (buffer).

FieldPoint Ethernet communication uses an asynchronous communication architecture called event-driven communication. The network module automatically sends updates to a client when data changes. The server then caches the data from I/O modules and uses it to respond to read requests from the virtual interface. The network module scans all I/O channels with subscriptions to determine whether a value has changed, comparing the current value to the cached value for each channel. If a change has occurred, the network module puts the difference between the two values in the transmit queue. The FP Server receives this information and sends an acknowledgement to the network module. The network module periodically sends and receives a time-synchronization signal so that it can adjust its clock and provide proper timestamping. When signals do not change over long periods of time, the client sends periodic re-subscribe messages to verify that the system is still online.



## Relational Database and Data Access

Relational databases and data access technologies are the core components in establishing the enterprise module for enterprise-control system integration in this study. A relational database stores all its data inside tables, and nothing more. And a relational database management system (DBMS) must manage its stored data using only its relational capabilities. All operations on data are done on the tables themselves or produce some other tables as the result. A table is a set of rows and columns. Each row is a set of columns with only one value for each. All rows from the same table have the same set of columns, although some columns may have NULL values, i.e. the values for the rows were not initialized.

Edgar Frank Codd has developed the famous "Twelve Rules for Relational Databases", which were published in two Computerworld articles "Is Your DBMS Really Relational?" and "Does Your DBMS Run By the Rules?" on October 14, 1985, and October 21, 1985, respectively. These twelve rules are still considered gospel for relational database implementations (Bostrup, 2005).

1. Information Rule: All information in the database should be represented in one and only one way -- as values in a table.
2. Guaranteed Access Rule: Each and every datum (atomic value) is guaranteed to be logically accessible by resorting to a combination of table name, primary key value, and column name.
3. Systematic Treatment of Null Values: Null values (distinct from empty character string or a string of blank characters and distinct from zero or any other number)

are supported in the fully relational DBMS for representing missing information in a systematic way, independent of data type.

4. Dynamic Online Catalog Based on the Relational Model: The database description is represented at the logical level in the same way as ordinary data, so authorized users can apply the same relational language to its interrogation as they apply to regular data.
5. Comprehensive Data Sublanguage Rule: A relational system may support several languages and various modes of terminal use. However, there must be at least one language whose statements are expressible, per some well-defined syntax.
6. View Updating Rule: All views that are theoretically updateable are also updateable by the system.
7. High-Level Insert, Update, and Delete: The capability of handling a base relation or a derived relation as a single operand applies not only to the retrieval of data, but also to the insertion, update, and deletion of data.
8. Physical Data Independence: Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representation or access methods.
9. Logical Data Independence: Application programs and terminal activities remain logically unimpaired when information preserving changes of any kind that theoretically permit unimpairment are made to the base tables.
10. Integrity Independence: Integrity constraints specific to a particular relational database must be definable in the relational data sublanguage and storable in the catalog, not in the application programs.

11. Distribution Independence: The data manipulation sublanguage of a relational DBMS must enable application programs and terminal activities to remain logically unimpaired whether and whenever data are physically centralized or distributed.
12. Nonsubversion Rule: If a relational system has or supports a low-level (single-record-at-a-time) language, that low-level language cannot be used to subvert or bypass the integrity rules or constraints expressed in the higher-level (multiple-records-at-a-time) relational language.

Databases need to be hosted by database servers. A database server is a data storage and retrieval system, and it is the back end part of a client-server database (Foggon & Maharry, 2004). A database server controls the storage of the data, grants access to users, updates and deletes records, and communicates with other servers when necessary. There are several database servers available for database construction, such as Microsoft DataBase (MDB), MySQL, SQL Server, Oracle, and so on. In this study, the dynamic database is constructed in the Microsoft SQL Server 2000 Desktop Engine (MSDE). MSDE is the free version of Microsoft's full SQL Server database server. Its main difference is that the number of clients that can access it at the same time is limited to 25. Since the system established in this study is for research purposes, MSDE is selected instead of another database server. Once it needs to be applied in the real industrial world, the database can be migrated to a more powerful version database server.

Data access technologies are the methods used to support data accesses to databases. In the more than ten year's history of data access technologies, the most important technologies are Open DataBase Connectivity (ODBC), Object Linking and Embedding

for DataBases (OLE DB), ActiveX Data Object (ADO), and ADO.NET. ODBC was developed at the beginning of the 1990s. It is a common set of functions and interfaces agreed upon by all the major database vendors at that time to be implemented by all their servers. ODBC was extremely successful and is still supported by all the major database servers in use today. However, ODBC works at quite a low level, it is difficult to use. OLE DB is a set of Component Object Model components designed for Windows application developers that makes accessing data a bit simpler. OLE DB was pretty successful and is still supported by several vendors, including Microsoft. ADO is a technology originally designed to give classic ASP pages a way to access databases. ADO.NET now takes over from ADO. ADO.NET can work with a database through a common set of methods and interfaces regardless of whether it supports ODBC, OLE DB, or its own proprietary access solution.

ADO.NET is used in this study to establish the communication method among the ASP forms, the dynamic database, and LabVIEW-based control systems. As the latest version of data access technology, ADO.NET is able to work with data away from database itself pulling information onto the web server and working with it there instead of on the database server. It is able to take a large number of simultaneous queries and keep the stability and performance at the same time. ADO.NET can bind information to any control on an ASP page.

## Chapter 3

### METHODOLOGY

#### Restatement of the Study Objectives

This study is to establish a generalized partial enterprise module for enterprise-control system integration in manufacturing enterprises based on the development of a real-time data communication mechanism among ASP web forms, LabVIEW control applications, and a dynamic database. It is a web-based enterprise module that provides interface for end users in the manufacturing enterprise over the Internet. Virtual remote panels for process control and monitoring, real-time data retrieving, data analysis, and system maintenance are available from the web-based interface. System security is executed by implementing ASP form authentication and authorization.

This partial enterprise module is a modular system that can be used to implement enterprise integration solutions in LabVIEW-based manufacturing control systems. The system's architecture is designed according to GERAM architecture and methodology. The information transaction methods, objects in the architecture, and attributes of each object are defined by applying ISA S95 standard in enterprise control system integration. The modular system consists of three components, a LabVIEW module running in LabVIEW applications, a database server, and a web interface.

## Overview of the Integration Module

### *System Architecture*

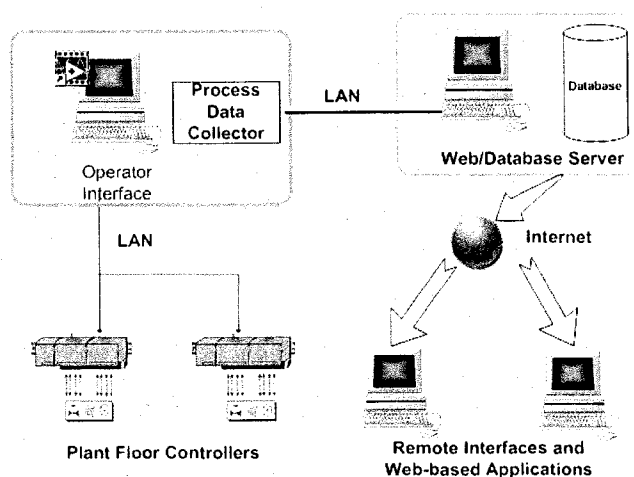


Figure 4. System architecture

Figure 4 displays the architecture overview of the web-based enterprise integration module. The modular system consists of three main components: a LabVIEW-based process data collector, a virtual server, and a web-based interface. The LabVIEW-based process data collector is a set of subVIs developed in LabVIEW environment to collect real-time process data from LabVIEW control applications and send the collected data to specified database tables. It was developed by using LabVIEW controls and Microsoft® ActiveX® Data Objects (ADO) so that it can access and manipulate data in an OLE DB database from LabVIEW control applications. This data collector has features of ease of use, high speed, low memory overhead, and reusability. It can be utilized by various LabVIEW applications for data collection activities with minor modifications. The

frequency of data updating is determined by the timer configuration of the While Loop in LabVIEW applications which is a configurable parameter.

The web server and the database server could be set up as two computer servers located in the same LAN network, or two virtual servers established and configured in one computer. In this study, the servers' capacity for client support is not the critical concern. Therefore, two virtual servers are established and hosted by one computer. One is the database server which hosts an MSDE database and provides a management interface for direct database control. The database server supports the communication channels for the data collector module running in the LabVIEW control applications. The database exchanges production information with manufacturing control systems in near real time through the communication channels. One virtual web server hosts ASP web pages and supports remote accesses to the interface. The web server compiles dynamic web pages according to different data requests received from clients, displays web pages to clients' browser, and reacts to clients' actions on the web pages.

The web-based interface consists of a series of ASP dynamic web pages. It is supported by the MSDE database server and the LabVIEW-based data collector module. Therefore, the web-based interface is not only a normal web site that can be accessed by authenticated users over the Internet, but also a remote real-time system control panel and data analyzer that provides real-time process data and historical data analysis for the decision-making process in the manufacturing enterprise. This interface is the end-user component of the web-based integration module. It needs to be customized and re-configured for specific usage of the integration module.

### Interface Features

The web-based interface in this integration module is developed in .NET environment. Due to the time limit and resources available in this study, the web-based interface is built only to provide the critical features that are necessary for system performance testing. In the module's future applications, the interface needs to be customized and enhanced to provide more functions and features. However, the most important features, which are the main purpose of this study, are integrated with this web-based interface. The main features are the real-time data retrieve from the database server, the data analysis, real-time direct control and monitor of the plant-floor process, and a live video of the process. The main window of the interface is shown in Figure 5.

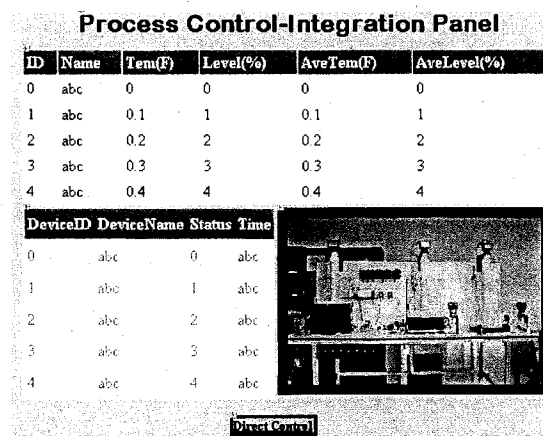


Figure 5. The main interface window

The main window of the web-based interface provides data tables to display the real-time process data from the plant-floor process which can be updated every 250 milliseconds. This window is developed to provide real-time data in a timely manner for



decision makers in a manufacturing enterprise. The simplicity of the interface will reduce the page load and access time and save the processing cycle and time of the server. As shown in Figure 5, the interface is in offline state with no data retrieved from the database server. Figure 6 and 7 provide the data table in online status with data filled from the database server.

ID	Name	Tem(F)	Level(%)	AveTem(F)	AveLevel(%)
0	TANK1	40	50	44	30
1	TANK2	42	20	41	22
2	TANK3	41	30	41	

Figure 6. Measurement data table

The data table in Figure 6 displays the temperature value and the level value of the liquid product in each tank of the process. The average temperature and level value of a specified time period are also displayed. The data table in Figure 7 displays the On/Off status of each device in the process.

DeviceID	DeviceName	Status
0	valve1	0
1	valve2	1
2	pump1	0
3	heater	1
4	pump2	1
5	valve3	1
6	valve4	1

Figure 7. Device status data table

The main window provides a hyperlink button which navigates to the real-time LabVIEW control panel of the plant-floor process as shown in Figure 8.

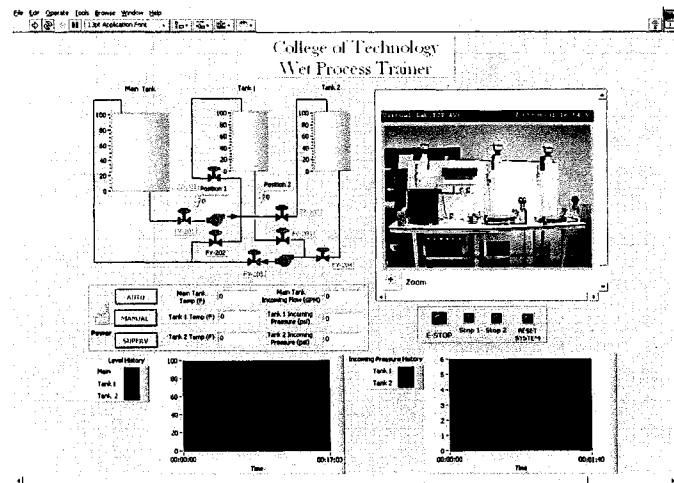


Figure 8. LabVIEW real-time control panel

This control panel is a virtual instrument programmed in LabVIEW environment. As shown above, controls and indicators on the interface provide the way for users to interact with the control process. A video clip was integrated into the interface for users to monitor the real process through an Internet camera. Two waveform graphics provide history data tracking of temperature and incoming pressure of each tank. The current value of tank levels, temperatures, incoming flow rate for main tank, and incoming pressure for tank 1 and tank 2 are also shown on the interface by different indicators. More details about this LabVIEW remote control panel will be discussed in the next section of the chapter. The possible enhancements and further integrations of this web-based interface for future applications will be discussed in detail in Chapter Four.

## Construction of the Data Collector

### *LabVIEW Environment and Database*

As introduced in the last chapter, LabVIEW is a graphic programming language to build virtual instruments (VIs) for control systems. The VI developed in LabVIEW environment provides an interface between a user and a control process. A database consists of an organized collection of data. Most modern Database Management Systems (DBMS) store data in tables. The tables are organized into records, also known as rows, and fields, also known as columns. LabVIEW programming language and database applications are different language-based software, and can not communicate with each other at the same level.

In order to access a database hosted by a Windows-based database server, LabVIEW applications have to utilize third-party software. National Instrument has released a software kit, called the Database Connectivity Toolset for LabVIEW. The LabVIEW Database Connectivity Toolset provides a method to communicate and pass data between LabVIEW and either a local or a remote database management system (DBMS). This third-party software kit costs around \$1000 and occupies processing capacity and time for communication. In this study, the LabVIEW data collector is a module integrated in virtual instruments providing a direct communication channel for LabVIEW applications and Windows-based database servers. The data collector is built in Microsoft .NET and integrated into the LabVIEW environment. These objects can make system calls into Microsoft's application programming interface (API) for database access called ODBC and use ADO method as the application interface for data communication. The data

collector can be utilized as an open source module to be integrated into various LabVIEW applications without cost.

### *Database Concepts*

A database consists of an organized collection of data. Every table in a database must have a unique name. Similarly, every field within a table must have a unique name. The database tables have many uses. Tables could be used with a simple test executive program to record sequence results. The data in a table could not be inherently ordered. Ordering, grouping, and other manipulations of the data occur when a SELECT statement is used to retrieve the data from the table. A row can have empty columns, which means that the row contains NULL values. NULL values for databases are not exactly the same as NULL values in the C programming language.

Non-relational databases are used to store all the information in one large structure. This method is sometimes inefficient, because all of the information is in one table, and searching for a specific piece of data can be difficult and time-consuming. Relational databases have information stored in multiple structures, or tables, where each table can be smaller and contain a specific subset of information.

### *Components of the Data Collector*

#### *ODBC standard.*

The SQL Access Group, including representatives of Microsoft, Tandem, Oracle, Informix, and Digital Equipment Corporations, developed the Open Database Connectivity (ODBC) standard as a uniform method for applications to access databases. ODBC 1.0 was released in September 1992. The standard consists of a multilevel API definition, a driver packaging standard, an SQL implementation based on ANSI SQL, and

a means for defining and maintaining Data Source Names (DSN). A DSN is a quick way to refer to a specific database. A DSN is specified with a unique name and by the ODBC driver that communicates with the physical database, local or remote. A DSN must be defined for each database to which an application program connects.

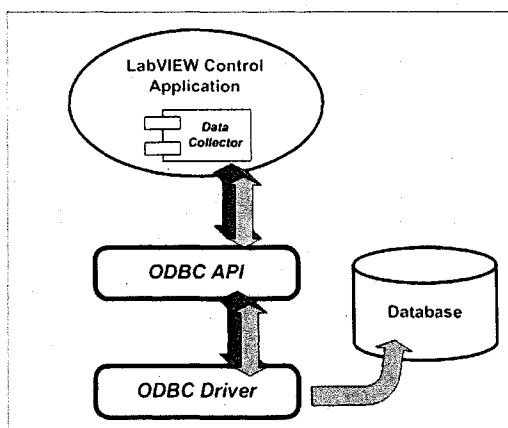


Figure 9. Communication path between LabVIEW and a database

The data collector complies with the ODBC standard, so that it can communicate with all ODBC-supported database applications. The data collector module in LabVIEW environment calls the Microsoft API for ODBC. ODBC then communicates with a database's specific driver that translates the call to the database's low level language, as shown in Figure 9. The data collector module is compatible with any database providing an ODBC driver that translates the ODBC calls to the native database language. ODBC API and drivers are integrated with all ODBC supported database servers, so that no extra software or application package is required to realize the communication.

### *Structured Query Language*

The data collector adopts the Structured Query Language (SQL) as command statements for data manipulation in data access. The Structured Query Language consists of a set of character string commands and is a widely supported standard for database access. The SQL commands can be used to describe, store, retrieve, and manipulate the rows and columns in database tables. IBM developed the language, and it became publicly available in the late-1970s. Since then, the American National Standards Institute (ANSI), the International Standards Organization (ISO), and the Federal Information Processing Standards (FIPS) have adopted SQL and most major commercial relational database products support it to some degree. It is a non-procedural language for processing sets of records in database tables.

There are three pertinent classes in SQL statements, Data definition/control language, Data manipulation language, and Queries. Data Definition/Control Language (DDL/DCL) statements define and control the structure of the database. They also define and grant access privileges to database users. Use the statements to create, define, and alter databases and tables. Data Manipulation Language (DML) statements operate on the data contents of database tables. These statements are used to insert rows of data into a table, update rows of data in a table, delete rows from a table, and conduct database transactions. Queries are SQL SELECT statements that specify which tables and rows are retrieved from the database. Table 1 describes the SQL commands that can be recognized and used by the data collector.

Table 1.

## SQL Commands Used by the Data Collector

SQL Command	Function
CREATE TABLE	Creates a database table and specifies the name and data type for each column therein. The result is a named table in the database. It is a DDL command.
INSERT	Adds a new data row to the table, allowing values to be specified for each column. INSERT is a DML command.
SELECT	Initiates a search for all rows in a table whose column data satisfy specified combinations of conditions. The result is an active set of rows that satisfy the search conditions. SELECT is a query command.
UPDATE	Initiates a search as in SELECT, then changes the contents of specific column data in each row in the resulting active set. UPDATE is a DML command.
DELETE	Initiates a search as in SELECT, then removes the resulting active set from the table. DELETE is a DML command.

*OLE DB Standard*

The ODBC standards design was to access only relational databases. Microsoft realized this as a limitation and developed a platform called Universal Data Access (UDA) where applications can exchange relational or non-relational data across intranets or the Internet, essentially connecting any type of data with any type of application. OLE DB is the Microsoft system-level programming interface to diverse sources of data. OLE DB specifies a set of Microsoft Component Object Model (COM) interfaces that support various database management system services. These interfaces can be used to create

software components that comprise the UDA platform. OLE DB is an API that allows for lower-level database access from a compiler. There are three types of COM components for OLE DB, OLE DB Data Providers, OLE DB Consumers, and OLE DB Service Providers. OLE DB Data Providers are data source-specific software layers that are responsible for accessing and exposing data. OLE DB Consumers are data-centric applications, components, or tools that use data through the OLE DB interfaces. Using networking terms, OLE DB consumers are the clients, and the OLE DB data provider is the server. OLE DB Service Providers are optional components that implement standard services to extend the functionality of data providers. Examples of these services include cursor engines, query processors, and data conversion engines.

The LabVIEW-based data collector module uses MDAC as data providers, which means MDAC needs to be installed for the data collector to function properly. The Microsoft Data Access Components (MDAC) are the practical implementation of Microsoft's UDA strategy. MDAC includes the ODBC, OLE DB, and ADO components. MDAC also installs several data providers that can be used to open a connection to a specific data source such as an MS Access database. Windows 2000 and Windows ME contain MDAC as part of the operating system. MDAC includes several OLE DB providers for various data sources. All data access in the data collector occurs through an OLE DB provider. Microsoft provides some relational data providers as part of the MDAC installation.

The main data providers used in the data collector are the OLE DB for ODBC and OLE DB for SQL server. OLE DB provider for ODBC acts as a conversion layer between



OLE DB interfaces and ODBC. The hierarchy of data interface layers between ADO and a database using the OLE DB provider for ODBC is shown in Figure 10.

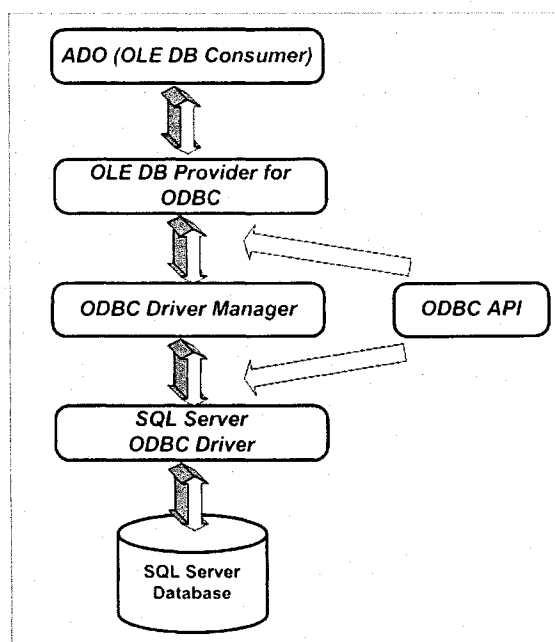


Figure 10. Communication path between ADO and SQL Server

### *ActiveX Data Objects (ADO)*

As mentioned previously, OLE DB is a system-level programming interface, and ADO is the application-level programming interface to diverse sources of data. ADO is an ActiveX wrapper to OLEDB so that any programming language or tool that supports COM can use the OLE DB technology through ADO. The LabVIEW-based data collector consists of ADO objects through invoke and Property Nodes.

The object model of ADO in this data collector is made up of three main COM objects, Connection, Command, and Recordset. A Connection object represents a unique session with a data source. In a client/server database system, it may be equivalent to an

actual network connection to the server. Depending on the functionality supported by the provider, some collections, methods, or properties of a Connection object may not be available. A Command object can be used to query a database and return records in a Recordset object, to execute a bulk operation, or to manipulate the structure of a database. Depending on the functionality of the provider, some Command collections, methods, or properties may generate an error when referenced. With the collections, methods, and properties, a Command object can define the executable text of the command (for example, an SQL statement) with the CommandText property. Alternatively, for command or query structures other than simple strings (for example, XML template queries) define the command with the CommandStream property. A command object can indicate the command dialect used in the CommandText or CommandStream with the Dialect property. It can also define parameterized queries or stored-procedure arguments with Parameter objects and the Parameters collection and indicate whether parameter names should be passed to the provider with the NamedParameters property. In some cases, a command object can execute a command and return a Recordset object if appropriate with the Execute method. Recordset object represents the entire set of records from a base table or the results of an executed command. At any time, the Recordset object refers to only a single record within the set as the current record. Recordset objects are used to manipulate data from a provider. When ADO is used, data is manipulated almost entirely using Recordset objects. All Recordset objects consist of records (rows) and fields (columns).

According to the ADO standard, each of these top-level objects can exist independently without the others. However, the data collector has a hierarchical structure

where the Connection object is necessary in order to use a Command or Recordset object.

The hierarchical structure of the ADO standard applied in the LabVIEW-based data collector is shown in Figure 11.

Some other ADO objects used in the ADO standard are the Record object, the Stream object, the Property object, the Error object, the Parameter object, and the Field object. They are also components in the ADO standard applied in the LabVIEW-based data collector. Their functions are described in Table 2.

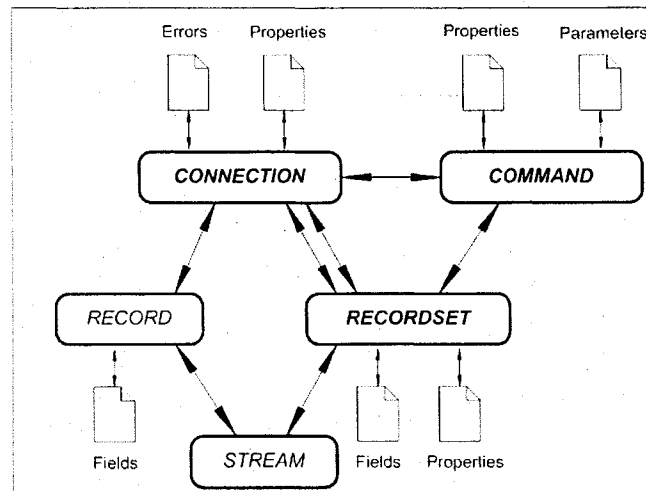


Figure 11. ADO object structure

Table 2.

## Other ADO Components

Components	Description
Record Object	This object represents a single row in a recordset.
Stream Object	This object represents binary data, usually stored in Unicode.
Property Object	This object is the building block of the other ADO objects. The properties collection contains only the properties added to the object by the data provider and does not contain the intrinsic properties of the object.
Error Object	This object represents a single error that occurs during operation.
Parameter Object	This object represents a single parameter for a Command object. Generally, parameters are used with any type of parameterized commands where an action is defined once but can have results changed depending on the variable values.
Field Object	This object represents a single column of data in a recordset.

*Integration of the Data Collector Module**ADO SubVI Structure*

The data collector module is integrated into LabVIEW applications as subVIs which can be called by the primary VI for database communication functions. According to the ADO hierarchical structure used, the data collector module has four groups of subVIs. Each of the groups represents one type of object applied in ADO database communication method. The four groups are the Connection Object VIs, the Command Object VIs, the Recordset Object VIs, and the SQL Statement VI. The advantage of

creating these four groups of subVIs in the names of ADO objects is to provide a simple and understandable structure of the subVIs for easier modular integration in LabVIEW applications.

Three subVIs are created in the Connection group, the ADO Create Connection VI, the ADO Open Connection VI, and the ADO Close Connection VI. VIs in this Connection group are used to create, open, or close a connection with a specified ADO object which is used for a database communication. They are the VIs used to initialize a database connection at the beginning and close the database connection when the tasks are done with the database. The ADO Create Connection VI initializes a database connection in LabVIEW application by calling an ActiveX function. The ADO Open Connection VI opens the database connection by calling a .NET function node. Also a SQL connection string must be specified to provide the database server's location path, the server's name, and the information for user identification. The ADO Open Connection VI uses the information to establish the communication channel between the LabVIEW application and the database hosted by a database server. The ADO Close Connection VI has the similar function as the ADO Open Connection VI, but its purpose is to disconnect the connection to the database. The block diagrams and functions of each subVIs in the Connection group are listed in Table 3.

Table 3.

## ADO Connection SubVIs

VI NAME	BLOCK DIAGRAM (LabVIEW Program)	FUNCTION
ADO Create Connection.vi		Initializes a database connection.
ADO Open Connection.vi		Opens a database connection.
ADO Close Connection.vi		Closes a database connection.

Three subVIs are created in the Command group, the ADO Create Command VI, the ADO Execute Command VI, and the ADO Set Command Text VI. The subVIs in this Command group are used to initialize a SQL command, set the SQL command statement, configure the SQL command, and execute the SQL command on the specified database. The ADO Create Command VI initializes a SQL command by calling an ActiveX function to connect to an ADO command object. The ADO Set Command Text VI is used to prepare a SQL command to be executed on the database. The ADO Set Command Text VI uses a .NET property node to set the SQL command type and the command text string. Also the .NET property node sets a Boolean datum to identify the status of the command, and an integer datum to limit the valid life time of the command. The command text

string can set all the five types of SQL commands, as mentioned previously, to retrieve or manipulate data in the specified database. The ADO Execute Command VI uses a .NET invoke node to execute the SQL command prepared by the Set Command Text VI.

Parameters associated with the command execution need to be specified as an input data string. The recordset affected by the command execution will be saved and sent as an output data string. The block diagrams of each subVIs in the Command group are listed in Table 4.

Table 4.

ADO Command SubVIs

VI NAME	BLOCK DIAGRAM (LabVIEW Program)	FUNCTION
ADO Create Command.vi		Initializes a SQL command.
ADO Set Command Text .vi		Prepares a SQL command.
ADO Execute Command.vi		Executes a SQL command and generate output.

Three subVIs are created in the Recordset group: the ADO Create Recordset VI, the ADO Open Recordset VI, and the ADO Close Recordset VI. The Recordset object represents the entire set of records from a base table or the results of an executed command. The Recordset object refers to only a single record as the current record within the set. Recordset objects are used to manipulate data from a provider. The ADO Recordset subVIs are designed to create a recordset during the process of database manipulation, save the retrieved data record in the recordset, manipulate data in the recordset, save changes to the recordset, and display the updated recordset to an interface or send back to update the corresponding record in the database.

The ADO Create Recordset VI initializes a recordset in LabVIEW application by using an ActiveX function. The ADO Open Recordset VI calls the invoke .NET node to open an existed recordset. The command text string needs to be specified as a text string input which gives the executable command to manipulate the data in the recordset. Also the ADO connection needs to be specified as another input to the subVI, giving the existed database connection path. Options are provided with the Open Record VI to configure some other actions that may be taken on the recordset, such as Get rows, Get strings, Requery, and Update. The ADO Close Recordset VI uses a .NET invoke node to close the existing recordset after the updated recordset has been used. The existing recordset connection needs to be provided to the Close Recordset VI as an input. It is an executable VI without any output. The recordsets created and used in a database connection session will not be kept in the LabVIEW application or the database server. They are only temporary files created for data manipulation on the database; the updated recordsets will be used in the same connection session, and affected data is updated and



used at the end of the session. So keeping the recordsets created in any previous connection session is meaningless. The block diagrams and functions of each subVIs in the Recordset group are listed in Table 5.

Table 5.

ADO Recordset SubVIs

VI NAME	BLOCK DIAGRAM (LabVIEW Program)	FUNCTION
ADO Create Recordset.vi		Initializes a recordset.
ADO Open Recordset.vi		Opens and manipulates a recordset.
ADO Close Recordset.vi		Closes a recordset.

These three ADO object VI groups are the fundamental subVIs that establish the communication between LabVIEW applications and the database. They provide the methods for LabVIEW applications to retrieve, manipulate, and update data in the

specified database. Based on these fundamental ADO object subVIs, a higher level SQL VI is created for easier database communication, which forms the fourth group of the subVIs for LabVIEW applications. This SQL Execute VI is created to perform a SQL query on a database, and if necessary, get the rows that are returned by the query. The input and output connections are indicated in the icon of the subVI in Figure 12.

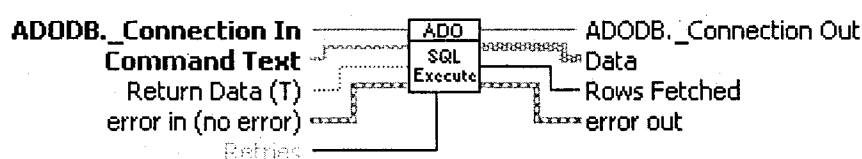


Figure 12. SQL Execute VI

To issue the SQL Execute VI, an ADO Connection object must be created and wired to the ADODB.\_Connection In input. Command text is the SQL statement string which specifies the SQL command to be executed on the database. The Command text must be terminated with a semicolon, as in the SQL programming. Return Data is a Boolean, which, when True (the default), tells the VI to retrieve data from the query. If a SQL command without return data is called by the VI, this Boolean must be set to FALSE, such as when an UPDATE command is called). Data is a 2D array of strings which contains the rows returned by the SQL command statement. Rows Fetched gives the number of rows returned by the function. Rows Fetched has no output when the Boolean is set to FALSE. Retries is an integer that specifies the number of retries that this SQL command can be executed. Normally this should not be set. However, it is useful when the LabVIEW application is to connect to a very slow database or the records are being

locked out while the SQL command is trying to execute. The block diagram of this SQL Execute VI is shown in Figure 13.

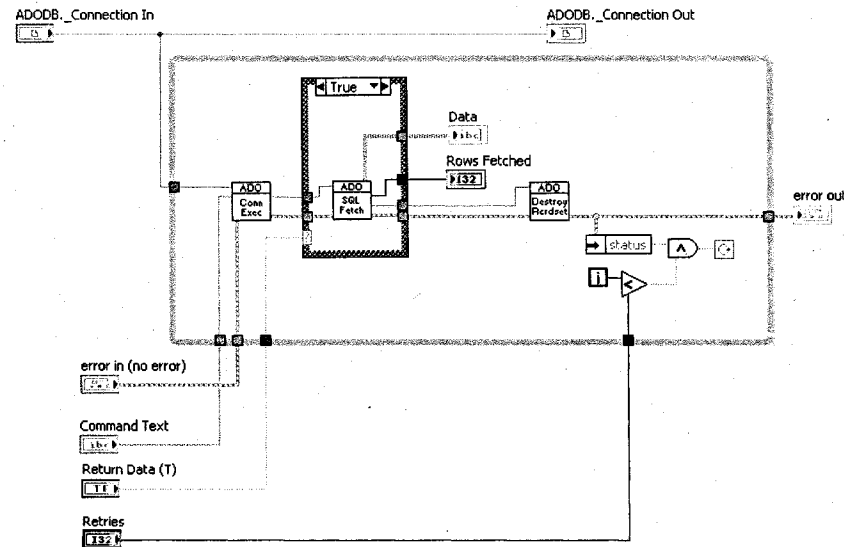


Figure 13. Block diagram of the SQL Execute VI

In this block diagram, a while loop is used to set the number of retries. A case structure is used to execute SQL command with returned data or without returned data separately. This higher level VI uses basic ADO object subVIs to perform the SQL query. It provides an easier way to integrate the module into LabVIEW applications, and simplifies the programming process when a huge number of database related commands need to be executed.

#### *ActiveX and .NET Functions*

LabVIEW programming language provides ActiveX and .NET functions as higher level communication functions. ActiveX functions are used to pass properties and methods to and from other ActiveX-enabled applications, such as ADO supported

databases. Several ActiveX functions are available in LabVIEW programming language; the one used in construction of the subVIs is called Automation Open function. The input and output connections of the Automation Open function are shown in Figure 14.

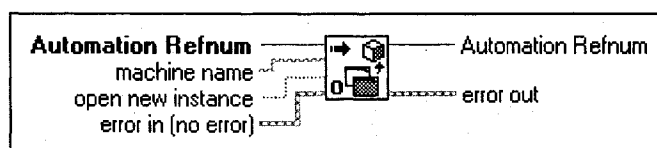


Figure 14. Automation Open Function VI

Automation Open points to a specific ActiveX object by specifying an automation reference number which is associated with the ActiveX object in Windows. Machine name indicates on which machine the VI should open the Automation Refnum. If no machine name is given, the object is opened on the local machine. If open new instance is TRUE, LabVIEW creates a new instance of the Automation Refnum. If FALSE (default), LabVIEW tries to connect to an instance of the refnum that is already open. If the attempt is unsuccessful, LabVIEW opens a new instance. Error in describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the error in value to error out. This VI runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in error out. Error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces. After the reference number is opened, it

can be passed to other ActiveX functions. Only creatable classes can be wired as inputs to this function. If a machine name is wired, the object opens on the remote machine.

Otherwise, the object opens on the local machine. Because of its connection capability to a specified ActiveX object, the Automation Open function is used to program the ADO Create Connection VI, the ADO Create Command VI, and the ADO Create Recordset VI.

Another type of advanced communication functions used in programming the subVIs is the .NET function. .NET functions are used to create .NET objects and set properties or methods on those objects. The ADO object is one type of the .NET objects. Therefore, .NET functions can be used in programs to set properties or methods for ADO objects. The .NET functions used in this study are the Invoke Node and the Property Node. The Invoke Node invokes a method or action on a referenced object. Most methods have associated parameters. The Property Node reads and/or writes properties of a referenced object. The Property Node automatically adapts to the class of the referenced object. LabVIEW includes Property Nodes preconfigured to access ActiveX properties. The input and output connections of the Invoke Node are shown in Figure 15.

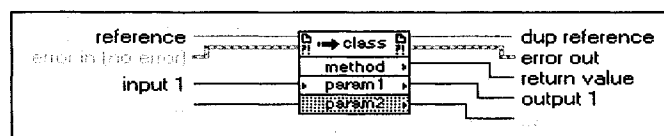


Figure 15. Invoke Node Function

Reference is the reference number associated with the ActiveX object on which a method is invoked or an action is performed. If the Invoke Node class is an Application

or VI, a reference number is not necessary. For the Application class, the default is the current instance of LabVIEW. For the VI class, the default is the VI containing the Invoke Node. Error in describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the error in value to error out. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in error out. Input 1..n is an example input parameter of a method. Dup reference returns reference unchanged. Error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces. Source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. Return value is an example return value of a method. Output 1..n is an example output parameter of a method. To select the class on which to execute the method, wire the refnum to the reference input. The node adapts to the class automatically. Parameters with a white background are required inputs and the parameters with a gray background are recommended inputs. The ADO Open Connection VI, the ADO Close Connection VI, the ADO Command Execute VI, the ADO Open Recordset VI, and the ADO Close Recordset VI are all programmed with the .NET Invoke Node function.

Another .NET function used is the Property Node. The input and output connections of the Property Node function are shown in Figure 16.

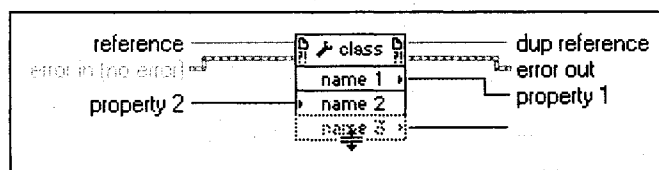


Figure 16. The Property Node function

Reference is the reference number associated with an ActiveX object across a TCP connection. If the Property Node class is an Application or VI, a reference number is not necessary. For the Application class, the default is the current instance of LabVIEW. For the VI class, the default is the VI containing the Property Node. Error in describes error conditions that occur before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the error in value to error out. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function runs, it runs normally and sets its own error status in error out. Property 2..n is an example of a property that needs to be written. Dup reference returns reference unchanged. Error out contains error information. If error in indicates that an error occurred before this VI or function ran, error out contains the same error information. Otherwise, it describes the error status that this VI or function produces. Property 1..n is an example of a property that needs to be read.

To select the class on which to execute the property, wire the refnum to the reference input. The node adapts to the class automatically. To get property information, right-click the node and select Change to Read from the shortcut menu. To set property information, right-click the node and select Change to Write from the shortcut menu. If a property is read only, Change to Write is dimmed in the shortcut menu. The node executes each

terminal in order from top to bottom. If an error occurs on a terminal, the node stops at that terminal, returns an error, and does not execute any further terminals. Right-click the node and select Ignore Errors Inside Node from the shortcut menu which can ignore any errors and continue executing further terminals. The error out cluster reports which property caused the error. This .NET Property Node function is used to program the ADO Set Command Text VI, so that the command string can be written to the command property.

### *Communication with the Database*

#### *MSDE Database*

The Microsoft SQL Server 2000 Desktop Engine (MSDE) is a new database technology released by Microsoft to compete with Oracle and IBM DB2. Because it has several advantages over other database technologies, MSDE is selected as the database server in this integration module. First of all, MSDE has all the features of the powerful SQL server except the limited number of clients. For example, MSDE has client-server architecture. Database operations occur on the database server driven by the database engine, not on the client. Also MSDE can host as many different databases as required and keep tabs on them all. The limited number of clients is not the major concern in this study since 25 clients' accesses will be enough for this study. Secondly, MSDE is supported by Microsoft Internet Information Services (IIS) which is an service integrated with all the current WINDOWS operating systems, like WINDOWS 2000 and WINDOWS XP. To set up the MSDE database server, no extra services are required to be installed. This will save the process time and capacity of the server. The last advantage is that MSDE is a free version of database server from Microsoft. They made it free and



redistributable so they could convince as many developers as possible to use it. The integration module developed in this study is aimed to be used in small-to-middle sized manufacturing enterprises, so the investment required is also an important factor.

However, MSDE does not include a user interface. ASP.NET Web Matrix Project is used to function as the interface to access the database. Microsoft ASP.NET Web Matrix Project is a free, light-weight community-supported web development tool for quickly building ASP.NET Web applications. Web Matrix project is written completely using C# and the .NET Framework. Specifically, the UI is built using Windows Forms (System.Windows.Forms). ADO.NET is used for data access (System.Data) and XML Web Services are used to communicate with backend services on [www.asp.net](http://www.asp.net) (System.Web.Services). And the ASP.NET designer APIs, integrated with the .NET Framework, are used for hosting ASP.NET server controls within designer (System.Web.Design and System.Web.Mobile.Design). Therefore, Web Matrix Project is compatible with ADO and .NET functions which are the main structure of the integrated module. It can communicate with the other parts of the module which also developed in .NET development environment – clients, server and services.

In this study, a virtual MSDE database server is set up on the local machine named db2000. A MSDE database is created inside the server named mydb. The security mode of the database server is set to SQL server security mode so that SQL username and password are used for user authentication instead of WINDOWS authentication mode. These functions are accomplished by issuing the following commands in the WINDOWS command prompt line.

```
c:\sql2Ksp3 >Setup.exe INSTANCENAME=db2000 USERNAME=SA
PWD=5101 SECURITYMODE=sql
```

Figure 17. Commands for SQL server security mode

After the virtual database server is set up, the MSDE database service will be enabled as a local service running on the local machine. The status of this database service can be monitored and modified from the system services window of the local machine as shown in Figure 18. In this window, the MSDE database service is highlighted with the service name "MSSQL\$DB2000". This service can be enabled, disabled, started, or stopped from this system services window. Also, it can be set in manual started or automatic started mode. In manual mode, the service needs to be started manually after the machine is restarted. In automatic mode, the service will be automatically started when the operating system is started.

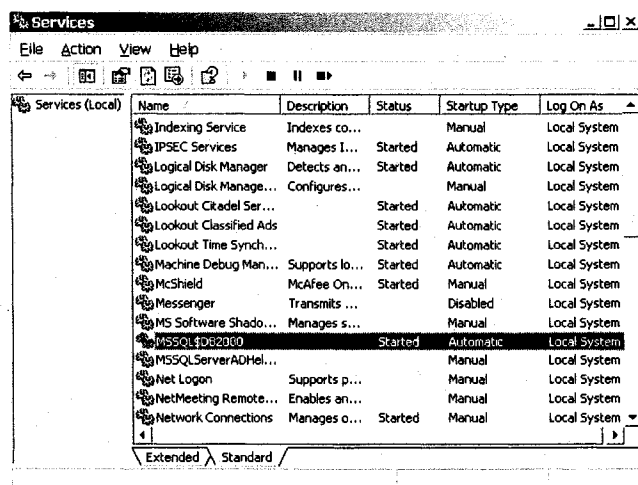


Figure 18. Local system service window

Web Matrix project provides the interface to connect to the MSDE database server. For proper connection, the SQL server username and password are required as shown in Figure 19.

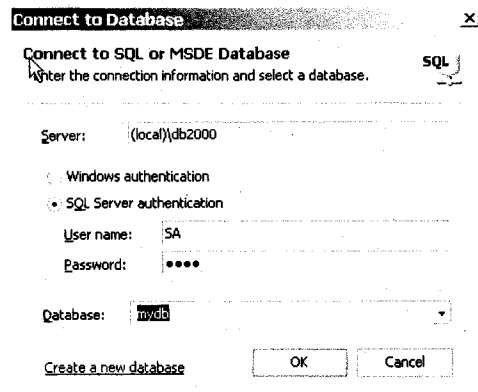


Figure 19. SQL Server authentication window

Once the database server is connected from Web Matrix project, the database server and all its components will be displayed in the data pane of the interface window as shown in Figure 20. In this study, the location path of the database is “(local)\db2000.mydb”, in which db2000 is the SQL server’s name, and mydb is the name of the database used in the integration module. The database can be expanded from the data pane to display all the tables and stored procedures in the database. In this study, two main tables used are the DevStatus table and the Measurement table storing the device status data and measurement data. The interface that is used to create and design a table in the database is shown in Figure 21.

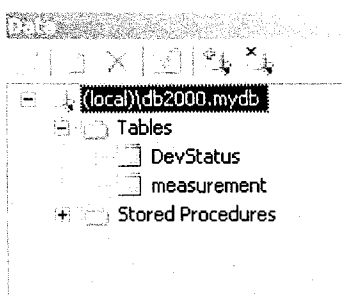


Figure 20. Data pane of the database interface

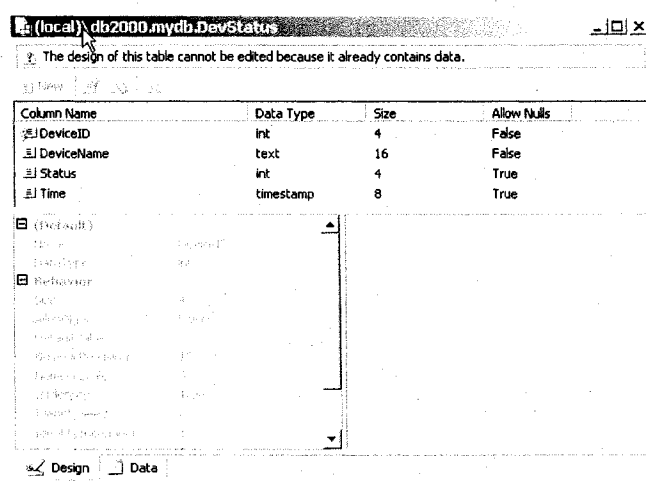


Figure 21. Design view of the database table

### *Communicating with MSDE from LabVIEW*

As discussed previously, six ADO Object VIs and one SQL VI have been programmed to enable the communication between LabVIEW control applications and SQL database. These VIs must be programmed properly into the LabVIEW control application to enable the database access. In the integration module established in this study, a process control application is used to simulate the plant floor control process in manufacturing industry. The data that needs to be written into the MSDE database from

this control application are the device status data and the measurement values. The device status data are the Boolean data to indicate the ON/OFF status of each device in the process, such as the status of valves and pumps. The device status data are recorded in the table of DevStatus in the mydb database in MSDE database server. The measurement values are the measurements of temperatures and levels of the product in the control process. These values are float data that are recorded in the Measurement table in the database. Part of the block diagram program with the database VIs is shown in Figure 22.

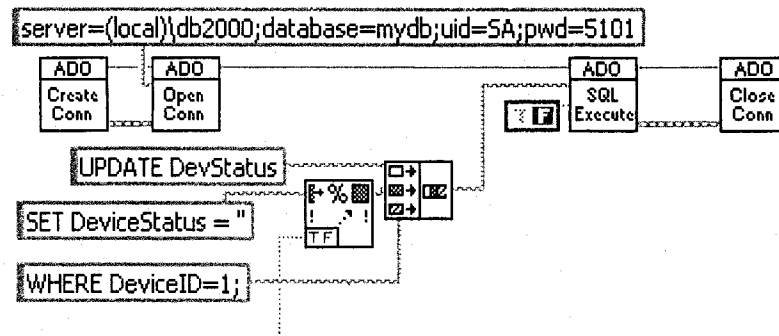


Figure 22. Block diagram of the database VIs

The block diagram program shown above is part of the data collector module in the LabVIEW control application to get a real-time device status data from the LabVIEW application and update the corresponding Boolean data in MSDE database. This part of the data collector module shows the simple method required to program the database VIs in LabVIEW control applications. First the ADO Create Connection VI initializes a connection with ADO based database by calling an ActiveX function. Then the connection to a specified MSDE database is opened by the ADO Open Connection VI

with the location path, database name, username, and password specified by a SQL connection statement in the command text string. In this case, the higher level SQL Execute VI is issued to execute the SQL UPDATE command. As the UPDATE command is issued, there is no recordset to be returned by the command. Therefore, the Boolean value of Return Data input is set to False which means there is no return data for this command. The SQL UPDATE command text is compiled by two string operation VIs before it is connected to the SQL Execute VI as the command input. The Format into String VI helps to format the Boolean value of device ON/OFF status into the command string. The Concatenate Strings VI combines all the statements to form a complete SQL UPDATE statement which will be executed by the SQL Execute VI. It follows the format of SQL language statement as shown below.

```
UPDATE { table_name}
{SET column_name = expression [WHERE condition_list] }
```

Figure 23. SQL statement format

This block diagram program shows only one execution cycle of the SQL UPDATE command to get data and update the database record. During the control process, the real-time data need to be recorded and historical data saved with timestamp. To achieve this function, the execution program is put into a While Loop so that it can be executed repeatedly to get real time status data. The execution rate of the SQL UPDATE command can be controlled by a Wait Until Next ms Multiple timer inside the While Loop. This timer is used to set the execution rate by specifying the milliseconds that it needs to wait

for next execution. In this case, it is set to execute every 250 ms. It means that the device status data in the database will be updated every 250 ms which is very close to the real time manner. The updated device status can be displayed in the Web Matrix project interface as shown in Figure 24.

	DeviceID	DeviceName	Status	Time
►	0	valve1	0	Byte[] Array
	1	valve2	1	Byte[] Array
	2	pump1	0	Byte[] Array
	3	heater	1	Byte[] Array
	4	pump2	1	Byte[] Array
	5	valve3	1	Byte[] Array
	6	valve4	1	Byte[] Array
*				

Figure 24. DeviceStatus data table

## The Database Server and the Web-based Interface

### *Introduction*

The basic features of the web-based interface were introduced at the beginning of this chapter. The most critical feature is the real-time communication with the MSDE database server. The web-based interface is developed in ASP.NET environment. The main objects and methods used in the development of the interface will be explained in this session. Also the web features of the interface, which are enabled by both LabVIEW and ASP.NET environment, will be examined in detail.

ASP.NET provides a platform for web application development. Microsoft tried hard to make the development process easy by providing quite a few controls and forms that can be used as GUI (Graphic User Interface) objects for application development. But

due to the complex methods and procedures required by dynamic web applications, coding is still an important and necessary tool for ASP.NET programming. In the programming process of the web-based interface, two steps are applied, logic design and coding.

#### *Procedure Logic*

In the interface, the object used to contain the retrieved data from the database server is DataGrid, which is a data bound list control that displays the items from data source in a table. The DataGrid control also can be used to select, sort, and edit these items. The procedure logic used to build the methods and events to enable the dynamic communication between the DataGrid control and the database follows the pattern in Figure 25.

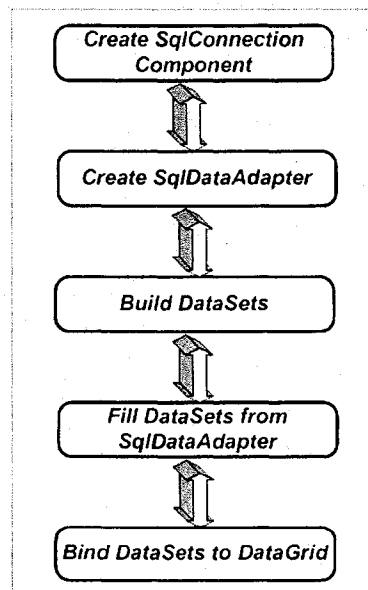


Figure 25. Procedure logic for DataGrid control



This procedure logic pattern is similar to the programming method used in the LabVIEW-based data collector. It uses the method of SQL statements to communicate with the MSDE database server. First, a SqlConnection component is built to establish the connection channel by specifying the connection string in VB (Visual Basic) coding. Then a SqlDataAdapter object is called and configured. The DataAdapter supplies the methods and properties to connect to a database, retrieve data, and populate the DataSet with that data. The DataAdapter object contains two key methods: Fill and Update. The Fill method takes a DataSet parameter to fill the records from a Command object's SQL statement. Command object is used when dealing with DataAdapters. The Command object always contains the SQL statements that interact with the data source. The Fill method of the DataAdapter refreshes the data in the DataSet based on the Command object's SQL statement. After the Fill method is completed, the connection is closed automatically.

After a SqlDataAdapter is created, DataSet needs to be built. Each data table in the interface, which needs a dynamic connection to the database server, must have a DataSet. DataSet is used to hold the data that the DataAdapter is going to retrieve. In ADO.NET, there are two types of DataSets: typed and nontyped. A typed DataSet contains strong type information about the fields and the allowable data types. In a typed DataSet, the field names can be referenced as any other property. A nontyped DataSet does not contain strong type information about the fields in the DataSet. Their main difference is how the field names are referenced when working with the DataSet. In this case, typed DataSet is used to work with the SqlDataAdapter. At the last, data retrieved by the SqlDataAdapter needs to be filled into the DataSet by VB coding. DataSet is similar to the RecordSet

created in ADO method, which is also used to temporarily hold data. In order to display the data saved in DataSet, DataSet needs to be bound with a specified DataGrid. The way the data is displayed in DataGrid can be customized by modifying the properties of the DataGrid. Figure 26 shows the schema of a DataSet used in programming the interface.

◆ E	DevStatus	(DevStatus)
⌘ E	DeviceID	int
E	DeviceName	string
E	Status	int
E	Time	base64Binary

Figure 26. DataSet for DevStatus table

### *Coding Structure*

The VB coding to realize the procedures above follows the structure explained in Figure 27.

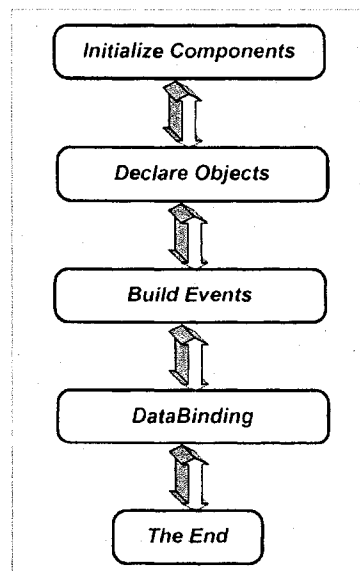


Figure 27. Structure of VB coding

The code begins with the component initialization by defining the class namespace that will be used in this program. Then each object that will be used must be declared, such as SqlConnection, SqlCommand, SqlDataAdapter, DataSet, and so on. The most important part in coding is building the events that realize the procedures and methods. An example of the event coding is shown in Figure 28 and the procedure coding is shown in Figure 29.

```
Me.SqlUpdateCommand2.CommandText = "UPDATE measurement SET Name
= @Name, [Tem(F)] = @Param5, [Level(%)] = @Param6, [AveTem(F)] = @Param7, [AveLevel(%)] = @Param8 WHERE (ID =
@Original_ID) AND ([AveLevel(%)] = @Original_AveLevel___ OR @Original_AveLevel___
IS NULL AND [AveLevel(%)] IS NULL) AND ([AveTem(F)] = @Original_AveTem_F_ OR
@Original_AveTem_F_ IS NULL) AND ([Level(%)] = @Original_Level___ OR @Original_Level___ IS NULL) AND ([Tem(F)] =
@Original_Tem_F_ OR @Original_Tem_F_ IS NULL); SELECT ID,
Name, [Tem(F)], [Level(%)], [AveTem(F)], [AveLevel(%)] FROM measurement WHERE (ID =
@ID)"
```

Figure 28. Example of Event coding

```

Public Sub Page_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    'Put user code to initialize the page here
    If Not IsPostBack Then
        SqlDataAdapter1.Fill(DataSet11)
        DataGrid1.DataSource = DataSet11
        DataGrid1.DataBind()
        SqlDataAdapter2.Fill(DataSet21)
        DataGrid2.DataSource = DataSet21
        DataGrid2.DataBind()
        ' The DataSave function will be added later.
    End If

```

Figure 29. Data binding procedure

### *Web-based LabVIEW Control Panel*

#### *Physical Setup of the System*

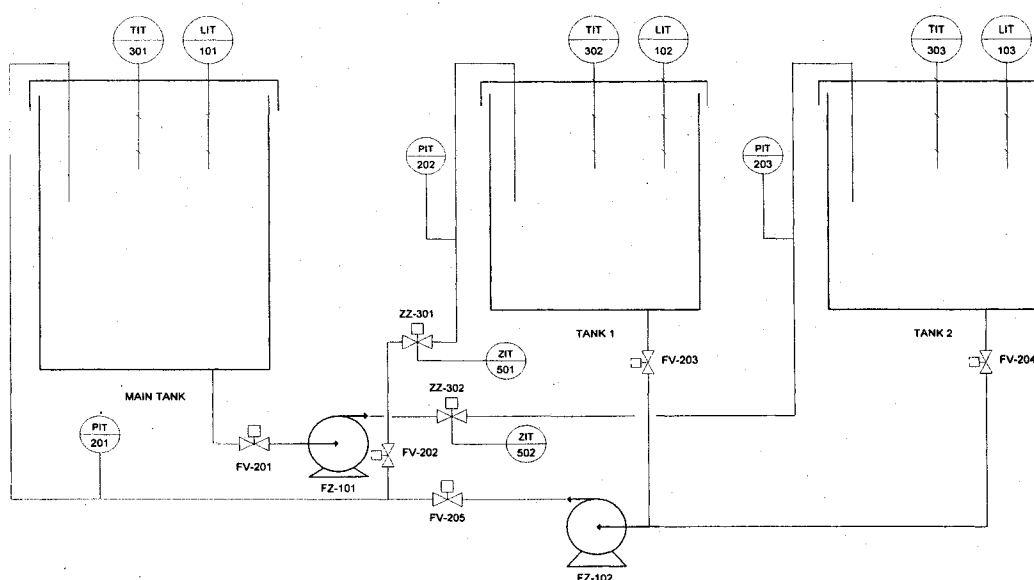


Figure 30. Components of the wet process trainer

The integration module in this study uses a wet process trainer as a plant-floor control process to realize the web-based integration. The process is comprised of three tanks, two pumps, five discrete valves and two continuous valves. Temperature transmitters and level transmitters were used to monitor each tank's water level and temperature. Two pressure transmitters were installed to monitor the incoming flow pressure of tank 1 and tank 2. A flow transmitter was installed to measure the incoming flow rate of the main tank. Table 1 below provides a detailed list of the input and output devices in the physical setup of the system and their address assignment of corresponding control components.

Table 6.

## I/O Addressing of the Control Process

LabVIEW Addressing in Programming		Device	Device Description
FP@139_102_29_56\cFP-DO-400@7 (Digital Output Module)	\Channel 0	FZ-101	Pump 1
	\Channel 1	FZ-102	Pump 2
	\Channel 2	FV-201	Discrete valve 1
	\Channel 3	FV-202	Discrete valve 2
	\Channel 4	FV-203	Discrete valve 3
	\Channel 5	FV-204	Discrete valve 4
	\Channel 6	FV-205	Discrete valve 5
FP@139_102_29_56\cFP-AO-200@5 (Analog Output Module)	\Channel 0	ZZ-301	Continuous control valve 1
	\Channel 1	ZZ-302	Continuous control valve 2
FP@139_102_29_56\cFP-RTD-124@2 (Temperature Module)	\Channel 0	TIT-301	Temperature Transmitter 1
	\Channel 1	TIT-302	Temperature Transmitter 2
	\Channel 2	TIT-303	Temperature Transmitter 3
FP@139_102_29_56\cFP-AI-110@1 (Analog Input Module)	\Channel 2	LIT-101	Level Transmitter 1
	\Channel 3	LIT-102	Level Transmitter 2
	\Channel 4	LIT-103	Level Transmitter 3
	\Channel 5	PIT-201	Flow Rate Transmitter
	\Channel 6	PIT-202	Pressure Transmitter 1
	\Channel 1	PIT-203	Pressure Transmitter 2

*LabVIEW Interfacing*

A virtual interface programmed by LabVIEW graphical language provides a control panel for users to interact with the wet process trainer through FieldPoint Ethernet communication and the communication between the FP controller and I/O Modules.

The communication between the FP controller and I/O Modules is similar among different types of network modules. Each I/O module cycles through its internal routine

of sampling all channels, digitizing the values and updating the values on the module channel registers (buffer). This cycle time is set for each module and is specified as the all channel update rate.

FiledPoint Ethernet communication uses an asynchronous communication architecture called event-driven communication. The network module automatically sends updates to a client when data changes. The server then caches the data from I/O modules and uses it to respond to read requests from the virtual interface. The network module scans all I/O channels with subscriptions to determine if a value has changed, comparing the current value to the cached value for each channel. If a change has occurred, the network module puts the difference between the two values in the transmit queue. The FP Server receives this information and sends an acknowledgement to the network module. The network module periodically sends and receives a time-synchronization signal so that it can adjust its clock and provide proper timestamping. When signals do not change over long periods of time, the client sends periodic re-subscribe messages to verify that the system is still online.

The virtual interface programmed for this wet process trainer is explained in former sessions of this chapter, and the diagram of the direct control panel is shown in Figure 8 on page 42.

### System Operation

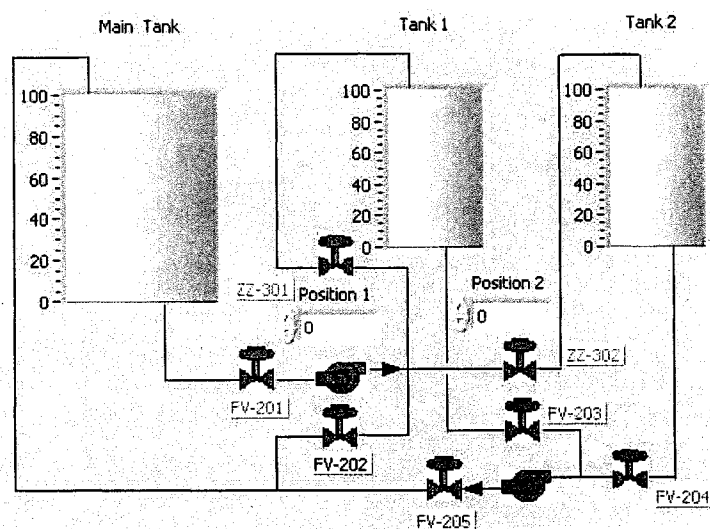


Figure 31. Process control diagram

Figure 31 shows the process control part on the virtual interface. This process diagram demonstrates the real process of the wet process trainer. Three tank indicators represent the main tank, tank 1 and tank 2 in the wet process trainer respectively. The green line of each tank will indicate the current water level which can vary from 0 percent to 100 percent. In this diagram, valves and pumps are controls for their corresponding part in the real process. FV-201, FV-202, FV-203, FV-204, and FV-205 are controls for discrete valves. These five discrete valves and two pumps are controlled by ON/OFF Boolean signals; their status can be changed by a mouse-click on them. The color of each control represents different status of these controls. Red represents OFF status while green represents ON status. ZZ-301 and ZZ-302 represent the two continuous control valves in the real process. The status of them is controlled by the



digital control below each valve symbol. The value of each digital control can be changed by clicking the arrows beside it from 0 to 100 with the increment of 10. The value of 0 represents close status of the valve, and the value of 100 represents the totally open status of the valve. The color of ZZ-301 and ZZ-302 will be changed to green color when the value of their digital control is equal or greater than 10 to indicate an ON status. Otherwise, it will be changed to red one indicating an OFF status.

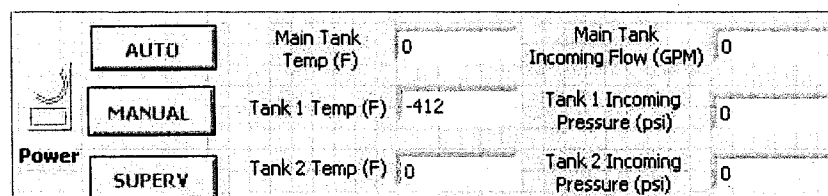


Figure 32. Mode controls and digital indicators

Figure 32 displays the part of the virtual interface with power control, mode controls and digital indicators. The power control is used to turn on/off the system, which will be in green color when the system is running. The six indicators to the right of the mode buttons are digital indicators displaying current values of tank levels, incoming flow rate, and incoming pressures. The buttons with red labels are system mode controls. This virtual interface provides three different modes for process control which include supervision mode, manual mode, and auto mode. Users can be assigned different control capabilities when different modes are enabled.

In supervision mode, all the valves and pumps in the process can be controlled by users. This mode can be enabled only for maintenance and trouble-shooting purpose.

In manual mode, the status change of valves and pumps will depend on both the current situation of the system and the commands from the user. For example, if the water level of the main tank is lower than 20 percent or valve FV-201 is closed, pump 1 can not be activated even if the user intends to do so by clicking on the pump symbol in the virtual interface.

In auto mode, the system will control the valves and pumps automatically, depending on the water levels of each tank. The only part that users can control is the continuous control valves. Users can adjust the percentage of the continuous valves by changing the value of the digital controls.

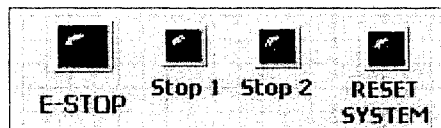


Figure 33. Emergency Stop and Reset buttons

The virtual interface provides three emergency stop buttons and one reset button. E-Stop button will stop the whole system when pressed. Stop 1 and Stop 2 button will disable pump 1 and pump 2 respectively when pressed. Reset button is used only in auto mode to reset the system when the main tank level reaches its limits.

### Implementing Security

#### *Security Methods for Three Components*

The integration module consists of three main components, a LabVIEW-based process data collector, a virtual server, and a web-based interface. The LabVIEW-based

process data collector is a software module integrated with the LabVIEW control application. No direct access is possible to this data collector from remote users. The data collector is installed on the local control server, and is a server side module, so the security of this part can be implemented by setting the security authentication method on the server machine. On the virtual database server, Web Matrix project provides an interface for direct access to the database. Its security is implemented by the SQL authentication method. The interface can only be allowed to access the database when the server name, server location path, SQL username and password are provided by the user. The last part of the integration system, also the most important part of the security issue, is the access to the web-based interface. From the interface, not only the real-time process data can be read, but also the direct control to the plant-floor process can be accessed. Therefore, in this study, an ASP login page is programmed to provide a form authentication method. The login page is set to be the start page in which users need to provide username and password before they actually gain access to the interface page.

#### *Forms Authentication for the Web-based Interface*

##### *Forms Authentication*

Forms Authentication in ASP.NET is handled by a special FormsAuthentication class. This class contains a number of static (or Shared) methods that can identify users via a login form. When an unauthenticated user visits a restricted page on the Web site, they will be automatically directed to the specified login form. Once they successfully log on, an authentication cookie can be issued to prevent authenticated users from having to log in time and time again.

A security system contains two very important features, authentication and authorization. Authentication is the means by which the identity of the user is validated against a known Authority, like Active Directory, Database Store, Microsoft Passport Account, etc. If the credentials can't be validated, then the Authentication process fails and the user will assume the Identity of IUSR\_Anonymous. The only way to determine who is a particular visitor, is to authenticate them by having them provide user credentials (a username/password, usually). Authorization occurs after Authentication and involves using information obtained during the Authentication process to determine whether to grant or deny access to a given resource based on that user's role in the Application.

Forms Authentication uses cookies to allow applications to track users throughout their visit. When a user logs in via forms authentication, a cookie is created and used to track the user throughout the site. If the user requests a page that is secure and has not logged in, then the user will be redirected to the login page. Once the user has been successfully authenticated, he/she will be redirected to their originally requested page. The use of Forms Authentication involves the configuration or development of three files in the web application, the Web.config file, the login web page, and the redirect web page.

#### *The Web.config File*

The Web.config contains all of the configuration settings for an ASP.NET application. Inside the Web.config file, both authentication and authorization sections need to be configured. The sections of the Web.config file programmed in the web-interface application are shown below.

```

<authentication mode="Forms" />
  <forms loginUrl="login.aspx" protection="All" timeout="30">
    <credentials passwordFormat="Clear">
      <user name="jeff" password="7336">
      <user name="mike" password="7336">
    </credentials>
  </forms>
</authentication>
<authorization>
  <deny users="?" /> <!-- Deny all users -->
</authorization>

```

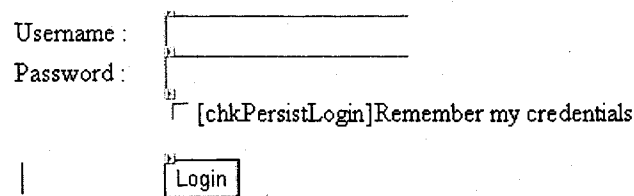
Figure 34. Web.config

The mode attribute for the <authentication> configuration section sets the authentication mode to Forms. Inside the <forms> section, the name attribute can be set which sets the name of the cookie. The path to the root of the application can be set. The loginUrl attribute sets a local page named login.aspx which is the ASP page to provide the login page. It can also be assigned as a URL path. Setting protection to all is the suggested value. This means that the cookie will be both encrypted (using the Triple DES data-encryption standard) and validated. The validation algorithm comes from the machineKey element located in Machine.config. This data validation ensures that the cookie data were not tampered with during transit (someone sniffing traffic and responding with a modified cookie). The timeout attribute refers to the number of minutes before a cookie expires and the user must log in again.

In the authorization section, the file needs to ensure that no unauthenticated users can access the application. The "?" means anonymous users, so a deny flag is set for all anonymous users.

### *Login.aspx Page*

Login.aspx is an ASP web page that is used to implement the Forms Authentication. All user authentication logic is performed here. In the forms coding, the System.Web.Security namespace refers to all methods of authentication that reside in this namespace. FormsAuthentication is a class of the System.Web.Security namespace. A simple server-side form is set up with one textbox and one password input for username and password as shown in Figure 35. A check box is included in case the user wants to have a permanent cookie set.



Username :

Password :

☐ [chkPersistLogin]Remember my credentials

Figure 35. The server-side login form

The Submit button has an event "onclick" which runs a sub called ProcessLogin. Inside ProcessLogin the Authenticate method of the FormsAuthentication class is executed, passing in the given username and password. This method checks the credential's tags inside the Web.config for the username and password. If they match, then the RedirectFromLoginPage will be executed with the username and persistent cookie state (if checked). This method then writes a cookie to the user's machine to track them and ensure they are authenticated. If the username and password do not match, then an error occurs and the user is notified. The VB codes written to call for events of authentication and authorization are as below.

```

<script language="VB" runat="server">
Sub ProcessLogin(objSender As Object, objArgs As EventArgs)
    If FormsAuthentication.Authenticate(txtUser.Text, txtPassword.Text) Then
        FormsAuthentication.RedirectFromLoginPage(txtUser.Text,
chkPersistLogin.Checked)
    Else
        ErrorMessage.InnerHtml = "<b>Something went wrong...</b> please
re-enter your credentials..."
    End If
End Sub
</script>

```

Figure 36. Coding example for authentication and authorization

### *Configuring the Interface Page*

This is the page the user is requesting or trying to access. Again the System.Web.Security namespace is referenced since authentication methods and properties are needed for the configuration. This page contains a simple div element that will display the current credentials and authentication type used. Also, a sign out input element runs a sub that deletes the user's cookie. They are then redirected to the login page. In the Page\_Load event, it is checked to see if the user has been authenticated using the User.Identity.IsAuthenticated property. This returns a Boolean value indicating whether or not the user has been authenticated. If the user is authenticated, the current user's name and the authentication method used are displayed. The currently logged on user's name can be assessed with the User.Identity.Name property.

User.Identity.AuthenticationType returns the mode of authentication used. A SignOut procedure is used that allows the user to sign out and deletes the cookie from the user's computer. This will even delete a persistent cookie. The VB code added into the web-based interface coding to realize the above security function is as below.

```

<script language="vb" runat="server">
Sub SignOut(objSender As Object, objArgs As EventArgs)
    'delete the users auth cookie and sign out
    FormsAuthentication.SignOut()
    'redirect the user to their referring page
    Response.Redirect(Request.UrlReferrer.ToString())
End Sub
Sub Page_Load()
    'verify authentication
    If User.Identity.IsAuthenticated Then
        'display Credential information
        displayCredentials.InnerHtml = "Current User : <b>" & User.Identity.Name
        & "</b>" & _
        "<br><br>Authentication Used : <b>" &
        User.Identity.AuthenticationType & "</b>"
    Else
        'Display Error Message
        displayCredentials.InnerHtml = "Sorry, you have not been authenticated."
    End If
End Sub
</script>

```

Figure 37. Coding for interface security

## Testing and Analysis

### *Introduction*

The critical component in this integration module is the LabVIEW-based data collector. By using this data collector, no third-party software is required for the data collection process. In this study, a queuing network model is established to analyze the effect of this integrated data collector on the existing computer system, the control server. A statistical method, one-way ANOVA, is applied to evaluate the effect on the system. The SPSS statistical software is used for the analysis.

### *Queuing Network modeling*

Queuing network modeling is a particular approach to computer system modeling in which the computer system is represented as a network of queues which is evaluated



analytically (Graham, 1984). A network of queues is a collection of service centers, which represent system resources, and customers, which represent users or transactions. The most common application of queuing network modeling involves projecting the effect on performance of changes to the configuration or workload of an existing system. Based on this modeling method, an approach is developed to analyze the effect of configuration changes on the system.

A single service model is established in this study. The service represents the system resource (CPU and memory), and the customer represents the transactions processed in LabVIEW application. The existing system is the system running a LabVIEW control application without a data collector. The modified system is the system running a LabVIEW control application with a data collector integrated. A statistical technique is required to provide answers to the question – is there a significant difference on the CPU usage and the number of threads between the existing system and the modified system. If the significant difference exists between the evaluation means, the execution of the data collector does have a significant effect on the existing system. Therefore, the development of the module does not meet the requirement of the study. If there is no significant difference, the execution of the data collector does not increase the system processing load significantly. Therefore, the data collector is verified to be an efficient module for the integration of data collection processes.

#### *Statistical Technique Used*

The statistical technique used in this study is one-way analysis of variance (ANOVA). The variance of a population of values is computed as:

$$\sigma^2 = \frac{\sum (x_i - \mu)^2}{N}$$

$\mu$  = the population mean

$N$  = the population size

The unbiased sample estimate of the population variance is computed as:

$$s^2 = \frac{\sum (x_i - \bar{x})^2}{n - 1}$$

$\bar{x}$  = the sample mean

$n$  = the sample size

In general, the purpose of ANOVA is to test for significant differences between means. ANOVA is derived from the fact that in order to test for statistical significance between means, variances are actually compared. At the heart of ANOVA is the fact that variances can be partitioned. The variance is computed as the sum of squared deviations from the overall mean, divided by  $n-1$  (sample size minus one). Thus, given a certain  $n$ , the variance is a function of the sums of (deviation) squares, or SS. The total SS can be partitioned into the SS due to within-group variability and variability due to differences between means. The within-group variability (SS) is usually referred to as Error variance. This term denotes the fact that it can not be readily explained or accounted for in the current design. However, the SS Effect can be explained. Namely, it is due to the differences in means between the groups. Put another way, group membership explains this variability because it is due to the differences in means.

In conclusion, the purpose of analysis of variance is to test differences in means for statistical significance. This is accomplished by analyzing the variance, that is, by

partitioning the total variance into the component that is due to true random error and the components that are due to differences between means. These latter variance components are then tested for statistical significance. If it is significant, the null hypothesis of no differences between means will be rejected, and the alternative hypothesis that the means are different from each other will be accepted.

### *Hypotheses*

$$H_0 : s_1^2 = s_2^2$$

Null Hypothesis 1: Mean difference between the CPU Usages on the existing system and the modified system is equal to zero. Any observed differences can be attributed to chance (sampling error) alone.

Null Hypothesis 2: Mean difference between the number of threads on the existing system and the modified system is equal to zero. Any observed differences can be attributed to chance (sampling error) alone.

$$H_A : s_1^2 \neq s_2^2$$

Alternative Hypothesis 1: The CPU Usage means on the existing system and the modified system are significantly different. The observed differences can not be attributed to chance (sampling error) alone.

Alternative Hypothesis 2: The means of thread numbers on the existing system and the modified system are significantly different. The observed differences can not be attributed to chance (sampling error) alone.

### *Assumptions and Limitations*

For the purpose of this study, it is assumed that the LabVIEW application and the web-based control integration system are the only applications running in the control server while testing. These remote control applications include LabVIEW VIs, motion control application, MAX (Measurement and Automation) application, and integrated web applications which are necessary for remote control operation. In fact, there are some other computer applications running in the server for supporting the Windows operating system, such as virus detecting. Comparing with the remote control applications and the performance management system, these computer applications occupy a very small percentage of server resources. Therefore, in this study, the resources used by these computer applications are ignored.

Also for this statistical study, it is assumed that samples are randomly selected from the population, the dependent variable is a continuous variable measured at the interval or ratio level, and there are more than one categorical-level independent variables. Data collected from each of the various sample groups in this study are assumed to be approximately normally distributed and have approximately the same variance. When sample sizes are large (i.e.,  $> 25$  measurements), this test is quite robust to violations of the assumptions of normalcy and homogeneity of variance (Diekhooft, 1992).

### *The Experimental Design*

This statistical study is based on a single service model established by applying the queuing network modeling method to test the effect of a system modification. The service represents the system resource (CPU and memory), and the customer represents the transactions processed in LabVIEW application. The existing system is the system

running a LabVIEW control application without a data collector. The modified system is the system running a LabVIEW control application with a data collector integrated. The performance of the existing system and the modified system is analyzed by comparing the means of their CPU usages and the numbers of threads sampled randomly during the operation. The significance of the difference is tested by using the one-way ANOVA method.

For data sampling, the existing system and the modified system are actually the same control server running the same LabVIEW control application. The network connection, the physical system setup, and the software environment are identical. Therefore, differences contributed by factors other than the data collector are eliminated.

#### *Set the Type I Error Rate*

The type I error rate for this study is set to 0.05.  $\alpha = 0.05$ . The  $\alpha$  determines the significance level used for rejecting the null hypothesis. If the probability is less than or equal to the significance level, then the null hypothesis is rejected and the outcome is said to be statistically significant. Traditionally, researchers have used either the .05 level (sometimes called the 5% level) or the .01 level (1% level), although the choice of levels is largely subjective. The lower the significance level, the more the data must diverge from the null hypothesis to be significant. Therefore, the .01 level is more conservative than the .05 level. A level of significance of 5% is the rate that we will declare results to be significant when there are no relationships in the population. If a lower level is set, we may take the risk of the type II error which means a false null hypothesis can fail to be rejected. Therefore, we may use the performance measurement software for further study

in false while actually the overhead caused by running the software has no relationship with the server's other performance data.

In a Type I error, a conclusion is drawn that the null hypothesis is false when, in fact, it is true. Therefore, Type I errors are generally considered more serious than Type II errors. The probability of a Type I error ( $\alpha$ ) is called the significance level and is set by the researcher. There is a tradeoff between Type I and Type II errors. The more an experimenter protects himself or herself against Type I errors by choosing a low level, the greater the chance of a Type II error. Requiring very strong evidence to reject the null hypothesis makes it very unlikely that a true null hypothesis will be rejected. However, it increases the chance that a false null hypothesis will not be rejected, thus lowering power. In this study, the Type I error rate is set to 0.05 for the purpose of our research.

#### *The Population and Sample*

In this study, the population is all the system performance measurements including CPU usage and the number of threads that can be measured by the Windows Task Manager on the existing system and the modified system. From the population, 40 pairs of data will be sample randomly. Twenty pairs of CPU usage (percentage data) will be sampled from the existing system and the modified system. Twenty pairs of the thread number will be sampled from the existing system and the modified system. The Task Manager window used for data sampling is shown in Figure 38.

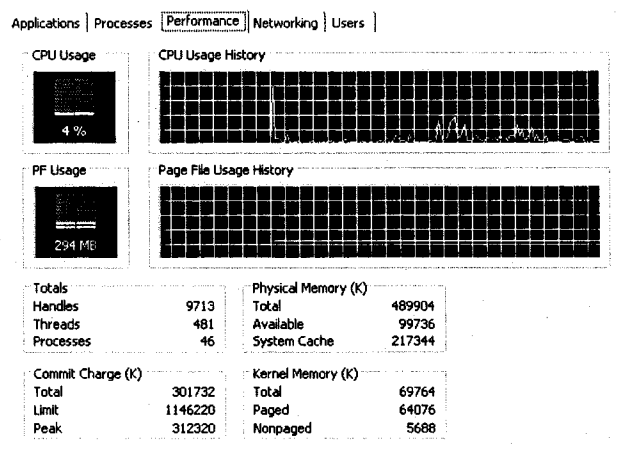


Figure 38. The Task Manager Window

In this study, the data are collected from the Windows system tool, Windows Task Manager. The data needs to be read randomly during the system operation. The entire data sampling process is arranged in 10 days at different time schedule, so that the data in various networking usage situations are included into the samples. In order to maintain the consistence of operating environments between the existing system and the modified system, the data samplings from two systems are paired together. One data sampling from the existing system is closely followed by one data sampling from the modified system. Therefore, the variance caused by chance errors is eliminated.

### *Descriptive Statistics*

Descriptive statistics of the dependent variables are analyzed to examine their distributions. The dependent variables are categorized into two groups, one for the existing system and one for the modified system. The CPU Usage is analyzed for each group and then for overall variables from both groups. The same analysis has been done on the variable of number of threads.

Table 7.

## Descriptive Statistics of CPU Usages in Each System

	N	Mean	Std.	Variance	Skewness		Kurtosis	
		Statistic	Statistic	Statistic	Statistic	Std. Error	Statistic	Std. Error
CPU1	20	13.60	5.87949	34.568	-.385	.512	-1.033	.992
CPU2	20	15.15	6.62749	43.924	-.520	.512	-.905	.992
Valid N (listwise)	20							

As shown in Table 7, the CPU Usages on the existing system have a mean of 13.6% and a standard deviation of 0.0587949%. The distribution of the dependent variable is a nearly normal distribution. Its skewness value of -.385 indicates the distribution of the dependent variable is slightly left skewed, which means the left tail is heavier than the right tail. Its kurtosis value of -1.033 indicates that the distribution of the dependent variable is slightly flat. The CPU Usages sampled from the modified system have a mean of 15.15% and a standard deviation of 0.062749%. The distribution of the dependent variable is also a nearly normal distribution. Its skewness value of -.520 indicates that the distribution of the variable analyzed is slightly left skewed, which means the left tail is heavier than the right tail. Its kurtosis value of -.905 indicates that the distribution of the variable is slightly flat. The Histogram Graphs of these two distributions are shown below.



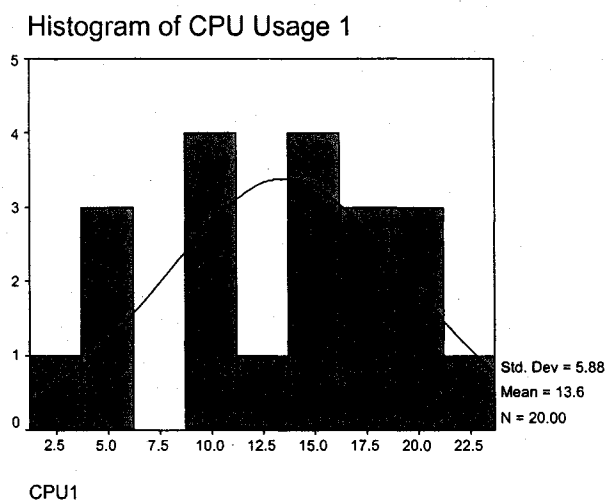


Figure 39. Histogram of CPU Usages on the existing system

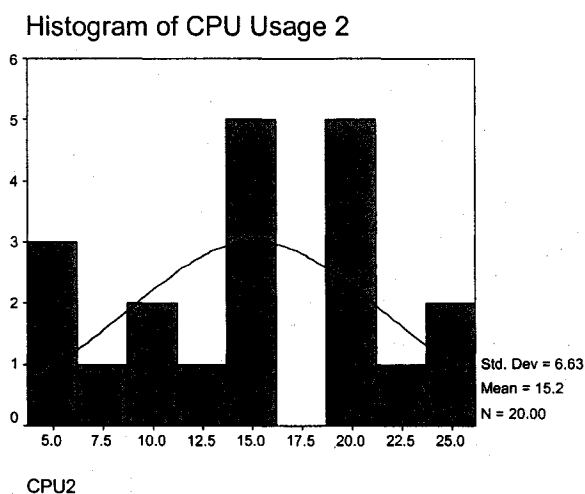


Figure 40. Histogram of CPU Usages on the modified system

The descriptive statistics of CPU Usages from both systems is analyzed as shown in Table 8. The Histogram Graph displaying the distribution of the variables is also shown below.

Table 8.

## Descriptive Statistics of CPU Usages in both Systems

	N	Mean	Std.	Variance	Skewness		Kurtosis	
		Statistic	Statistic	Statistic	Statistic	Std. Error	Statistic	Std. Error
CPU Valid N (Listwise)	40 40	14.375	6.23344	38.856	-.391	.374	-1.000	.733

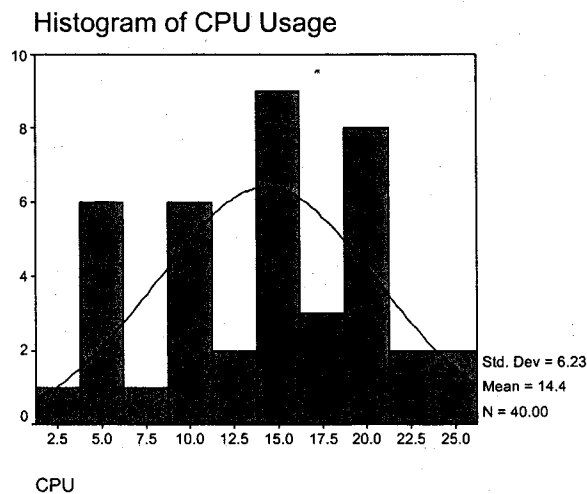


Figure 41. Histogram of CPU Usages on both systems

As shown in Table 8, the CPU Usages on both the existing system and the modified system have a mean of 13.6% and a standard deviation of 0.0587949%. The distribution of the dependent variable is a nearly normal distribution. Its skewness value of -.391 indicates the distribution of the dependent variable is slightly left skewed, which means the left tail is heavier than the right tail. Its kurtosis value of -1.000 indicates that the distribution of the dependent variable is close to a perfect normal distribution.

Table 9.

## Descriptive Statistics of the Number of Threads in Each System

	N	Mean	Std.	Variance	Skewness		Kurtosis	
		Statistic	Statistic	Statistic	Statistic	Std. Error	Statistic	Std. Error
THREADS1	20	465.75	12.34962	152.513	-1.970	.512	3.895	.992
THREADS2	20	467.90	9.42505	88.832	-.230	.512	-1.039	.992
Valid N (Listwise)	20							

Table 9 displays the descriptive statistic result for the number of threads from each system. The numbers of threads sampled from the existing system have a mean of 465.75 and a standard deviation of 12.34962. The distribution of the dependent variable is a nearly normal distribution. Its skewness value of -1.97 indicates the distribution of the dependent variable is left skewed, which means the left tail is heavier than the right tail. Its kurtosis value of 3.895 indicates that the distribution of the dependent variable is peaked. The numbers of threads sampled from the modified system have a mean of 467.90 and a standard deviation of 9.42505. The distribution of the dependent variable is also a nearly normal distribution. Its skewness value of -.230 indicates that the distribution of the variable analyzed is slightly left skewed, which means the left tail is heavier than the right tail. Its kurtosis value of -1.039 indicates that the distribution of the variable is slightly flat. The Histogram Graphs of these two distributions are shown below.

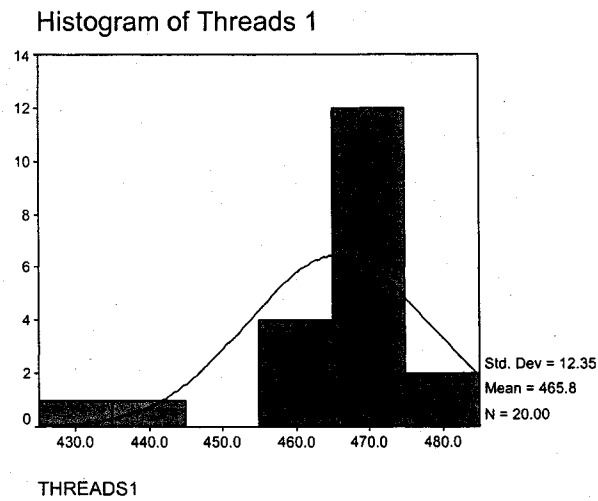


Figure 42. Histogram of thread numbers on the existing system

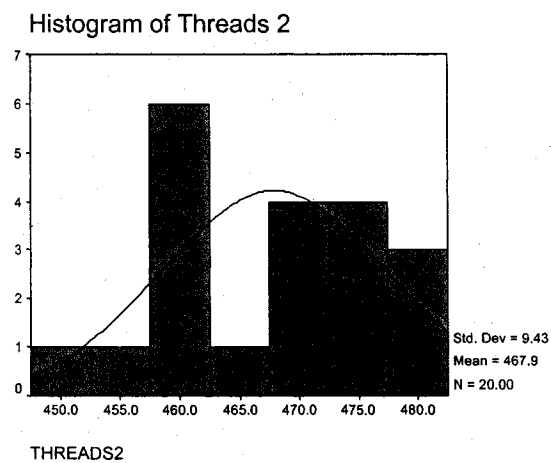


Figure 43. Histogram of thread numbers on the modified system

The descriptive statistics of thread numbers from both systems is analyzed as shown in Table 10. The Histogram Graph displaying the distribution of the variables is also shown below.

Table 10.

## Descriptive Statistics of the Number of Threads from Both Systems

	N	Mean	Std.	Variance	Skewness		Kurtosis	
		Statistic	Statistic	Statistic	Statistic	Std. Error	Statistic	Std. Error
THREADS1 Valid N (Listwise)	40	466.825	10.89786	118.763	-1.473	.374	3.026	.733

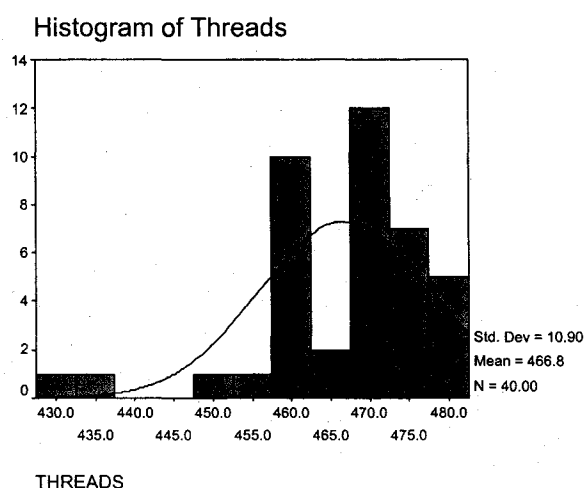


Figure 44. Histogram of the thread numbers on both systems

As shown in Table 10, the numbers of threads on both the existing system and the modified system have a mean of 466.825 and a standard deviation of 10.89786. The distribution of the dependent variable is a nearly normal distribution. Its skewness value of -1.473 indicates the distribution of the dependent variable is left skewed, which means the left tail is heavier than the right tail. Its kurtosis value of 3.026 indicates that the distribution of the dependent variable is very flat.

In conclusion, from the descriptive statistics above, it can be concluded that the sampled data of CPU Usage and the number of threads from the existing system and the modified system are acceptable for ANOVA analysis.

*Output Tables of the Statistical Test*

Table 11.

Descriptive Statistics of CPU Usage for ANOVA Test

	N	Mean	Std.	Std. Error	95% Confidence Interval	
					Lower	Upper
1	20	13.600	5.87949	1.31469	10.8483	16.3517
2	20	15.150	6.62749	1.48195	12.0482	18.2518
Total	40	14.375	6.23344	.98559	12.3814	16.3686

Table 12.

Homogeneity Test on CPU Usage Variable

Levene Statistic	df1	df2	Sig.
.127	1	38	.724

Table 11 displays the means and standard deviations of the evaluation values of each group that is calculated by SPSS software. The CPU Usage mean of the modified system (group 2) is 15.15%, which is slightly higher than the CPU Usage mean of the existing system 13.60%. The standard deviations of the two groups are quite close. Table 12 is the Levene's test for equality of error variances. This test examines for possible violations in

the assumption of Homogeneity of Variance among groups. As shown in the table, the degree of freedom 1 is 1 and the degree of freedom 2 is 38. The critical F value for this test is 4.098 (checked from F Distribution table). The Levene statistic value from this test is 0.127 which is less than the critical F value. Therefore, it can be concluded that this test is not significant, and there is no difference in the variances between each group.

Table 13.

ANOVA Test on CPU Usage Variable

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	24.025	1	24.025	.612	.439
Within Groups	1491.350	38	39.246		
Total	1515.375	39			

Table 13 is the ANOVA test results calculated by SPSS software. The total variability of the CPU Usage variable is 1515.375. It is partitioned into the SS due to within-group variability and variability due to differences between means. In this test, the SS due to within-group variability is 1491.35, 98.4% of the total SS. And the variability due to differences between means is 24.025, only 1.6% of the total SS. Therefore, the variability due to differences between means is not significant. This analysis is also verified by the F test. By using  $df_1=1$ ,  $df_2=38$ , the critical F value of this test is 4.098. The F value in this ANOVA test is 0.612, which is less than the critical F value. The test is not statistically significant. It can be concluded that the null hypothesis about the CPU Usage variable is accepted.

Table 14.

## Descriptive Statistics of Thread Numbers for ANOVA Test

	N	Mean	Std.	Std. Error	95% Confidence Interval	
					Lower	Upper
1	20	465.750	12.34962	2.76146	459.9702	471.5298
2	20	467.900	9.42505	2.10751	463.4889	472.3111
Total	40	466.825	10.89786	1.72310	463.3397	470.3103

Table 15.

## Homogeneity Test on Thread Numbers Variable

Levene Statistic	df1	df2	Sig.
.041	1	38	.841

Table 14 displays the means and standard deviations of the evaluation values of each group that are calculated by SSPS software. The thread numbers mean of the modified system (group 2) is 467.9 which is slightly higher than the thread number mean of the existing system 465.75. The standard deviations of the two groups are 12.34962 and 9.42505 respectively. Table 15 is the Levene's test for equality of error variances. As shown in the table, the degree of freedom 1 is 1 and the degree of freedom 2 is 38. The critical F value for this test is 4.098. The Levene statistic value from this test is 0.041 which is less than the critical F value. Therefore, it can be concluded that this test is not significant, and there is no difference in the variances between each group.



Table 16.

## ANOVA Test on Thread Number Variable

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	46.225	1	46.225	.383	.540
Within Groups	4585.550	38	120.672		
Total	4631.775	39			

Table 16 is the ANOVA test results on the number of threads calculated by SPSS software. The total variability of the thread number variable is 4631.775. It is partitioned into the SS due to within-group variability and variability due to differences between means. In this test, the SS due to within-group variability is 4585.55, 99% of the total SS. And the variability due to differences between means is 46.225, only 1% of the total SS. Therefore, the variability due to differences between means is not significant. This is also verified by the F test. By using  $df_1=1$ ,  $df_2=38$ , the critical F value of this test is 4.098. The F value in this ANOVA test is 0.383, which is less than the critical F value. The test is not statistically significant. It can be concluded that the null hypothesis about the number of threads is accepted. The evaluation means in the existing system and the modified system are equal.

*Conclusion*

As the one-way ANOVA test was conducted based on 40 sampled CPU Usage variables and 40 sampled thread numbers, it can be concluded that the evaluation CPU Usage means from the existing system and the modified system are equal. It can also be concluded that the evaluation means of Thread Numbers from the existing system and the

modified system are equal. Any observed differences can be attributed to chance (sampling error) alone. It is verified that in the single service model, the system modification does not have significant effect on the existing system.

## Chapter 4

### FINDINGS AND RECOMMENDATIONS

#### Introduction

In this study, a modular integration system is developed for implementing enterprise-control integration with an emphasis on dealing with manufacturing process data. It is proved in the study that small-sized modular application can be applied in the LabVIEW-based process to implement manufacturing data integration instead of installing third-party software. An experimental template of the structured manufacturing information system with the focus on relational databases is implemented on a laboratory control process. The findings about this modular integration system, its appropriate application in manufacturing industry, and personnel involvement in the application will be discussed in this chapter. Recommendations about the possible development and enhancement on this template system to achieve better manufacturing system integration are included in the following session. From the view of a researcher, recommendations are made on future studies and researches on the issues of manufacturing system integration focusing on databases based on this study.

#### Modular and Structured System

##### *The Preference on Modular and Structured System*

The current problems faced by manufacturing enterprise integration include the lack of business justification, the plethora of conflicted solutions and terminology, and an

insufficient understanding of the cross-domain technology by the end users. This especially inhibits, or at least delays, the use of relevant methods and tools in small-to-medium-sized manufacturing enterprises. The development of a modular and structured integration system aims at these problems by providing simplicity, flexibility, and efficiency to integration solutions with less investment.

As demonstrated in previous chapters, the enterprise-control system integration is the field in which the most complicated and confusing problems exist. This is caused by the variety of manufacturing control devices and the various formats of manufacturing data. This study is focused on the process of dealing with manufacturing information data based on LabVIEW related control processes. The system developed in this study eliminates the need for relying on third-party software to provide the communication channel between the LabVIEW controller and the SQL relational database. Instead, a modular LabVIEW-based data collector is integrated into the LabVIEW control application. The modular data collector resides in the LabVIEW environment as programmable subVIs. The subVIs can be accessed and used by any LabVIEW applications when they are saved in the LabVIEW Dynamic Link Library (DLL). The data collector can be used as a component of modular integration systems in any LabVIEW-based manufacturing process.

First, the data collector is small in size which occupies less than 300 kilobytes memory. Compared with the size of any software supporting the database communication with Megabytes, it occupies very small memory in the control server. When the data collector is running inside the LabVIEW application, the machine is not in the multi-tasking status with several applications running together. All the data processing

and function realization is executed within a single control application. In the last section of Chapter Three, the ANOVA test results show that the data collector does not make significant change on the existing computer system. Second, the data collector provides an economic way to implement integration solutions. This is also a critical factor when small-to-medium-sized manufacturing companies consider their investment budget on enterprise integration implementation. The same data collector may not adapt to various cases in real industry world. However, the theories and principles applied in the development process can be used by manufacturing engineers and IT engineers to develop similar modular application in various control processes. For example, the flexible usage of ADO objects, the combination of .NET functions and ActiveX function, and the encapsulation of SQL statements, can be applied in the communication of control applications with relational databases, not limited to LabVIEW-based applications.

Last, the whole integration system is simplified by adopting modular components. It is easier to update and maintain the operation of the system, and deal with the product obsolescence. ActiveX objects, ADO objects, and SQL statements are basic programming components for any web-based applications. The revision of the web-based interface, the update of the relational database, or the change of the manufacturing control process can be achieved by modification of each module.

#### *The Template of a Structured System*

The study provides a basic template for a manufacturing information integration system. Based on this study, the primary design criteria for a manufacturing data integration project include documented code, plant maintainability, open database

connectivity, and flexibility. The template system has three essential components, a process data collector, a database server, and a web-based interface or reporting system.

#### *Process data collector*

The collector is the data acquisition point for the manufacturing process data. Its essential function is to collect data from plant-floor controllers, store data in a "lite" version of the manufacturing database, and provide transaction management of records to the process data server.

The manufacturing database in the collector will store real-time data values in the same table structure and data formats as the process data server. The server that hosts the data collector can be an industrial computer located in proximity to the manufacturing process. By using a Web server, such as Microsoft IIS (Internet Information Services) Server, to provide diagnostics and essential reports, the control engineer can ensure data integrity between the process data server and the collector.

When the network connection between the data collector and data server is lost or blocked, data transactions are stored on the data collector and rolled forward to the data server once the connection is re-established.

#### *Process Data Server*

The process data server is a database server and an Internet/intranet Web server. The process data server can be located in the front office computer room or offsite via secure Internet. Memory and storage requirements for most manufacturing facilities are a nominal 512 MB and 40 GB of storage. The system archives data older than six months. Plant-floor controllers automatically feed data to a manufacturing database through the data collector. Manufacturing database design can provide a marketplace edge if the

information is used to advance business goals. To successfully design the database, one needs to understand table definition—where database views are needed and how to write stored procedures.

#### *Web-based Interface or Reporting*

End-users of manufacturing information are typically plant managers, quality control staff, and production supervisors. The manufacturing information normally needed by them are quality data, downtime reports, work order system, ingredient usage, and production reports. Reports are provided via an Internet browser. When the plant manager points the Web browser at the process data server, an information interface becomes available. The report can be thought of as a "process database record," providing an information snapshot for a selected set of parameters, such as shift, batch, or work order. The Web-based server runs active server pages, with Visual Basic scripts often triggering stored procedures in the database to bring up the desired information. Web-based reporting, established in this way, requires no applications to be loaded on each desktop and information is accessible over an intranet or secure Internet connection.

#### *The Personnel Involvement*

The design, development, and maintenance of the integration system require the involvement of personnel from several functional departments with cross-domain knowledge and experiences, typically the IT staff, the manufacturing engineer, and the control engineer.

The data collector resides on the plant floor and is therefore typically the responsibility of the control engineer. Control engineers should have an essential role in determining minimum essential criteria for speed and quantity of data. In a

homogenization process, for instance, the record for the process stored in the manufacturing database need only have a high temperature, low temperature, and an average temperature logged for a certain time interval. The control engineer should write PLC code or LabVIEW code to provide these parameters to the process data collector. This is not enough for developing and maintaining the data collector. It is required that either the control engineers have the knowledge of information systems or IT staff needs to help to work through the process.

The control engineer can take responsibility for data in the collector and the database administrator can take responsibility for data in the process data server. In this way, the manufacturing database becomes common ground for IT and control engineers to support the needs for plant information. To support the needs for production and process information and succeed in the design/building of a manufacturing database requires a level of communication and cooperation between the IT department and control engineers not commonly exercised. The reason is that computer and software skills needed to set up a manufacturing database are the same basic skills that a database administrator would possess to design and build a financial, inventory, or contact database. When comparing the development of a manufacturing database to other types of database projects, the differences begin with conception and design. It is here the input of the control engineer and the manufacturing engineer is needed to identify the origin of the data and format it into proper engineering units or production metrics so that it can be understood by production management.

The design and construction of the web-based interface or other web-based reporting tools require the collaboration of the IT staff and the manufacturing engineer. The IT staff



is responsible for web-based programming and coding. However, the manufacturing engineer and manufacturing management need to provide the input for the format of reports and web-based interface, the way or method used to display or report the manufacturing data, the priority of the data, any necessary alarm report, or any graphic tools needed in the reports.

#### The Potential Development and Enhancement of the System

To design and construct a complete enterprise control integration system for implementation in a manufacturing enterprise is far beyond the time and efforts invested in a study for dissertation. Therefore, further development and enhancement could be added to the system developed in this study. Here, based on the experience and knowledge gained from this study, several recommendations are proposed.

In this study, there is only one manufacturing database established. The data retrieved by the data collector is fed directly into this database. As stated in the previous session, a more feasible way is to set up two manufacturing databases. A smaller one is hosted by a machine with the plant-floor process, which can be used to for temporary data storage. For example, data collected in a short period of time can be saved in this database. The separation of the database can improve the performance of the system in several aspects. First, the temporary data storage can provide a direct data access to plant-floor workers, supervisors and engineers. They do not need to retrieve data from the main database server when needed. Second, the data updating rate from the small database to the main database can be set to a lower rate. In this situation, the network traffic can be controlled to an optimized level. Also, the separated database server provides a data backup for each other.

Another recommendation is about the enhancement on the web-based interface. In a manufacturing enterprise, personnel from several functional departments or different management levels require different manufacturing information from the plant floor. For example, the supply manager concerns about the raw material usage while maintenance personnel needs to know the machine's current status. Therefore, different web-based interfaces or web reporting formats can be developed to get the right manufacturing information to the right person for correct and fast decision making.

Improvement can also be made on the security mechanism implemented in the system. As demonstrated in Chapter Three, Forms Authentication is applied to protect the access to the web-based interface. However, the credential information, valid usernames and passwords, are saved in the Web.config file which is a component of the web application. This is not the best way to keep the credential information, especially when the users may gain the direct control right over the plant-floor processes. It is recommended to save the credential information in a data table hosted by a secure database server. Events for connecting to the database and retrieving the information for authentication purposes need to be programmed into the Web.config file and the login.aspx file. When implemented on plant floor, the system can also apply secure Internet protocols to protect the data transmission. Typical secure Internet protocols are TSL, Transport secure layer, and SSL, Secure Sockets Layer. Typical networking topology on the shop floor could be organized as isolated LAN; hence the data transmitted between nodes are confined within the network. In the case some nodes need to access outside of the shop floor, they can be identified and communication with outside through a firewall.

At the last, the database server can be connected to higher level manufacturing management and business applications to support decision making on the top levels of the manufacturing enterprise. Since the database is compatible with most of the Microsoft office applications, it is very convenient to realize the integration.

#### Recommendations for Future Studies

The engineering- and business-driven need for manufacturing process data has led to the development of manufacturing information systems, with the focus on the relational database. This study has proposed an approach for developing a modular integration system to deal with manufacturing data process. This approach has technically improved the flexibility and efficiency of the communication between plant-floor control process and the database server, with reduced cost. However, further studies are recommended to get the statistic data about the implementation of enterprise-control integration solutions in small-to-medium-sized manufacturing companies, compare the efficiencies between the modular system and a conventional system, and to evaluate the feasibility of the modular integration system.

Collecting and analyzing real-time data from the plant floor plays a critical role in meeting the market's demand for consistent product quality and improving time to market for a manufacturer's products. However, for quite a few small-to-medium-sized manufacturing companies, complete implementation of the manufacturing information system is still above their short-term plan, or even long-term plan. For they still have stand-alone workstations running on the plant floor as isolated information islands and old version PLC controllers without advanced data integration module, the implementation becomes even more expensive.

In order to solve the problem, the first step is to describe the problem in a realistic way. A survey is recommended to be conducted on manufacturing companies for their current situation of manufacturing information systems. The criteria for selected manufacturing companies should include but not be limited to: (1) small-to-medium-sized; (2) produce different types of products; (3) have different production types; and (4) located in different geographic areas. Items listed in the survey questionnaires may include but not be limited to: (1) current facilities used in manufacturing data processing including both hardware equipment and software applications; (2) current integration levels; (3) short-term strategic plan in improving the manufacturing information system; (4) long-term strategic plan in improving the manufacturing information system; and (5) personnel training plans and involvement. Based on the result of the survey, statistic analysis should be taken to figure out the current situation of the manufacturing information systems in these companies and also problems.

After the survey is conducted, establish possible collaboration relationship with one or more companies. With contributions from experienced control engineers and manufacturing engineers in the companies, an experimental study needs to be conducted to compare the efficiencies between the modular integration system and a conventional system for data collection. Both the modular integration system, the system with the modular data collector, and a conventional system, a system with third-party software for data collection, are implemented into the same control process in the real-world manufacturing environment. Measurements including database response time, system resource usage, and database updating cycle, are sampled from both systems during the

operation periods. Specified software or hardware may be required for the accurate data sampling. ANOVA or t-test can be applied to compare the statistical significance between the modular integration system and the conventional system. Therefore, the efficiency improvement of the modular data collector can be further verified and tested in the real-world manufacturing environment.

Technical enhancements need to be applied to the modular integration system. This step is also needed to be conducted with experienced control engineers, manufacturing engineers and IT staff from real world environments that are willing to support the project. The enhanced system can be implemented to improve the manufacturing information system in the company. Tests need to be conducted to evaluate the system performance based on measurements from the real control process on the plant floor.

In conclusion, to improve the implementation of manufacturing system integration for business-driven and engineering-driven purposes requires the efforts from cross-domain personnel in both academia and industry.

## REFERENCES

- Beyon, Y. J. (2001). LabVIEW Programming, Data Acquisition and Analysis. Prentice Hall PTR, Upper Saddle River, New Jersey.
- Chang, T. C., Wysk, R., & Wang, H. P. (2005). Computer-Aided Manufacturing. Third Edition. Prentice Hall, Upper Saddle River, NJ 07458.
- Dewar, I. (1999). Real-time optimization equals on-line performance improvements and off-line benefits. ISA TECH 1999.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Design Patterns: Elements of reusable object-oriented software. Addison-Wesley.
- Francois, V. (1996), Enterprise Modeling and Integration: Principles and Applications, Chapman & Hall.
- Foggon, D., & Maharry, D. (2004). Beginning ASP.NET 1.1 Databases. Apress.
- Fuggetta, A., Picco, G. P., & Vigna, G. (1998). Understanding code mobility. IEEE Trans. Softw. Eng., 24, 342-361.
- Gavalas, D., Ghanbari, M., O'Mahony, M., & Greenwood, D. (2000). Enabling mobile agent technology for intelligent bulk management data filtering. In Proc. Of NOMS'00, Honolulu, HI, April 11-13, pp. 865-876. IEEE Press, Piscataway, NJ.
- Getting Started With NI Motion Control (2003). National Instrument.
- Grundgeiger, D. (2001). Programming Visual Basic .NET. O'Reilly.

- Gunasekaran, A. (2001). Agile Manufacturing: the 21st Century Competitive Strategy. Elsevier.
- Herb, M. S. (1997). Considerations for integrating process with business plan in the real world. ISA TECH 1997.
- Hua, J., & Ganz, A. (2003). A New Model for Remote Laboratory Education Based on Next Generation Interactive Technologies. Frontiers in Education Conference.
- Hoover, S. (1999). Quick product-data management. Think out of the box when implementing PDM. Quality, Vol38, Iss9.
- Manufacturing Engineering Laboratory (2001). Issues in enterprise integration. retrieved from MEL website January 23, 2005. <http://mel.nist.gov>
- Mick, R. (2003). Building Agility into the Manufacturing Value Chain. ARC White Paper. ARC Advisory Group.
- Motion Control. NI-Motion User Manual. (2003). National Instrument.
- Schneider, S., Pardo-Castellote, G. & Hamilton, M. (2000). Using Ethernet for real-time communication. ISA EXPO2000.
- Sowell, T. (1999). Completing a manufacturing system by framing time series data in manufacturing context. ISA TECH 1999.
- Taqui, A. (2002). Increase ROI by providing real-time manufacturing process information. IMS 2002.
- Tham, K. D. (1997), Enterprise Modeling, Enterprise Integration Laboratory. Retrieved March 3<sup>rd</sup>, 2005, from <http://www.ie.utoronto.ca/EIL/comsen.html>