**Proceedings of the 2009 DETC**
**2009 ASME International Design Engineering Technical Conferences**
**August 30 – September 2, 2009, San Diego, California, USA**

**DETC2009-86666**

# QUANTIFYING THE PERFORMANCE OF MT-CONNECT IN A DISTRIBUTED MANUFACTURING ENVIRONMENT

**John Michaloski, Byeongeon Lee, and Frederick Proctor**
National Institute of Standards and Technology
Gaithersburg, MD USA

**Sid Venkatesh and Sidney Ly**
Boeing Company
Seattle, WA USA

## ABSTRACT

In the CNC manufacturing world, the continuing pressure to reduce costs and improve time to market places a premium on smarter ways of manufacturing and intensifies the need to integrate feedback from the shop floor into the enterprise business systems. There have been many efforts to improve CNC factory floor integration with varying degrees of success. Recently, MTConnect has been developed as an open and free communications standard to facilitate the exchange of data on the manufacturing floor for machine tools and related devices. This paper will look at quality of service issues concerned with implementing MTConnect in a "Dual-Ethernet" machine tool network configuration.

## NOMENCLATURE

| | |
|---|---|
| API | Application Programming Interface |
| CNC | Computer Numerical Control |
| COM | Microsoft Component Object Model |
| DA | Data Access |
| ERP | Enterprise Resource Planning |
| HTML | Hypertext Markup Language |
| HTTP | Hypertext Transfer Protocol |
| MAP | Manufacturing Automation Protocol |
| MMS | Manufacturing Message Specification |
| MTC | MTConnect |
| OLE | Object Linking and Embedding |
| OMAC | Open Modular Architecture Control |
| OPC | OLE for Process Control |
| PDH | Performance Data Helper |
| QoS | Quality of Service |
| XML | eXtensible Markup Language |
| XSD | XML Schema Definition |

## INTRODUCTION

In today's manufacturing world, "Design Anywhere, Build Anywhere, Support Anywhere" is predicated on universal connectivity and information sharing across all facets of design, manufacturing, distribution, and maintenance. To achieve this ubiquity, integration of all company elements, especially manufacturing, must be done. For years the lack of widely adopted machine-tool integration standards has hindered factory integration, making it difficult and costly.

Historically, CNC integration has been an afterthought to the primary goal, making parts. Indeed, machine tool monitoring remains a largely untapped area for linking real-time plant data into the enterprise. Informative, accurate and timely machine knowledge can be vital to successful manufacturing. Machine monitoring serves as an analysis and diagnostic tool that offers greater visibility into manufacturing processes and help determine ways for improving operation and increasing productivity. The key paybacks of CNC machine monitoring include collection and management of pertinent production information, real-time monitoring and preventive and reactive maintenance, and real-time quality assurance.

Over the years, numerous standards have been promoted to enable machine tool integration. An early attempt at control system integration was the MAP (Manufacturing Automation Protocol) standard, which is a communication standard for intelligent factory floors devices. [1] Another legacy factory floor integration standard is the Manufacturing Message Specification (MMS), which is a messaging system for exchanging data between control applications and networked devices [2;3]. Both MAP and MMS are large and extensive standards, with requirements for supporting several Open Systems Interconnection (OSI) Reference Model layers of communication functionality. These standards enjoyed limited

success as the breadth, complexity and as a consequence, increased cost, resulted in the lack of vendor and user support.

In general, difficulties arise for companies and vendors to adopt standards that are excessively broad and complex, making it costly to develop the required hardware and software. Further, the limited adoption of a standard yields marginal benefits. The latest trend in developing successful manufacturing standards is to leverage widely-adopted, ubiquitous computer technology. In this manner, MTConnect is a new standard sponsored by the Association for Manufacturing Technology developed to facilitate the exchange of data on the manufacturing floor. MTC is an open and royalty free communications standard "built upon the most prevalent standards in the manufacturing and software industry, maximizing the number of tools available for its implementation and providing the highest level of interoperability at a reasonable cost with other standards and tools in these industries." [4]

Excited by the potential of MTC, Boeing worked with NIST and other members of the Open Modular Architecture Controllers (OMAC) User Group, to test MTC on the plant floor. OMAC is an affiliate organization of the Instrumentation, Systems and Automation Society (ISA) working to derive common solutions for technical and non-technical issues in the development, implementation and commercialization of open, modular architecture control. The OMAC Machine Tool Group has been working and promoting the challenging issue of "mainstreaming" machine tool and factory floor integration.

This paper will look at performance and quality of service issues concerned with using MTC in a distributed factory floor environment. Section 2 will give a brief overview of MTConnect technology. Section 3 describes an implementation of Machine Monitoring using MTC that relies on CNC OPC Data Access. Section 4 will analyze various MTConnect issues and then evaluate the quality of service performance. Finally, a discussion including future directions will be given.

## MTC OVERVIEW

Integration of factory floor CNC information has been difficult, if not impossible, as traditionally, factory floor machines have been "islands of automation." So why is machine tool integration such a problem? Although integration standards have and are routinely used elsewhere in manufacturing, acceptance has been slow within the CNC discrete parts industry. The problem stems from the perceived, if not actual, lack of return on investment from integration. It is at times too hard, too costly and/or too complicated. The MTC specification addresses these issues from the onset bringing an approach that ensures affordability as well as performance.

MTC is an open, royalty free, standard that acts as a bridge between all design, manufacturing, distribution, and maintenance data. MTC is based upon prevalent Web technology including XML [5] and HTTP [6]. Using prevailing technology and providing free software development kits minimizes technical and economic barriers to MTC adoption.

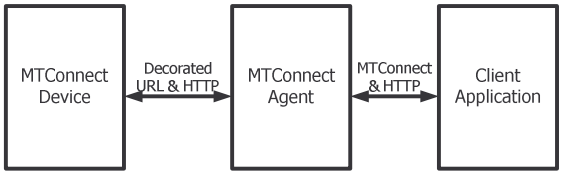


**Figure 1 MTC System Architecture**

Figure 1 shows the high-level system architecture of the MTC standard. An "MTC Device" is a piece of equipment organized as a set of components that provide data. The core of MTC is the "Agent", which is a process that acts as a "bridge" between a factory device and a "Client Application". The MTC Agent receives and stores single or a time series of data samples or events that are made available to the Client Application. An MTC sample is the value of a continuous data item at a point in time. An MTC event describes an asynchronous change in state.

MTC uses HTTP as the underlying communication protocol. HTTP is a standard that serves as the main communication protocol for the Internet and the World Wide Web. Devices that support HTTP on the internet are accessible worldwide. Everybody using the Web is at least vaguely familiar with the HTTP protocol. After you type in http://www.nist.gov and click "GO", HTTP protocol communicates to the NIST Web Server and the HTTP requests receive and display Web pages from the NIST Web Server, most often described by the hypertext markup language (HTML).

Instead of HTML, MTC uses the more flexible Extensible Markup Language (XML) which is used to transport data to and from the MTC Agent (which acts much like a Web Server). XML is extensible, unlike HTML, because you use it to define other languages or information, such as the MTC information model. XML describes information as a series of paired start/end tags, i.e., <DATA> </DATA> and information is contained between the tags, e.g., <DATA> 10.0 </DATA>. XML provides meta-data attributes to the tags such as <DATA id="Xaxis"> 10.0 </DATA>.

An MTC device is modeled in XML as a set of components, and initially the MTC specification is targeted at machine tools and their substituent components – axes, spindle, program, and control sequencing. Because XML is a metalanguage MTC has defined a formal XML Schema Definition (XSD), which describes the allowed structure of the MTConnect XML data.

The MTC Agent is flexible and handles incoming Device stores and outgoing Application requests with the same HTTP mechanism. Detailed communication with the MTC agent leverages the concept of the URL parameter, which has been included within the specification since the earliest HTTP drafts. [6] The URL parameters are described as `"name=value"` pairs appended to the URL query, with multiple URL parameters separated by a '`&`'.

The "Decorated URL" is sent by either the MTC Client Application or the Device to the MTC Agent containing all the


**Figure 2 MTC XML Response to Client Application Status Query**

command and parameters required for the sample request. Below, a Decorated URL `storeSample` command with parameters from a MTC device is shown:

http://pc1.acme.com/storeSample?deviceName=NistTest&dataItemName=Srpm&value=2000

The first part of the Decorated URL indicates what protocol to use (i.e., `http`). The second part specifies the IP address or the domain name (i.e., `pc1.acme.com`) where the MTC agent is located. Next a series of Decorated `name/value` pairs follow, where the `deviceName` is `NistTest` and the `dataItem` is the Spindle RPM or `Srpm`.

As an example of the HTTP/XML communication between the MTC Agent and MTC Client Application, we will assume an MTC Agent is running on our local machine `localhost`. To retrieve current Device status from the MTC agent, we can simply do an HTTP query in Internet Explorer by navigating to the following Web address:

http://localhost/current

This HTTP request will retrieve all the current data and events in the MTC device. Figure 2 shows the MTC XML response as displayed by Internet Explorer.

## FACTORY IMPLEMENTATION

The OMAC MTC deployment was constrained by the security configuration of the targeted factory network. The networking scheme uses a dual Ethernet solution to provide a safety and security buffer for every CNC by isolating it from the main corporate intranet but still allowing TCP/IP connectivity through a front-end PC (FEPC).

A major issue was the lack of native MTC Agent support for some of the targeted CNC machine tools. As an alternative, these CNCs supported OPC, which also allows machine integration. OPC is a more established open specification, for which the authors have previous factory-floor integration experience. [7;8] OPC provides a standard mechanism for communicating to data sources on a factory floor historically targeting process and batch industries, but now with some CNC market presence.

The OPC specification describes a client/server object model to allow communication between client applications (OPC Clients) and control device servers (CNC OPC Server). The OPC-DA specification leverages Microsoft COM [9] to specify OPC COM objects and their interfaces. The control device object and interface are called OPC Servers. Applications, called OPC Clients, can connect to OPC Servers provided by one or more vendors. At runtime, OPC Clients

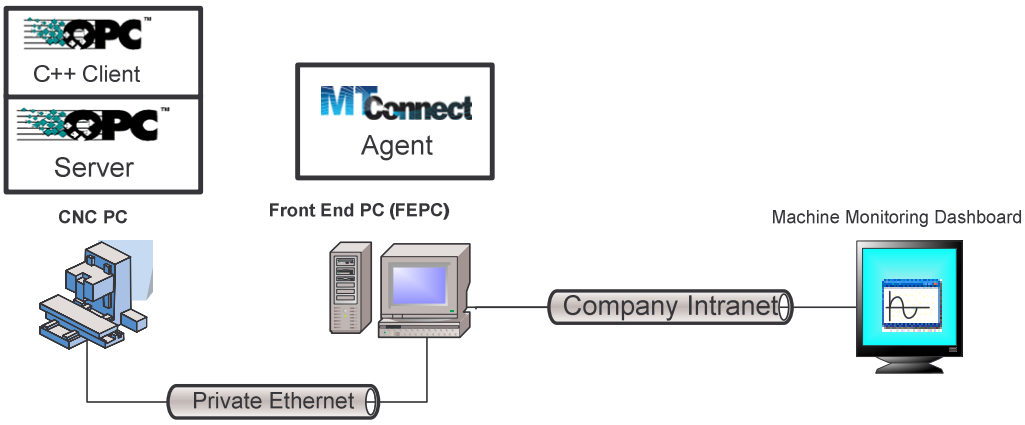


**Figure 3 MTC and OPC Dual-Ethernet Networking Architecture**

connect to an OPC Server, and an OPC Group is created and OPC Items are added for data monitoring and control.

The next design option was whether to tightly integrate the MTC Agent and OPC in the CNC or to use an OPC Client Adapter to communicate to a standalone MTC Agent. MTC provides the Georgia Tech Agent Software Development Kit (SDK) to help implement a turnkey MTConnect agent for the Microsoft .Net software environment. [10] The MTC SDK is flexible and can meet different integration requirements, either embedded or standalone application outside of the device using an MTC Adapter.

The Dual-Ethernet networking scheme led to the decision of using an OMAC MTC Agent to run as a standalone application on the FEPC, so that the CNC would need an Adapter to communicate to it using HTTP. Embedding the MTC Agent on the CNC is possible, but would still require some forwarding agent through the FEPC to allow MTC Clients on the Corporate Intranet. Further, embedding the .Net Framework in the CNC was also not as straightforward.

Figure 3 shows the Dual-Ethernet Network Architecture with the OPC Client/Server running on the CNC PC and the MTC running on Front-End PC to the CNC acting as either a Relay Agent or as the MTC data server. A simple C# .Net windows wrapper program was written around the Georgia Tech MTC Agent core .Net DLL. It runs as a standalone Microsoft Windows Service. A C++ OPC client Windows application was developed that uses COM to communicate to the CNC OPC Server and transmit this data to the MTC Agent using HTTP.

One major architecture design consideration was the decision to avoid DCOM. OPC supports remote Data Access using DCOM, so that the OPC Client could have run on the FEPC. However, this architecture has many disadvantages as OPC and DCOM are beset with security issues, firewall problems, and a multitude of Windows security policies. Further, HTTP is generally not blocked by a firewall, making MTC setup and deployment easier.

## DESIGN AND PERFORMANCE ANALYSIS

A reliable CNC must exhibit deterministic performance under all conditions. To achieve determinism, the system must account for both functional and timing requirements. Clearly, performance is important to maximize a system's potential. However, in the realm of CNC systems, the guarantee of correct and real-time quality of service (QoS) reigns paramount.

MTC provides a useful specification for machine tool and device integration, but to guarantee CNC QoS requirements, the need for better understanding of the network and resource performance is essential. We took a performance engineering approach to predicting the likely performance of systems [11] where our goal was to run some tests that would assess MTC QoS.

Several design issues required attention up-front. First, the question arises as to the frequency of MTC Agent sample updates. The first constraint was the OPC Server data access speed, which was limited to a rate at 10 Hz (or new data every 100 ms). This update limit was weighed against the requirements of the potential Application Clients. Sample and event updates of a second were deemed sufficient to satisfy the majority of requirements. Even though MTC allows bundling of a larger group of samples for storing and retrieving data to approximate real-time performance, one second updates preclude any sorts of real-time adaptive control, and so was deemed out of scope.

Within these update limits, the range of potential Application Clients was quite extensive, including shop floor dashboard, Overall Equipment Effectiveness (OEE), and data mining of asset and process knowledge. The next step was to use these design timing bounds and examine computational and network performance under various conditions. We were most interested in computational loading and QoS under progressively higher network traffic and under network failure conditions.

The testing setup included a desktop PC hosting the MTC Agent and a PC running both the OPC Client and the Siemens 840D "Sinutrain" CNC simulator. The Sinutrain CNC simulator includes an OPC server. This dual-PC networked setup has proved invaluable in testing and verifying the basic integration software. The MTC testing hardware consists of Commercial Off–the–Shelf (COTS) PCs and Ethernet networking: two Intel x86 CPUs, running Windows XP SP2 operating system (OS), communicating over a 10Base-T (100 Mbps) local area Ethernet network through a router. The CNC PC simulator

clock speed was 2GHz and had 2G Physical RAM. The MTC FEPC contained a Pentium 4, with controller clock speed was 3.59GHz, and had 2G Physical RAM.

We used the Intel Pentium processor Read Time Stamp Counter (RDTSC) feature which is ably suited for analyzing timing performance on single-processor machines. RDTSC tracks processor cycles and is incremented every clock cycle [12]. Since RDTSC keeps track of cycles, not time, resolution is based on processor frequency. Thus, a 1GHz CPU has five times the resolution as a 200MHz CPU.

MTC adopts the Internet paradigm, which is based on the Ethernet network and is a shared resource. Contention can result as more and more systems share the Ethernet, and at some point results in lower network throughput between any pair than the nominal bandwidth available. However, since the OPC Client and MTC Agent communicate over a dedicated Ethernet, contention is not a QoS issue.

Another QoS aspect of OPC Client and MTC Agent communication related to network failure – either the cable is broken or disconnected or the MTC Agent program is terminated or the FEPC is turned off. The OPC Client uses the Microsoft Windows Internet (WinINet) C++ API that enables applications to interact with HTTP, FTP and other protocols to access Internet resources. Initially, the OPC client configured WinINet functions using synchronous communication and default options for HTTP communication. Table 1 shows the test results timing the performance of `storeSample`, which sent a WinInet `HttpSendRequest` to the MTC Agent server under "Running" and "Down" conditions, with the Client and MTC Agent running on different computers. An average was generated from one thousand test cycles. The uncertainty $u_c$ of the measurements was computed using the square root of the sum-of-the-squares methodology.

**Table 1 Timing Under Different MTC Conditions**

| Action Between Machines | Time (sec) | $u_c$ (sec) |
|---|---|---|
| MTC Agent Running | 0.002030 | 0.000548 |
| MTC Agent "Down" – Default Timeout | 0.955262 | 0.037581 |
| MTC Agent "Down" – 20 ms Timeout | 0.030824 | 0.000118 |
| Network "Down" – 20 ms Timeout | 0.030889 | 0.000126 |

Clearly, there is significant performance degradation when the connection to the MTC Agent is down. This performance drop led to software reconsiderations. First, the substantial timing difference between "Running" and "Down" was determined to be due to the WinINet timeout setting, which was lowered. When WinInet was configured to timeout after 20 ms, a corresponding drop can be seen in Table 1 from almost one second to about 30 ms for between machine MTC messaging.

Second, WinINet asynchronous communication was evaluated as an alternative to the synchronous. In asynchronous communication, the data packet is sent without waiting for an acknowledgement. Because of the small size of the MTC Agent XML responses, asynchronous was slower due to a higher overhead and was determined to be unnecessarily complicating. Finally, if a HTTP problem was detected; the OPC-MTC software was changed so that MTC Agent communication was postponed until the next update cycle.

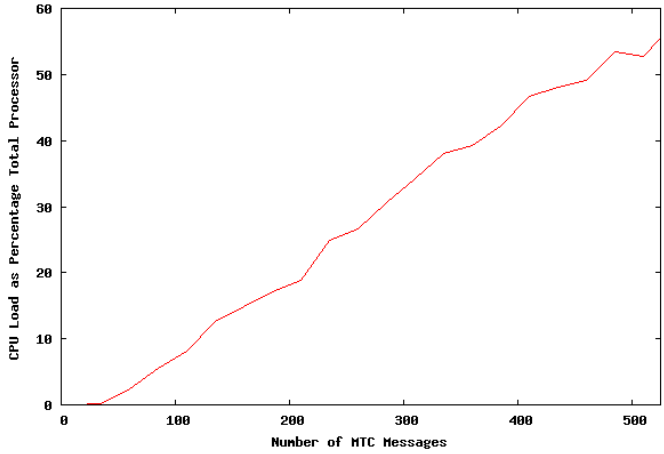


**Figure 4 Progressive MTC CPU Loading**

To assess the MTC QoS for CPU loading, a dedicated MTC–communication–only program was developed. Inside the program, a Windows timer ran every second and in stages sent progressively more and more `storeSamples` to the MTC Agent. While this progressive MTC message loading was running, a separate program captured the CPU loading using the Microsoft Performance Data Helper (PDH) Library [13]. PDH provides a query interface to retrieve a variety of detailed performance data, such as process CPU utilization, disk activity, etc. Figure 4 shows Progressive MTC CPU loading, captured as a percentage of total Windows processor loading. The graph shows MTC communication as a linear CPU loading factor. Our "Dual-Ethernet" system expected to make less than 50 MTC messages per second, so our expected MTC-only load was under 10%.

In conjunction with MTC communication, MTC Adapter could use either OPC synchronous polling or asynchronous notification updates. Data change notification requires less average bandwidth, but in some cases can succumb to excessive loading. Previous work found that data values can jitter minutely, triggering continuous OPC updates that resulted in a sluggish Operator Interface (ultimately no effect on the real-time CNC.) From this observation, even updates at a rate of 10Hz can lead to Operator Interface process starvation and must be acknowledged.

With this in mind, QoS design considerations were undertaken to prevent this problem. First, the OPC Client MTC Adapter communicated to the OPC Server using synchronous data updates exclusively. This data polling approach bounded the computational and communication bandwidth to a fixed limit, and was fixed at the Client Application update rate. Second, the OPC Client Windows process priority was reduced from "Normal" to "`IDLE_PRIORITY_CLASS`".

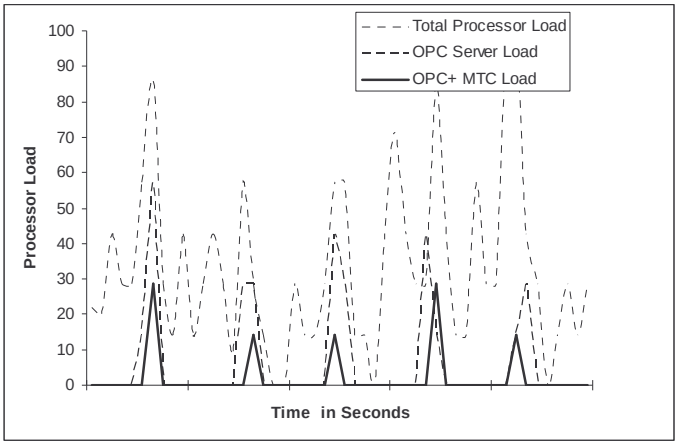


**Figure 5 Snapshot OPC and MTC Loading**

Next we considered the CPU loading when MTC communications was combined with MTC-OPC communication activity. Figure 5 shows the CPU loading mix for the Siemens simulation measured in 100 ms intervals. The test had 1) a part program a continuous loop, 2) 20 items of data communication per second update, and 3) loading measurement for Total Processor Load, OPC server loading and the combination of the OPC-MTC communication. Note, the Total Processor load often reflects the summed loading of the OPC Server and the OPC-MTC together. Our previous MTC-only tests measured loading to about 10%. When OPC and MTC communication are combined, surges in OPC-MTC loading double that to about 20% that occur for about 10 ms out of the 1 second cycle. Overall, there are surges in CPU loading that can be tied to the OPC Server and OPC. Although there is sufficient computational bandwidth to spare, we are evaluating better ways to bundle any data communication.

CNC kernels reside in a hard real-time operating system and are mostly immune Operator Interface issues. However, the Operator does interact with the CNC in a command and control fashion so that problems can arise should the Operator Interface fail or suffer process starvation. Often, the machine tool itself may not be vulnerable, but any part being made could be compromised. Since Windows XP is not a real-time operating system, there are no absolutes when it comes to evaluation of the QoS tests. The QoS tests do not indicate correctness of operation. Instead QoS tests can be best used to determine and avoid problems. Since QoS only measured for one set of computer hardware and software, the QoS tests can only offer realistic approximations to performance. However, using our QoS tests, we did find that a network disconnection led to excessive timeout delay, which we fixed. In this realm, the MTC QoS CPU and network QoS tests showed MTC QoS performance to be met our 1Hz performance bandwidth requirements.

## DISCUSSION

In this paper, we analyzed MTC Agent performance in a distributed “Dual-Ethernet” factory setup. We presented a summary of MTC features and an MTC configuration matching the requirements of Dual-Ethernet distributed shop floor architecture. In verifying the MTC QoS, our performance tests measured computational load under different network conditions – network failure and increasing computational and network load. If we limit our areas of concentration to machine monitoring with timing expectations on the order of seconds, while avoiding adaptive or other real-time control areas, we found that MTC performance exceeded our requirements.

MTC has laid the groundwork for a factory communication (HTTP with Decorated URLs) and Information Models (XML with XSD for structural correctness) that satisfied their initial goals. Since MTC leverages the software and hardware Internet paradigm, and because of the systemic prevalence of the Internet, there are an abundance of tools to ease MTC deployment. For example, Microsoft .Net Framework underscores the embracing of Internet technology with code to simplify parsing XML and handling HTTP communication for the average programmer. The Ga. Tech SDK reflects this parsimony, as much of the code leverages the existing .Net Framework in a concise and straightforward manner.

This is not to say MTC solves all the machine tool integration problems. Rather, although MTC recently approved its first official Version 1.0, it acknowledges that it is as a work-in-progress standard, with plans for improving alarms, and many manufacturing information model elements, such as tooling. We anticipate as the complexity of MTC specification grows, so will the evaluations of its efficacy. But, with our testing facilities in place, we should be able to quantify the QoS performance of future MTC versions.

## DISCLAIMER

Commercial equipment and software, many of which are either registered or trademarked, are identified in order to adequately specify certain procedures. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology or Boeing Aerospace, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

## REFERENCES

[1] C. D. Valenzano and L. Ciminiera, *MAP and TOP Communications: Standards and Applications.* New York: Addison Wesley Publishers, 1992.

[2] International Organization for Standardization, "ISO/IEC 9506-1, Industrial Automation Systems - Manufacturing Message Specification - Part 1: Service Definition," Geneva, Switzerland: 1990.

[3] International Organization for Standardization, "ISO/IEC 9506-1, Industrial Automation Systems – Manufacturing Message Specification - Part 2: Protocol Specification," Geneva, Switzerland : International Organization for Standardization, 1990.

[4] A. Vijayaraghavan, W. Sobel, A. Fox, P. Warndorf, and D. Dornfeld, "Improving Machine Tool Interoperability Using Standardized Interface Protocols: MTConnect," University of California, Laboratory for Manufacturing and Sustainability, Green Manufacturing Group: 2008.

[5] The World Wide Web Consortium, "Extensible Markup Language (XML) 1.0 (Fourth Edition),". T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, Eds. 2006.

[6] The Internet Society, "Hypertext Transfer Protocol - HTTP/1.1," http://www.isoc.org : 1999.

[7] S. Venkatesh, B. Sides, J. Michaloski, and F. Proctor, "Case Study in the Challenges of Integrating CNC Production and Enterprise Systems," in *Proceedings of IMECE 2007 ASME International Mechanical Engineering Congress and Exposition* Seattle, Washington: 2007.

[8] S. Venkatesh, R. Morihara, J. Michaloski, and F. Proctor, "Closed Loop CNC Manufacturing - Connecting the CNC to the Enterprise," in *Proceedings of IDETC/CIE 2007 ASME International Computers and Information in Engineering Conference* Las Vegas, NV: 2007.

[9] Microsoft Corporation, "COM Specification," www.microsoft.com/com.

[10] Georgia Institute of Technology, "MTConnect .NET Agent Software Development Kit," http://www.mtconnect.org: 2009.

[11] F. Gao and J. Gutierrez, "Challenges in Providing Management Strategies For Networked Systems," in *Proceedings of the First International Conference on Information Technology (ICITA)* Bathurst, Australia: 2002.

[12] Intel, "Intel Processor Identification and the CPUID Instruction," Application Note 485: 2008.

[13] K. Braithwaite, "Custom Performance Analysis using the Microsoft Performance Data Helper," *IBM WebSphere Developer Technical Journal*, Oct.2003.