Final Report

Investigating the Stability and Efficiency of ADO Modules in a Web-based Manufacturing Control Interface

Yuqiu You, Ph.D.
yu.you@morehead-st.edu
Department of Industrial & Engineering Technology
Morehead State University

I. Introduction

In today's new manufacturing environment, manufacturing enterprises are facing rapidly changing situations. To be competitive, enterprises must adapt to changes and evolve to be reactive so that changes become natural dynamic states rather than something forced onto the enterprise [2]. This evolution requirement necessitates the need for enterprise integration with an increasing emphasis on agility. The Manufacturing Engineering Laboratory (MEL) of the National Institute of Standards and Technology (NIST), defines 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 [1]. The purpose of enterprise integration 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 [3]. 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 [4].

The management of complex value chains in manufacturing enterprises requires increased integration of disparate plant control systems and other computerized enterprise processes [5]. 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 [6].

The purpose of this study is to provide a modular and economic solution in implementing manufacturing enterprise-control integration. To achieve the objective, a web-based module for a LabVIEW control station 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 on enterprise-control system

integration. It is composed of three parts, Active Server Pages (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 dynamic database is to support real-time data management in the system. 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 module provides a template for enterprise-wide web applications to communicate with LabVIEW interfaced control and monitor processes in real time. It is small in scale, but its module function adds flexibility, compatibility, and extendibility for future development.

In order to evaluate the efficiency of the module, a single service model was 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, one-way ANOVA, is applied 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.

II. Methodology

2.1 Overview of the Integration Module

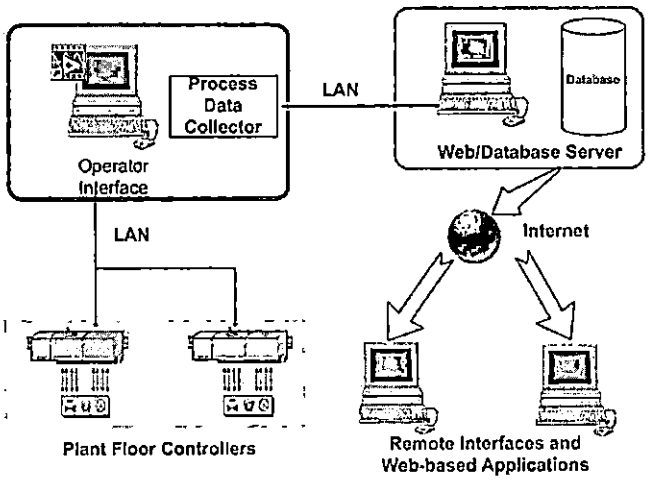

Figure 1: System Architecture

Figure 1 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. The data collector is developed by using LabVIEW controls and Microsoft ® ActiveX ® Data Objects (ADO) and it can access and manipulate data in a 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 [7].

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 a Microsoft Desktop Engine (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. The other one is a virtual web server, which 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, and interacts with clients.

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.

The web-based interface in this integration module was developed in .NET environment. Due to the time and resource limitation in this study, the web-based interface is built only to provide 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. The major features of the module are real-time data retrieving from the database server, data analysis, real-time controlling and monitoring of the plant-floor process, and a live video of the process. The main window of the interface is shown in Figure 2.

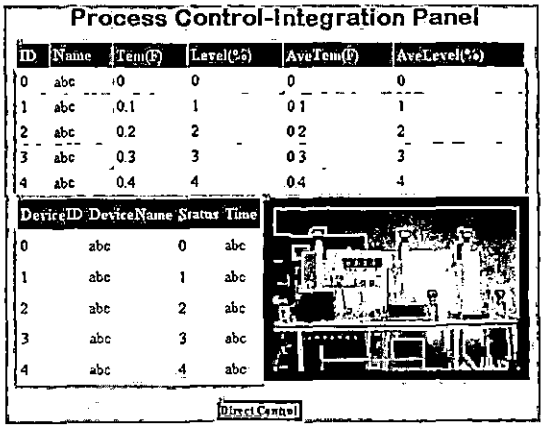

Figure 2: The Main Interface Window

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 time of the server

The main window provides a button from which authorized personnel can navigate to the real-time LabVIEW control panel of the plant-floor process as shown in Figure 3.

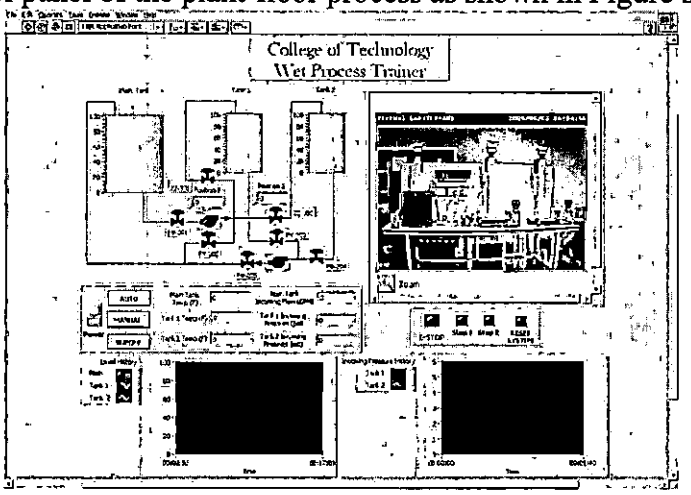

Figure 3: LabVIEW Real-time Control Panel

The control panel above is a virtual instrument programmed in LabVIEW environment. As shown above, a wet process control station with tanks, valves, and sensors is built to simulate an automatic manufacturing process in this study. Controls and indicators on the interface provide a way for users to interact with the control process. A video clip is 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, and incoming flow rates for tanks are also shown on the interface by digital indicators.

2.2 Construction of the Data Collector

The construction of the data collector includes four components: the Open Database Connectivity (ODBC) standard, the structured query language (SQL) for command statements, the OLE DB standard for lower-level database access, and the ActiveX data objects (ADO).

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 Application Programming Interface (API) for ODBC. ODBC then communicates with a database's specific driver that translates the call to the database's low level language. The data collector module is compatible with any database providing an ODBC driver that translates the ODBC calls to the native database language [8]. 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.

The data collector adopts the Structured Query Language (SQL) as command statements for data manipulation in data access. 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. T hey also define and grant access privileges to database users. Data Manipulation Language (DML) statements operate on the data contents of database tables [8]. 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.

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 [9]. 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. OLE DB Service Providers are optional components that implement standard services to extend the functionality of data providers. 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. All data access in the data collector occurs through an OLE DB provider.

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 [10]. The LabVIEW-based data collector consists of ADO objects through invoke and Property Nodes. The object model of ADO in this data collector was made up of three main COM objects, Connection, Command, and Recordset. A Connection object represents a unique session with a data source. 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. Recordset object represents the entire set of records from a base table or the results of an executed command.

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 can be used to create, open, or close a connection with a specified ADO object which is used for a database communication. 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 can be used to initialize a SQL command, set the SQL command statement, configure the SQL command, and execute the SQL command on the specified database. 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.

2.3 The Database Server and the Web-based Interface

The most critical feature of the web-based interface is the real-time communication with the MSDE database server. The web-based interface is developed in ASP.NET environment. ASP.NET provides a platform for web application development. 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. 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 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 4.

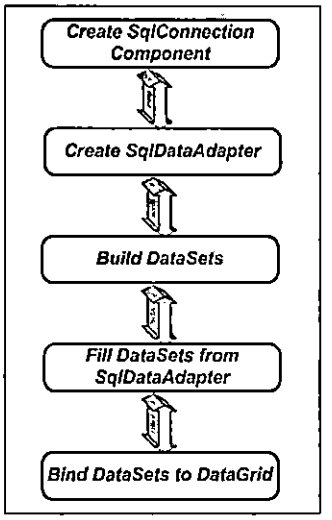

Figure 4: Procedure Logic for DataGrid Control

This procedure logic 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 code begins with the component initialization by defining the class namespace that will 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.

2.4 Testing and Analysis

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.

By applying a queuing network modeling method, a single service model is established. 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 [11]. A statistical technique, one-way ANOVA, is applied to test if there is 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 was verified to be an efficient module for the integration of data collection processes.

In this study, ANOVA test has been conducted on 40 pairs of data sampled randomly – twenty pairs of CPU usage (percentage data) and twenty pairs of the thread number sampled respectively from the existing system and the modified system.

$H_0 : s_1^2 = s_2^2$

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.

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 are sampled randomly. Twenty pairs of CPU usage (percentage data) are sampled from the existing system and the modified system. Twenty pairs of the thread number are sampled from the existing system and

the modified system. The data is read randomly during the system operation. The entire data sampling process is arranged in 10 days at different time schedules, so that the data in various networking usage situations could be 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 could be eliminated.

Table 1. ANOVA Test Results on CPU Usage

ANOVA

CPU

	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 1 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 df1=1, df2=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 2. ANOVA Test Results on the Number of Threads

ANOVA

THREADS

	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 2 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 df1=1, df2=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. T he 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.

As the one-way ANOVA test is conducted based on 40 sampled CPU Usage variables and 40 sampled thread numbers, it could be concluded that the evaluation means from the existing system and the modified system are equal. Any observed differences could 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.

III. Findings and Recommendations

3.1 Findings

As demonstrated above, 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 after 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. The advantages of the data collector are described below.

First, the data collector is small in size, occupying 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. 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 system integration implementation. The same data collector may not adapt to various cases in the 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.

3.2 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 statistical 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. If 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. The follow-up studies will aim at testing the integration module in a manufacturing company with data analysis, and providing a more mature solution for control system integration for LabVIEW-based systems with economic and technical adjustment.

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 the 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, the 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, improving 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.

IV. References

[1] Manufacturing Engineering Laboratory (2001). Issues in enterprise integration. retrieved from MEL website January 23, 2005. http://mel.nist.gov

[2] Chang, T. C., Wysk, R., and Wang, H. P. (2005). Computer-Aided Manufacturing. Third Edition. Prentice Hall, Upper Saddle River, NJ 07458.

[3] Francois, V. (1996), Enterprise Modeling and Integration: Principles and Applications, Chapman & Hall.

[4] Weston, R. H. (1998), "The importance of holistic model driven manufacturing systems" Proceedings of the Institution of Mechanical Engineers, Part B, Journal of Manufacturing Engineering, Vol. 212, No.1, pp. 29-44.

[5] Mick, R. (2003). Building Agility into the Manufacturing Value Chain. ARC White Paper. ARC Advisory Group.

[6] Dewar, I. (1999). Real-time optimization equals on-line performance improvements and off-line benefits. ISA TECH 1999.

[7] Getting Started With NI Motion Control (2003). National Instrument.

[8] Foggon, D., & Maharry, D. (2004). Beginning ASP.NET 1.1 Databases. Apress.

[9] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). Design Patterns: Elements of reusable object-oriented software. Addison-Wesley.

[10] Grundgeiger, D. (2001). Programming Visual Basic .NET. O'Reilly.

[11] Whitten, N. (1995). Managing Software Development Projects. Second Edition. John Wiley & Sons, Inc.

[12] Gavalas, D., Ghanbari, M., O'Mahony, M., and 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.

[13] Gunasekaran, A. (2001). Agile Manufacturing: the 21st Century Competitive Strategy. Elsevier.

[14] Hua, J., & Ganz, A. (2003). A New Model for Remote Laboratory Education Based on Next Generation Interactive Technologies. Frontiers in Education Conference.

[15] Wojcik, M. & Ranganathan, G. (2000). Using Ethernet and web for process monitoring andcontrol. ISA TECH 2000.