



# Framework Implementation Project Status Report

August 9, 2000

Andrew Dugenske  
Manufacturing Research Center  
Georgia Institute of Technology



## Introduction

### *Overview*

This report describes the status of the Framework Implementation Project, which is being conducted at the Georgia Institute of Technology's Manufacturing Research Center (MARC). Intel, Avaya (Lucent), DEK Printing Machines, and Fuji are sponsoring this project. Other participating companies include Agilent, Celestica, Cyberoptics, and Universal Instruments. The project follows the recommendations of the National Electronics Manufacturing Initiative (NEMI) Plug and Play Factory Project, which addressed the need for a common framework for electronics manufacturing. The purpose of this project is to build upon the work conducted by the NEMI project, address the unanswered questions posed by the NEMI project, develop specific solutions to the needs of the sponsors, accelerate the acceptance of standards, and potentially demonstrate the concepts at sponsor sites. The project makes use of the Center for Board Assembly Research (CBAR) surface mount technology lines that are located in MARC.

Over the past few months, the project team has built the preliminary infrastructure to accomplish the project goals. This infrastructure is capable of demonstrating many of the emerging technologies that have been recommended by the Plug and Play project and various emerging standards for the transfer of information within an electronics manufacturing enterprise. The constructed framework consists of applications, equipment adapters, and a messaging system--all based upon standards. Now that this infrastructure has been built, many of the issues associated with defining a complete framework for electronics manufacturing can be addressed and solved.

This project team can greatly assist in recommending a complete framework for electronics assembly by combining expertise, experimentation, and consensus building. Once a suitable framework has been designed, tested, and documented, the electronics manufacturing industry should see significant economic benefits from a flexible, low cost, and adaptable factory information system. Until that time, factories will continue to lag corporate business needs.

### *Background*

The National Electronics Manufacturing Initiative (NEMI) publishes a technology roadmap every other year. Hundreds of individuals, representing most of the major electronic manufacturers and suppliers in the industry, contribute to the roadmap's development. The purpose of the roadmap is to predict how technology will evolve in the future and identify technological gaps that will greatly impede industry growth.

A strong conclusion of the past few roadmaps is that factory information systems (FIS) are **too expensive** and are **too inflexible** to adapt to the accelerating change in electronics manufacturing. Consequently, several members of NEMI such as Celestica, Solectron, Intel, Lucent, Universal and GenRAD, undertook a project entitled "The NEMI Plug and Play Factory

Project.” The purpose of the project was to develop a framework that **reduces costs** and **decreases cycle time** by fostering interoperability among software applications and equipment by making use of a framework based on standards. (Please see figure 1.)

The NEMI Plug and Play project was conducted from December 1997 until December of 1999. A great deal was accomplished during that time period. Several framework technologies were investigated by the group and tested on the Surface Mount Technology (SMT) lines located at Georgia Tech. In addition, several industry standards have been undertaken through the IPC that provide a document standard that can be used in the future. More specific information about the motivation for undertaking the project can be found in a journal paper written by members of the project team<sup>1</sup>.

Unfortunately, the job of defining an industry based framework for use in electronics assembly is a large one. Although the NEMI project made a great deal of progress, there were a significant amount of unanswered questions raised by the project that have not yet been answered. The purpose of this project is to build upon the expertise gained through the NEMI project, and address many of the unanswered challenges so that a practical framework for electronics assembly can be recommended to the industry.

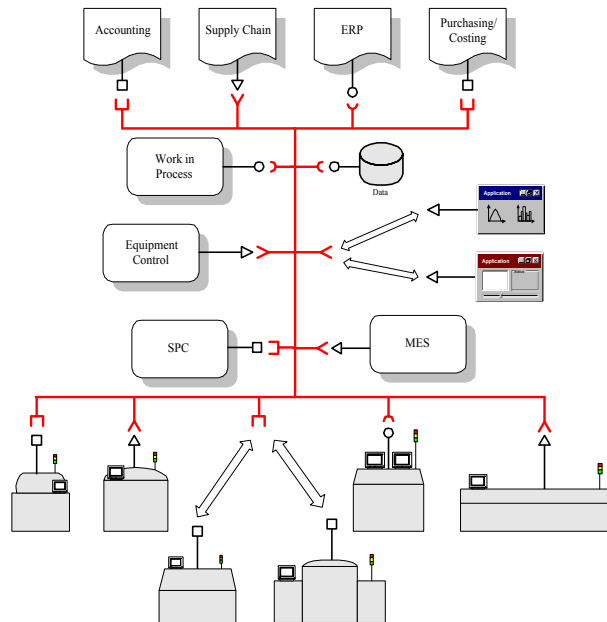


Figure 1. A plug and play framework in which equipment and applications can easily be installed or exchanged.

### ***Previous Work***

The NEMI project team investigated six different sets of framework technologies. Each set was suggested by the framework definition task team and was tested by the test bed task team at Georgia Tech’s Center for Board Assembly Research (CBAR) SMT lab. Whenever possible, current standards were used and industry practices observed. Through the project, several technologies showed a great deal of promise. The basic technologies recommended by the project are summarized in Table 1.

Table 1. Technological recommendations of the NEMI Plug and Play Factory

Use	Technology
Wire Protocol	TCP/IP
Middleware Type	Message Oriented Middleware (MOM)
Message Transfer Protocol	Hypertext Transfer Protocol (HTTP)
Message Syntax	Extensible Markup Language (XML)
Message Semantics	IPC 2500 Series

It is envisioned that a framework making use of a message-oriented middleware (MOM), that passes XML messages by HTTP will be the technological building blocks of a recommended framework. The message content will be based on standards underway through the IPC. Although these technologies provide a fine foundation for building a framework, additional work must be undertaken to define specifically how these technologies will be used to construct an appropriate framework.

## **Current Activities**

### ***Architecture***

The NEMI project reviewed six different architectures to determine their suitability for electronics manufacturing. After reviewing the various frameworks, it became apparent that the technologies outlined in Table 1 should be used for the framework, but it was not clear exactly how these technologies should be applied. Consequently, the first step was to define an architecture for this project from the lessons learned through the NEMI project. This architecture would then dictate how the technologies are to be used.

It became clear early in the NEMI project that XML messages would be passed among entities in the framework. A clear vision as to how these messages would be passed among framework entities had not been articulated. Consequently, one of the first major architecture decisions was to outline a way in which XML messages would be exchanged. Initially, it was thought that all entities would act as web servers and that messages would be exchanged using HTTP. Requiring that each entity act as web server to participate in the framework seemed like an undue burden to place on every entity that desires to participate in the framework. Plus, web servers don't provide all the functionality that is required to pass messages.

### ***Message Broker***

Consequently, the team decided to make use of message-oriented middleware (MOM) with an HTTP interface to pass XML messages in the framework. MOM is ideally suited for this task. It provides many services such as multi-user connection capabilities, message routing, load balancing, publish and subscribe services, security, and fault tolerance. Plus, there are several MOM products on the market that can be used for this purpose such as MQSeries by IBM, MSMQ by Microsoft, Java Message Queue by Sun, and Smart Sockets by Talarian. By making use of a central MOM product, each entity would not have to implement MOM functionality independently. Rather, MOM services would be implemented by software specifically designed for this purpose.

Therefore, the team proposed an architecture that makes use of a centralized message broker as indicated in Figure 2. Applications and equipment (entities) will connect to the message broker via HTTP and exchange XML messages through the broker. The message

broker will act as a web (HTTP) server, and the entities will act as web clients. This technique greatly simplifies framework requirements of applications and equipment within the framework.

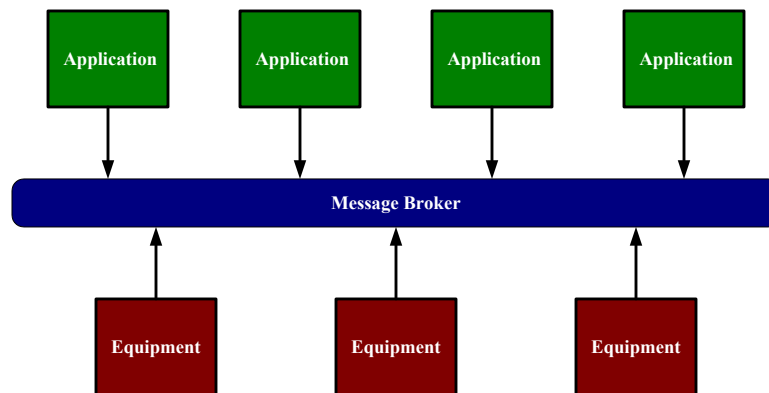


Figure 2. Framework architecture making use of a central message broker.

If this architecture is to be offered as a commercial product, the project team felt that the message broker must be economically feasible. In other words, it should be available off the shelf or easily constructed from available software components. If a low cost solution is not available to instantiate the message broker, it is doubtful that this architecture will be commercialized—a goal of this project.

After some searching, a middleware product that provides an HTTP interface could not be found. Hence, the team decided to build one by combining a web server and a middleware product. The web server provides the HTTP functionality, and the middleware provides the other services. Custom code was written that connects the two applications.

Early in the investigation it was determined that a persistent connection between the message broker and the framework entities is required to reduce communication bandwidth. HTTP 1.1 web servers adhere to a persistent scheme, whereas servers adhering to HTTP 1.0 do not. Hence, the web server used in the message broker would have to adhere to the HTTP 1.1 specification.

The combination of Microsoft's Internet Information Server (IIS) and Talarian's Smart Sockets was the first combination of applications tested. IIS was suggested since it is included with Windows NT at no cost. Smart Sockets was used since the API appeared straightforward and well documented. Custom code was developed that communicated with IIS through the ISS API and contained API calls to Smart Sockets.

This combination was not found to be practical since IIS is not well documented, requires complex programming, its behavior is often unpredictable, and does not adhere to the HTTP 1.1 specification. In addition, multiple connections between IIS and Smart Sockets cannot be accomplished without using a third party thread management application.

To overcome these challenges, a set of applications which were all written in Java were used. By combining products all written in Java, it was felt that thread management could be handled by the Java virtual machine. Hence, the second set of applications tested were Sun's Java Web Server (JWS) connected to Sun's Java Message Queue (JMQ) through Java servlets. Unfortunately, the JWS did not adhere to HTTP 1.1. To overcome this limitation, the team tested the Apache Web Server but found that it also did not adhere to HTTP 1.1. Each server

would break the connection between a client and the server after only a single message was passed. Reestablishing a connection after each message passed would require too much time.

Finally, the team made use of the Tomcat server. Tomcat was developed by Sun, was donated to the Apache Software Foundation and can be downloaded at no cost. Although it currently does not conform to the HTTP 1.1 specification, the project team was able to modify it to meet HTTP 1.1. Figure 3 shows the structure of the message broker using the Tomcat server, a servlet, and the Java Message Queue (JMQ).

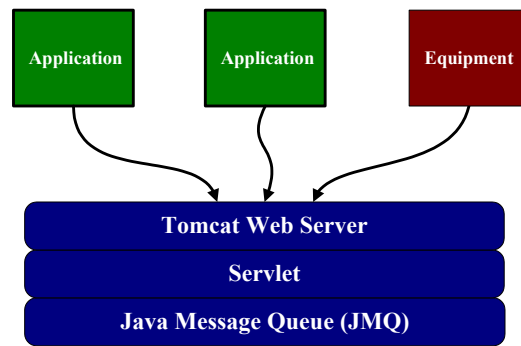


Figure 3. The current message broker using the Tomcat web server, servlets, and the JMQ.

To communicate with the message broker, applications and equipment exchange XML messages with the broker. A generic wrapper structure was developed by the project team and is used by the broker to route messages to the appropriate location. The wrapper contains two sections—a control and a data section. The control section contains information such as the message destination, message sender, message date, and type of message. The data section contains the data to be sent or the “payload.” This wrapper can be used to transfer information of any type within the framework. An XML schema has been developed for the wrapper and has been proposed as an IPC 2501 standard. It can be found at [www.fis.marc.gatech.edu/xml](http://www.fis.marc.gatech.edu/xml).

The Tomcat server, the Java Message Queue, associated servlets, and the IPC 2501 schema are currently being used in the project and appear to be providing satisfactory performance.

### ***Equipment Adapters***

The IPC 2541 is the first standard being developed to define the messages that are to be exchanged within the framework. The 2541 defines a generic set of messages that can be used to communicate with electronics manufacturing equipment. Hence, the first XML messages that were exchanged on the framework were based on the 2541 standard.

Unfortunately, the 2541 had not developed an XML Schema which defines XML versions of the messages to be passed. Hence the project team developed an XML schema for the emerging 2541 standard and proposed it to the standards development team.

In addition to the 2541 messages, several other messages have been developed and are being used by the project team on the framework. These messages have been documented in a separate schema, the GT2541Extensions schema. The 2541 standards development team (or alternate) may wish to consider these messages for inclusion in a standard. Both the 2541 and the GT2541Extensions schemas can be found at [www.fis.marc.gatech.edu/xml](http://www.fis.marc.gatech.edu/xml). They have been

placed on the web so that others may comment on their acceptability. Had this project not been undertaken, the XML schemas most likely would not have been written at this early date.

Since the method in which IPC 2541 messages are exchanged on a framework has not been defined, and the IPC 2541 standard has not been completed, no equipment currently communicates using IPC 2541 messages. To emulate machines that would communicate using IPC 2541 messages, equipment communication adapters have been written. The adapters communicate with the framework using XML messages, emulating a machine that is fully compliant to the emerging IPC standards. They communicate with the using the SECS/GEM protocol. The adapters allowed real-time messages to be exchanged between the equipment and the framework as if the equipment were fully IPC 2541 compatible. (Please see Figure 4.)

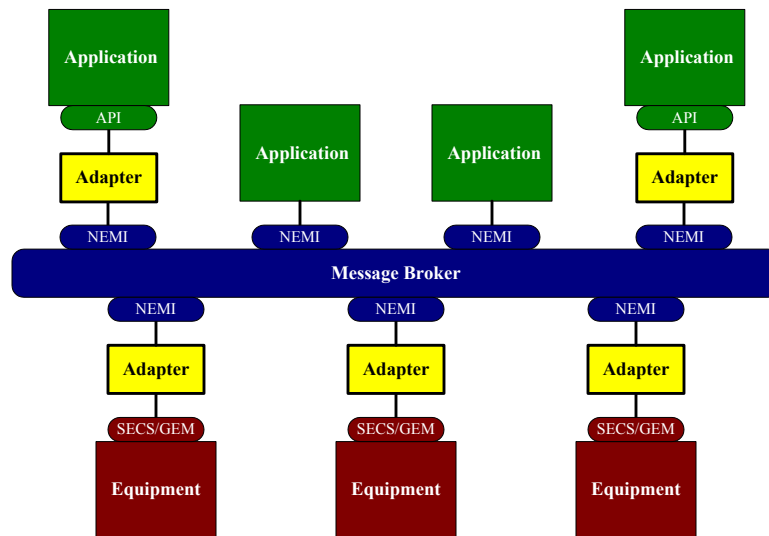


Figure 4. The current framework with equipment and application adapters.

The adapters are capable of communicating in a synchronous mode (respond to a request) or asynchronous (event driven) mode. Synchronous mode allows applications in the framework to query equipment about certain parameters, whereas asynchronous mode allows equipment to provide messages when conditions are met.

The NEMI Plug and Play project made use of Cimatrix software for SECS/GEM communication. The Cimatrix software uses a constantly running application, GEM Host to send information to equipment. Applications that communicate with SECS/GEM equipment via the Cimatrix software first communicate to the GEM Host application, which in turn communicates to the equipment. GEM Host communicates with user written applications through C call back functions. This method of communication was thought to be too complex, especially if the user developed program was written in Java, since an intermediary program would have to be written in C to communicate with the GEM Host. With an intermediary program, a message would have to be handled by three separate applications before it reached the desired piece of equipment.

To reduce equipment communication complexity, the GW Associates SDR software was next product that used to build an equipment communications adapter. The GW Associates software does not require a separate application to communicate with SECS/GEM equipment as with the Cimatrix system. Rather, C libraries are referenced from a calling program written by

the user. The SDR software allowed the development of a simpler communication scheme, but did not allow for asynchronous communication. Hence, a machine would have to be “polled” to determine the value of a parameter, versus receiving an event notification when conditions changed. Not being able to receive events appeared to be unacceptable since parameter changes could be missed, and the efficiency of communication would greatly suffer.

Using the Cimetrix and GW Associates software required that the user-developed applications be written in C. Communicating with the message broker, however, is much more straightforward using Java since communication with the broker is accomplished through HTTP. HTTP requires a simple socket connection which can be accomplished by using the appropriate Java class. Whereas, C requires that developers write socket drivers using low level function calls. This was found to be tedious and error prone.

The next software solution evaluated by the team was the GEM Equipment Libraries (GEL) produced by Intelware of Moscow, USSR. The GEL software consists of a number of Java software classes. They are simply imported into a user developed Java program to accomplish both synchronous and asynchronous SECS/GEM communication. Using the GEL libraries, a single application could be constructed that can communicate to both the equipment and the framework. The current equipment communication adapters make use of the GEL software.

In addition to passing information to the equipment and the framework, the equipment communication adapters must also be able to process XML and SECS messages. The appropriate parsers and interpreters have been embedded in the adapters to accomplish these translations.

### ***Application Development Background***

The Factory Information System Framework Architecture is a functional test bed for the NEMI Plug and Play concept and the standardization efforts for generic machine information exchange (e.g., IPC 2541, IPC 2501, and IPC machine sectionals). This Plug and Play concept is designed to support the breadth of communication needs for the manufacturing equipment used in state-of-the-art factory settings. There are four key forms of information exchanged in the envisioned architecture:

- board data consisting of the attributes of the manufactured products, such as unique identifiers and product classifications;
- machine data consisting of the states of both present and past performance in its operation
- process information concerning the interrelationships between boards and the machines producing those boards
- publish and subscribe information exchange to control the information flow identified in the first three bullets above.

The developmental nature of the Framework architecture and information exchange is supported by a flexible user interface design to permit testing the user interface information display implications under the following three typical uses.

Shop Floor Surveillance. Detect, localize, track, classify, identify, and report the boards under production in the manufacturing environment and the machines producing these boards. On-site and remote surveillance will be considered.

Archival Analysis of Shop Floor Production. Detect, localize, track, classify, identify, and report boards already produced from the manufacturing environment and the machines performance in producing these boards.

Concept Demonstration and Familiarization. Training of manufacturing partners and potential interested parties in the roles they may play in the above missions.

### ***Application Development Status***

A preliminary front-end mission, function identification, function allocation, and task requirements analysis of the Framework User Interface was conducted. Two informal interviews of research scientists at MARC were conducted (one was the project director), and a review of sponsor requirements for the envisioned Plug and Play framework were used to further identify the information requirements and verify initial assumptions. The outcome of this analysis identified an initial (unverified) design of the user interface information requirements and also defined the scope for its operation. The operational modes, and principle usage of the interface were mocked up as a notional design that was reviewed by the FIS Framework sponsors. As the framework design evolved, the user information requirements were adjusted to continue to meet the three required uses as described in the background section above.

A Back-End GUI Services software structure was developed to meet the requirements for data persistence and real-time processing considering these scope-limiting and design requirement factors:

- Use of SQL-compliant database technology for data storage.
- Business-oriented firewall security constraints on the method of information connection between Framework machines (exclusive use of port 80).
- Maintain an equipment configuration in terms of enterprise, factory floor, line, and SMT board types as privately held data (not communicated by the Framework).
- Maintain an activity/event archive with associated marked time interval tracking.
- Subscription to a message broker with persistent connection for data queues.
- Back-end interface with subscription control of Framework XML messages, and status and error reporting of message parsing and interpreting.
- Translation of XML message into a XML event based communication of machine state and board movements appropriate for information display on the front-end client.
- Real-time content parsing of the XML messages received from the subscribed queues of the message broker with appropriate error checking to gracefully handle invalid/improperly formatted XML IPC messages.
- Real-time storage of events for archival retrieval and reporting.
- Real-time event notification to GUI Front-End clients (described below).
- Real-time calculation of process-monitoring statistics for production and machine utilization.

A proof-of-concept GUI client framework that met the functional description identified in the front-end analysis discussed above and provided the real time display of FIS Framework

process-oriented data was developed. This design supports multiple GUI clients connecting through a custom XML communication protocol to the Back-End GUI Services.

A proof-of-concept GUI client interface with a custom set of interface widgets for display of data information items such as machine state, utilization, heartbeat, cycle time, product movement, and alerts was developed. The user interface design defines consistent areas for information display and interaction including navigation, framework alerts, focus area of interest, dynamic titling of focus window, amplification area, and selection area for interface mode of operation. Navigational features support selecting the framework data layer for display (enterprise, factory, and line). Interface modes of operation include real-time and archive presentation of framework data, selection of information items in the focus, and future support for reports and analysis.

A set of generic icons for machine display of each of the categories of processes covered by IPC 2546 (should this be another number?? I didn't find this number in your first proof??) that includes material movement systems like conveyors and buffers, manual placement, automated screen printing, automated adhesive dispensing, automated surface mount placement, forced convection, and infrared reflow ovens were developed. Each icon is represented in normal operational state as well as pass-through processing mode of operation.

The general framework Java code development was supported with an initial tutorial on Java code development and Java package construction and as-needed support for developing and debugging Java, including XML optimized parsing and socket programming.

### ***Support Tools***

To test various components associated with the framework, a series of framework tools were developed. The tools consist of a message sender, a message receiver, and various simulators. These simulators can inject messages into the message broker, emulating a framework entity. They have been invaluable in debugging the applications, services, and systems associated with the project. Figure 5 shows a detailed representation of the current framework including the message broker, equipment adapters, application specifics, and framework tools.

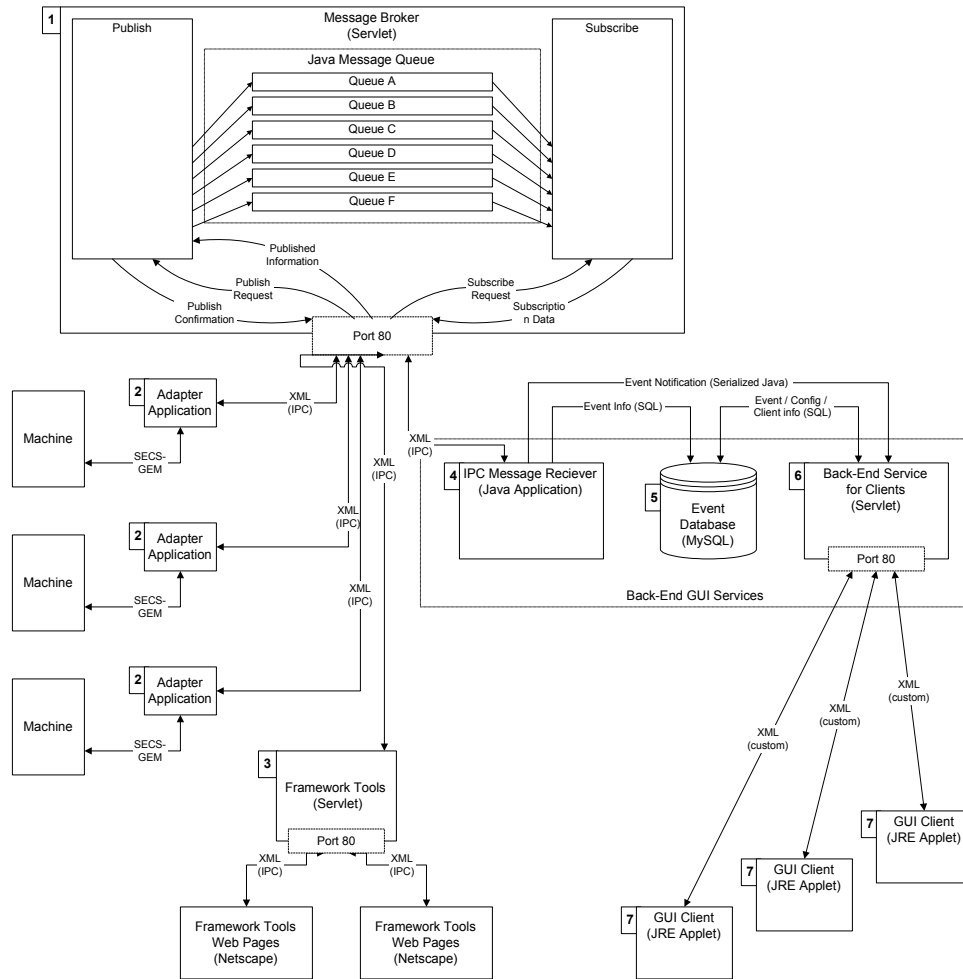


Figure 5. A detailed representation of the current framework containing the message broker, equipment adapters, application specifics, and support tools.

## Conclusion

Although this project team has made significant progress over the past few months, the task of recommending a framework for electronics manufacturing is challenging. The first few months of the project have primarily been spent building relationships, software infrastructure, and hardware capabilities so that the concepts recommended by the team can be tested. With these logistics in place the team is making significant progress that can only be made once the appropriate infrastructure is in place. Over the next few months, we feel that this project will provide significant benefits to the sponsors, and the industry as a whole.

In addition to the current project members, several companies have inquired about participation. They include: Universal Instruments, Siemens, MAPICS, GenRAD, Panasonic, Philips, among others. By continuing this project and working with the appropriate partners, the current sponsors should receive significant return on investment over the next few months. It is also anticipated that several IPC standards will be finished more rapidly by receiving the results of this project.