Interoperability Testing for Shop Floor Measurement

Fred Proctor
NIST
100 Bureau Drive, Stop 8230
Gaithersburg, MD 20899
frederick.proctor@nist.gov

Bill Rippey
NIST
100 Bureau Drive, Stop 8230
Gaithersburg, MD 20899

John Horst, Joe Falco and Tom Kramer
NIST
100 Bureau Drive, Stop 8230
Gaithersburg, MD 20899

***Abstract*—** Manufactured parts are typically measured to ensure quality. Measurement involves equipment and software from many different vendors, and interoperability is a major problem faced by manufacturers. The I++ Dimensional Measuring Equipment (DME) specification was developed to solve interoperability problems and enable seamless flow of information to and from dimensional metrology equipment. This paper describes validation testing of the I++ DME specification. The testing was intended to improve the specification and also to speed up its adoption by vendors. Testing issues are described, and a software test suite is detailed. Interoperability testing with real equipment was done over several years, and lessons learned from the testing will be presented. The paper concludes with recommendations for improving this type of testing.

***Keywords*:** *interoperability, measurement, software testing*

I. Introduction

Automated geometric inspection of parts is done using coordinate measuring machines (CMMs). Traditionally, CMM vendors have sold a tightly-coupled software-hardware system for programming and controlling the inspection process. The last 15 years have seen large manufacturers acquire CMMs from many different vendors and endure the overhead of supporting multiple software applications. Further, 3rd party software vendors have been offering high quality products that often cannot be used because they are incompatible with some CMMs.

Automakers are major users of measurement equipment, and suffer from the cost and time to work around these incompatibilities. They have responded by supporting a specification for dimensional measurement equipment interoperability, called the I++ Dimensional Measuring Equipment Interface specification (I++ DME). The goal of I++ DME is to allow automakers, and any other manufacturers, to select the best software and equipment for their purposes and budgets and ensure that they work together seamlessly out of the box.

Specifications, like any result of a human endeavor, are never perfect and need to be tested (validated) to make sure they fulfill their requirements. For I++ DME, this means answering the questions, "Does I++ DME handle all of today's measurement activities, or are important types of measurements or equipment left out? Is the specification written clearly and unambiguously, or will implementers have to make assumptions?" Likewise, products that claim to support I++ DME are never perfect and need to be tested (verified) to make sure they comply with the specification. This means answering the questions, "Does the product send only valid I++ DME messages? Does it respond appropriately to both valid and invalid messages?"

NIST has written an I++ DME test suite designed to help the specification writers make a better specification and the product vendors make better products. The test suite includes a simulated client that acts as the software that runs measurement plans, and a simulated server that acts as the equipment that makes the measurements. Test scripts cover all measurement activities, from startup through measurement and shutdown, including error conditions. A logging feature allows for later analysis of test results.

The I++ DME has undergone testing in a series of demonstrations involving real software and equipment at several important international quality technology expositions, including the 2004 International Manufacturing Technology Show (IMTS), the 2005 Quality Expo, and the 2005 – 2007 Control Shows. These multivendor demonstrations have included combinatorial testing of several software packages with several measurement machines. Comments from the participants, and their continuing participation, show that this level of testing rigor is valuable and helps to ensure quality products that meet customer requirements.

II. The Measurement Process

Before parts can be measured, they must be designed and at least partially manufactured. Design is normally done using computer-aided design (CAD) workstations that generate electronic design files that define the product requirements for subsequent downstream manufacturing operations. From the point of view of measurement, the design files contain dimensions and tolerances, and other requirements such as surface finish. A standard for the output of CAD information is ISO 10303, "Standard for the Exchange of Product Model Data," also known as STEP [1]. STEP Application Protocol

(AP) 203 deals with design data; the second edition includes geometric dimensioning and tolerancing.

Although not part of the measurement process, computer-aided manufacturing (CAM) and computer- numerical control (CNC) are steps that define how the part is to be manufactured. It is worth noting that manufacturers would like to inspect as much as possible on the equipment used to manufacture the parts, in order to save the time it takes to move parts between equipment. Supporting this flexibility is one goal of interoperability specifications like I++ DME.

Given a part design, measurement plans are then developed which guide how specialized equipment or human experts are to inspect the part. A standard for the output of measurement planning is the Dimensional Measuring Interface Standard (DMIS) [2]. DMIS plans define the measurement sensors to be used (typically touch probes), features to be measured (such as surfaces and holes), and reports to be made.

Measurement plans are executed by software that connects to measurement equipment such as coordinate measuring machines. During this phase, commands are directed toward the equipment to select sensors, capture points of interest and return the results. Measurement plans may consist of thousands of individually acquired points, with coordinate systems set and branch points taken depending on intermediate results. The I++ DME specification covers the exchange of data between the execution software and the measurement equipment.

Once measurement data has been acquired, an analysis phase is performed in which the raw results are compared against the design requirements (e.g., dimensions and tolerances) so that quality conclusions can be made. A draft standard for reporting results is the Dimensional Markup Language (DML), being prepared by the Automation Industry Action Group.

While interoperability between these different phases of measurement is the overall goal, this paper focuses on validation testing of the I++ DME specification. The authors are conducting similar testing on STEP, DMIS and DML.

III. Challenges for Standards-Based Measurement

A challenge for any standards-based activity is constraining the data exchange to a set that can be documented and thus standardized, while enabling vendors to innovate their products and thereby benefit manufacturers. For measurement, this challenge is made more difficult by the wide range of equipment used for measurement, and the many types of measurements done. For example, measurement equipment includes sensors such as touch-trigger probes, capacitance gages, lasers and other optical sensors; and machines ranging from small hand-moved portable arms through large granite-based fully automatic coordinate measuring machines. This technology continually evolves, and defining a set of capabilities to be used as the basis for a standard is difficult and requires compromise. In any case, there must be a process in place to revise the standard as technology improves and new sensors and measurement capabilities become available.

IV. The I++ DME Specification

The I++ committee is comprised of measurement equipment end users primarily from the automobile manufacturing sector. The I++ Dimensional Measuring Equipment (DME) specification [3] was written by I++ members and targeted toward equipment and software vendors. The goal was to enable manufacturers to pick best-in-class equipment and software reflecting their particular needs for sensor type, part size and measurement tasks.

I++ DME is a messaging protocol between measurement plan executors and measurement equipment. It uses TCP/IP sockets as the communication mechanism, and defines a message set and a client-server architecture. Clients are measurement plan executors, and servers are the equipment that carries out the measurements. For example, a client could read DMIS measurement plans produced by some upstream application, interpret the DMIS statements, send I++ DME messages to the measuring equipment, accumulate the measurement results that return as I++ DME messages from the server, and output a DMIS or DML measurement report. This is shown in Figure 1.

I++ DME consists of Unified Modeling Language (UML) descriptions of the messages, accompanied by natural language (English) that describes the semantics. Production rules in Backus-Naur Form (BNF) are provided that define the syntax of message composition. Numerous examples are provided as guidance to implementers. A sample I++ DME session is shown below, with messages from the client not underlined and responses from the server underlined.

```
00002 StartSession()
00002 &
00002 %
00003 GetDMEVersion()
00003 &
00003 # DMEVersion(1.4.2)
00003 %
00027 ChangeTool("ProbeB")
00027 &
00027 %
00078 SetProp(Tool.GoToPar.Speed(25.0))
00078 &
00078 %
00079 GoTo(X(2.626), Y(-4.656), Z(-4.100))
00079 &
00079 %
00094 PtMeas(X(2.47), Y(-4.13), Z(-5.10),
IJK(-0.01,-0.99,-0.00))
00094 &
00094 # X(2.44), Y(-4.64), Z(-5.99),
IJK(-0.019,-0.997,0.074)
00094 %
```

V. I++ DME Testing

As a product of a human endeavor, the I++ DME specification inevitably contains errors. The purpose of validation testing is to find the errors and suggest changes to the specification that fix the errors, before the specification is published and implementations are released. Validation ensures that the specification is complete, correct and

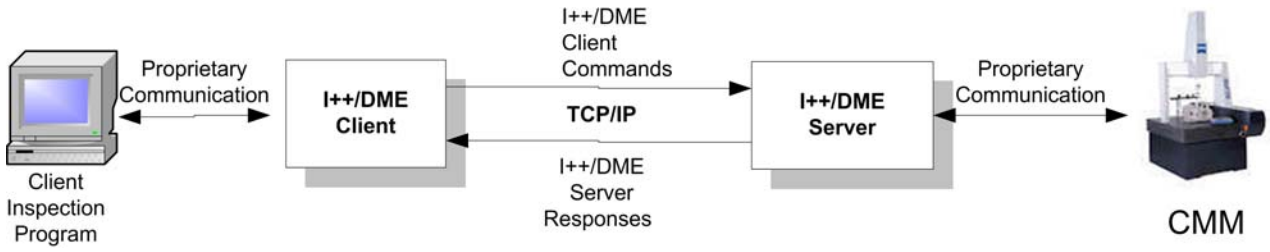

Fig. 1. The I++ DME activity model.

unambiguous. "Complete" means that it covers all the requirements set forth by the I++ members. Due to compromises, these may not completely satisfy the requirements of everyone. Nevertheless, it is the job of validation testing to discover any requirements that are not expressible in I++ DME. "Correct" means that there are no factual errors, including typographical errors but also inconsistencies in descriptions and conflicts with stated requirements. "Unambiguous" means that two readers of the specification will agree what is meant. This is difficult to achieve in practice, if for no other reason that the authors do not all speak the chosen natural language (English) as their native language. Ambiguity can be mitigated through the use of pictures or figures, and good examples.

Another objective of testing was to speed the commercialization of products that support I++ DME. This was achieved as a side effect of including vendors in the testing activities.

Testing can also lead to product conformance, if the testing tools persist after validation testing has concluded. In this case, all the hard work of testing can benefit newcomers, who can run the tests themselves privately and improve their products before releasing them.

The approach to testing taken by the authors was to provide a software test suite that enables controlled, comprehensive testing, in source code, paired with a series of public interoperability tests and demonstrations at trade shows that included real products and real measurement tasks.

VI. The I++ DME Test Suite

The I++ DME Test Suite [4] was written by the authors as a utility to enable internal testing of conformance to the specification. It is comprised of two applications, a server and a client, many test scripts, and source code for a C++ class library and parsers that parse client and server messages. The source code is free and intended to help newcomers implement I++ DME without having to incur the tedium of developing message handling code.

Figure 2 shows the I++ Server Utility. The server simulates the response of measurement equipment to I++ commands, maintaining a coarse world model and simulation of a coordinate measuring machine and responding plausibly to requests from a client. Developers of client software typically use the Server Utility as a stand-in for real servers (e.g., coordinate measuring machines) that are expensive to obtain. Developers of client software can use the Server Utility to verify that their commands are valid, and to see what responses they should be prepared to receive. The Server first opens up a socket on a port specified by the user, and awaits connections from a client. Every message received or sent by the Server is logged, displayed in a window and written to a file. Some attributes of the simplified models are configurable, for example the radius of the probe.

Figure 3 shows the I++ Client Utility. The client simulates the actions of plan execution software, sending requests to the server to select sensors and measure attributes of the part, and collecting responses back for later analysis. Developers of server equipment typically use the Client Utility as a stand-in for execution software. This allows them to see what commands they are expected to handle, and to check that their responses are valid. The client connects to a running server on a socket specified by the user, who then loads a script file for reading and execution, similar to the excerpt shown below:

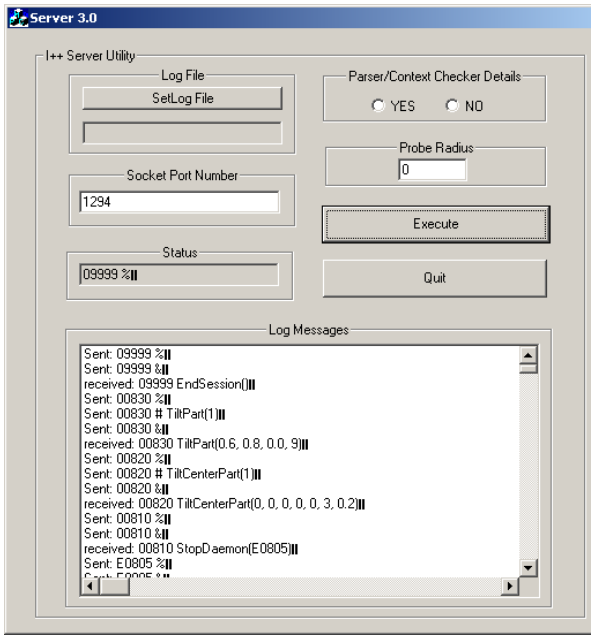

Fig. 2. The I++ DME Server Utility used is a surrogate for measuring equipment, used for testing client software.

```
AlignPart(1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 2.0)
AlignTool(0, 0, 1, 30)
CenterPart(2.0, 3.0, 4.0, 0.1)
ChangeTool("Probe1")
```

Each script file of I++ DME commands has an associated response file that is compared against what is received from the server. If responses don't match what is expected, errors are noted in the log file. These errors are not necessarily true errors, since the server messages in general include data points that vary depending on the actual sensed values of probe points. Strict comparisons against a pre-written response file may not match exactly yet still be valid. This is a challenge for automated testing, and one that requires balancing the difficulty of building an intelligent automated analysis tool against the value it provides, given that people will eventually be viewing the results and can be expected to make more difficult determinations of acceptability.

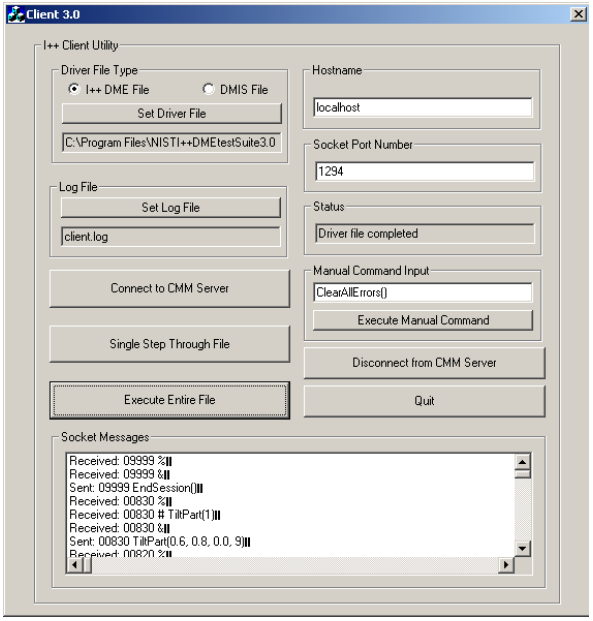

Fig. 3. The I++ DME Client Utility is a surrogate for measuring plan execution software, used for testing measuring equipment.

VII. Public Demonstrations

The I++ Test Suite allows developers to build compliant applications within their companies and test them before releasing them to their customers. At some point, applications will be run in production at customer facilities, and will interface with compliant applications from other vendors. It is important to have some experience with production interoperability prior to full release. This is the purpose of public demonstrations.

Three I++ public demonstrations have taken place, during the Control Shows in 2005, 2006 and 2007. The participants

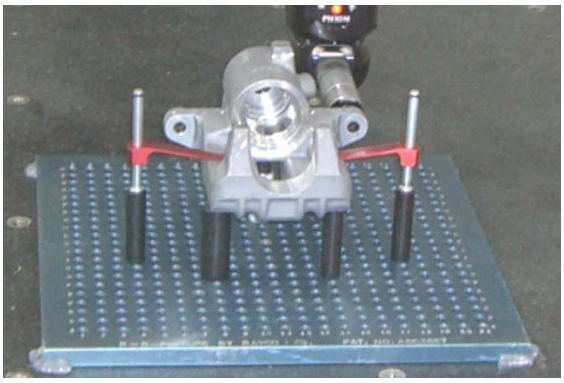

Fig. 4. Representative automobile part used for public demonstrations.

varied during each show, with the intent to include some number of client providers (e.g., measurement plan execution software developers) and some number of server providers (e.g., coordinate measuring machine builders). In 2007, the public demonstration included six clients and four servers, for 24 combinations possible for testing.

Unlike private testing with the I++ Test Suite, public demonstrations used real measurement plans (e.g., DMIS or some vendor proprietary plan formats) and real parts. A representative automobile part was selected, as shown in Figure 4. No test scripts were used, and thus no pre-written response files were written. Tests were done point-to-point, client-to-server, with people observing the measurement process on the machines and determining if the results of the measurement were acceptable.

The burden on the test judges was lessened somewhat by their experience with the test part. It was usually obvious when failures occurred, and where the source of the problem lay. If each test took place with a randomly-generated part, understanding what constitutes correct measurement would have been more difficult. The challenge is therefore to select a part with enough features to cover what is required by most manufacturers, simple enough to machine easily.

VIII. Recommendations

Practical experience with the I++ Test Suite and the series of public demonstrations has led to some recommendations for others who are undertaking similar validation efforts.

- Pre-testing components with simulated "mates" uncovers many simple errors that can be fixed early, saving time at the more expensive public demonstrations or installations on plant floors.
- Misinterpretation of specifications by people is to be expected. Formal methods of describing syntax and if possible semantics are preferred over natural language, especially when the audience members do not all speak the natural language natively.
- Examples should be provided where possible. Forgo the temptation to write all examples in the same style. For

example, if the specification allows variations in white space, examples should show this variation.

- Where the specification is ambiguous, expect that two developers will each interpret it differently. In cases where the resolution is a choice between two arbitrary options, each vendor will argue that their choice is the right one. There must be an arbiter whom all parties agree has the final word, and everyone must be prepared to go back to their benches and change.
- Standards validation is expensive, and should include line-by-line reading of the specification by experts; ongoing meetings to discuss revisions to the specification; development of testing tools to be shared by all participants; and commitment to a series of public interoperability testing under real-world conditions.

REFERENCES

[1] S. Kemmerer, Editor, "STEP: The Grand Experience." NIST Special Publication 939, July 1999.

[2] Consortium for Advanced Manufacturing - International, "Dimensional Measuring Interface Standard," Revision 3.0, ANSI/CAM-I 101-1995.

[3] International Association of CMM Vendors, "I++ DME," Version 1.5. Available: www.isd.mel.nist.gov/projects/metrology_interoperability/specs/idmespec.1.5.pdf

[4] J. Horst, T. Kramer, J. Falco, W. Rippey, F. Proctor and A. Wavering, "User's Manual for Version 3.0 of the NIST DME Interface Test Suite for Facilitating Implementations of Version 1.4 of the I++ DME Interface Specification," October 4, 2002. Available: www.isd.mel.nist.gov/projects/metrology_interoperability/NISTI++DMEtestSuite3.0UsersManual.pdf