NISTIR 5879

Conformance Testing and Specification Management

James A. St. Pierre
Kevin G. Brady
Electronics and Electrical Engineering Laboratory
Electricity Division

S. L. Stewart
Manufacturing Engineering Laboratory
Manufacturing Systems Integration Division

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Gaithersburg, MD 20899-0001

September 1996

TABLE of CONTENTS

LIST of FIGURES

Executive Summary

The final report for the second year of a joint project between the National Institute of Standards and Technology (NIST) and SEMATECH follows our previous report [1] and elaborates on two areas:

- Conformance Testing
- Specification Management

The growing acceptance of Java, an object-oriented programming language, makes it feasible to consider testing across heterogeneous computer platforms, as well as remotely across a network. The use of Java and the Object Management Group's Common Object Request Broker Architecture (CORBA) for developing a conformance testing environment is only one way that SEMATECH member companies may benefit from this distributed object paradigm. Although there are still technical issues, such as, performance and security, the trend towards adoption by the industry is currently the biggest advantage that Java has over other object-oriented languages.

We recommend that SEMATECH continue to explore automatic test generation techniques. This is apart from the proposal for Java-based testing but would be complementary and should be done in conjunction with Java-based testing.

The Internet has long been used in the computer research community for exchanging documents, specifications, and computer software. More recently, the development of Hypertext Transfer Protocol and the Hypertext Markup Language have created the World-Wide Web. We feel that it is more important than ever to consider the adoption of HTML as a standard publication format, and even as an exchange and working format. This can be done with conventional word processing tools with HTML capability or with specialized HTML-based tools. The reorientation to Web publication and HTML will require some reorganization of the Specification text.

1. Introduction

This is the final report for the second year of a joint project between the National Institute of Standards and Technology (NIST) and SEMATECH under a Cooperative Research and Development Agreement (CRADA) between the two organizations. A year ago, we submitted our first report [1]: *Roadmap for the Computer Integrated Manufacturing Application Framework*. We are pleased to note that a number of our recommendations from that report have already been acted upon, and in some cases going beyond what we expected. The *Computer Integrated Manufacturing (CIM) Application Framework Specification 1.3* [2] has been made public in electronic form, it has been converted to a more accessible word processing format, and it has been divided into more manageable subdocuments for rapid evolution. SEMATECH has, to its credit, gone beyond our recommendations by adopting a networked based document management system (Lotus Notes) that will support a much larger and distributed group of people participating in the further development of the CIM Framework.

During the last year we have concentrated on two areas: conformance testing and specification management. Both topics are extensions of last year's work with more attention to the latest technological developments. This year we discuss how Java might be used to support testing of individual components of the CIM Framework. When we reported on conformance testing in last year's report, we never mentioned Java because we had not anticipated how quickly it would become widely available. Now, we discuss the application of this new technology to conformance testing of the CIM Framework.

We made a case for single-source specification management last year. This means that the CIM Framework Specification is maintained in a single-source format from which all the necessary distribution forms can be derived as automatically as possible. When paper publication was the only distribution form, the only issue was version control as changes were made to the document. Now documents must be distributed in paper, on the World-Wide Web, and the formal specification portions must be suitable for object request brokers (ORB's). We discussed the role that Hypertext Markup Language (HTML) might play in this process. However, this last year has seen remarkable growth in this Internet technology. In this report we update our recommendations to track those changes.

The World-Wide Web, also known as WWW or simply just, the "Web," has taken a central role in our work. It is the infrastructure that makes electronic publication of documents like the SEMATECH CIM Framework Specification practical. One of the consequences of this rapid change to Internet-based technology is that the references in this paper include universal resource locators (URL's), the keys to finding information on the Web. This paper can be found at **http://www.cme.nist.gov/msid/pubs/stew96b/** in its HTML version. Two other significant technologies in our work are also based on the Internet and converging with the Web: the Java programming language and object request brokers (ORB's).

This work was carried out at NIST by the authors. The authors wish to thank Ed Barkmeyer, Neil Christopher, Michael McCaleb, Michael McLay, and Evan Wallace for important contributions to the evolution of this work.

This work was supported by SEMATECH contract #95101037 OP and by NIST under its Scientific and Technical Research and Services budget.

2. Conformance Testing Using Java

2.1 Overview

According to a Java overview on one of SUN Microsystems Web pages [3]:

> Java is a simple, robust, object-oriented, platform-independent multi-threaded, dynamic general-purpose programming environment. It's best for creating applets and applications for the Internet, intranets and any other complex, distributed network.

One of the major advantages of HTML documents is that they can contain hyperlinks which provide access to additional information about the original text. Java is intended to extend the WWW to include the capability of downloading software applications (or applets) in addition to HTML documents. One of the primary benefits of Java is that it will allow applications to be written once and run on any platform that supports a Java execution environment. Conformance testing methodologies have always been faced with the difficulty of providing a test harness that is portable across a wide variety of heterogeneous platforms. Java solves this problem by running in an interpreted environment. Java programs can be developed either as an applet (runs in a Web browser) or an application (runs as a stand-alone application in a Java environment). Although Java programs run in an interpreted environment there is a compile step. Sun currently provides a free Java compiler for Solaris, Windows 95, Windows NT, and the Macintosh platforms. Development for other platforms is in the works.

The major platform for which there is not currently a Java capability is Windows 3.1. One reason for this is the lack of threads in a Windows 3.1 environment. However, Netscape has announced [4] plans to support Java for Windows 3.1 in a future beta release.

It should be noted that along with the power of being able to download an application from the Internet and run it on your local machine, there are inherent dangers involved in this architecture. There is work underway to ensure that the Java environment is a controlled one in which the users can decide upon the level of access that a Java application should be given to their local computer systems. This report does not focus on the security issues involved with Java, but the conformance testing work (in addition to all other Java related work) will rely heavily on the ability of the developers to provide a safe Java environment.

2.2 Approach

This initial proof-of-concept project will only provide tests for a small portion of the CIM framework specification. A complete system will contain a full set of tests, which could be developed with combined automatic and manual test generation techniques, see [1].

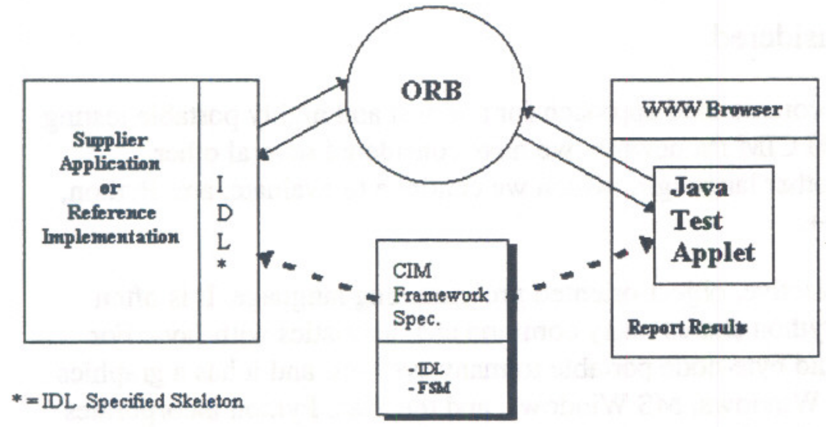

Figure 1. Conformance Testing Environment Using Java and CORBA

As indicated in Fig 1. the Java-scripted test and the suppliers implementation are both based on the CIM Framework specification. A typical scenario will involve a developer of a CIM Framework compliant application *e.g.*, a scheduler application, accessing a Web site which contains the Java test suite. The supplier will be presented with a form to fill out indicating which tests they wish to run. An *a la carte* menu will allow developers of individual classes to pick tests for their own classes. In addition, test suites for a component of the CIMF or groups of components could be selected. Once the tests are selected the next step will be for the user to fill in information about the ORB to be used during the testing, via an HTML form interface, or a Java AWT (Abstract Window Toolkit) interface. This information will be required to determine the location (name or address) of the ORB which will be used to connect the supplier's implementation under test with the test itself. Note that in some configurations multiple ORB's might be used to communicate between the test and the object under test. The address of the ORB will then be passed to the test applet or application before it is downloaded to the user. The first thing the applet will do when it is actually downloaded by a user will be to register itself with the local ORB. The ORB could either be local to the supplier's machine or it could be remote. For the proof-of-concept testing we are planning, all three of the main elements of the testing system (ORB, implementation, and test) will reside on the same machine. In some cases it may be that the local ORB that the supplier is using may not have an interface to Java. In this case another ORB (which supports an interface to Java) will be used and the inter-ORB communication will be required to actually run the test.

The test itself could either be a low level test for a given class, for example it might exercise one of the finite state machines described in the CIM Framework specification. Or, the test could actually be a test script that exercises the implementation under test by simulating some series of steps within a semiconductor manufacturing facility. In the case of a low-level object, a factory simulator may or may not be required depending on the objects being tested. In this scenario, the test itself will provide the emulation of the factory. If a given object is waiting for an event to occur before proceeding to its next state, the test will be required to generate that event in order to force the object (or system) under test to continue. As the test executes it will indicate status via the Web browser from which the test was initiated.

2.3 Other Technologies Considered

In the process of developing a recommended approach for a robust and highly portable testing environment for the SEMATECH CIM framework, we have considered several other programming languages. These other languages, which we continue to evaluate, are: Python, Tcl/Tk, Perl, Visual Basic, C/C++.

Python [5] is an interpreted, interactive, object-oriented programming language. It is often compared to Tcl, Perl, or Java. Python shares many common characteristics with Java. For instance Python is source-code and byte-code portable to many systems, and it has a graphics module that is portable across X Windows, MS Windows, and the Mac. Python incorporates separate name spaces for imported modules and includes exception handling and garbage collection.

Some of Python's disadvantages are that it does not directly support static typing. Static type checks can be done in C. Exceptions can then be raised in Python if the C type violates the static type checking requirements. Another disadvantage of Python is simply that it does not have as much industry support or documentation as Java has. A Web browser called Grail is available which will run Python applications; however, this would require suppliers to install a second browser, the Grail browser, if they were not currently using it.

One of Python's advantages over Java is that it is a higher-level language in which everything is a first class object. Also, Python has the flexibility of being a dynamically typed language, it can be operated in an interactive mode, and it provides full access to metadata. In addition Python has built-in generic container classes for lists, tuples, and dictionaries (also known as associative arrays in Perl). Python has extension modules to support many things that will be required to conduct CIM Framework testing. This includes an Internet Inter-ORB Protocol (IIOP) interface for communicating with a CORBA ORB. There is a second interface under development that will use CORBA's dynamic invocation interface (DII). Since Python is dynamically typed this should allow Python to access CORBA objects directly. There will be no need to compile a special module for each object interface.

Technically, the Python language is a viable alternative to using Java for the entire testing process; however, since it is not as widely used and supported, it is not recommended as the sole language on this project. For this reason its role on the project will be limited to those tasks for which Python provides a capability that is not available from Java.

Tcl/Tk (tool command language/tool kit) [6] refers to Tcl, which is an embeddable scripting language, and Tk, which is a graphical user interface toolkit based on Tcl. Both packages are freely available. The Tcl/Tk project at Sun Labs (headed by John Ousterhout) is leading the development of Tcl and Tk and building the infrastructure to use them as a universal scripting platform for the Internet. Tcl/Tk allows programmers to implement user interfaces quickly for their applications. Although Tcl/Tk runs on many platforms it is not focused on solving the same problems as Java. Instead Tcl/Tk provides an easy-to-use technology for developing user interfaces and small scale applications. John Ousterhout has described the relationship between Tcl/Tk and Java by comparing it to the relationship between Visual Basic and C++, "C++ and Java are system programming languages, and Visual Basic and Tcl/Tk are scripting languages. In the 21st century, Tcl/Tk will be to Java what Visual Basic has been to C++."

Perl [7] is a very popular and widely used scripting language. It is one of the most widely used languages for writing common gateway interface scripts (cgi-scripts) on the WWW. These cgi-scripts are programs that run on Web servers and handle the data input to HTML forms. Perl has been described as being somewhere between Unix shell-scripting languages and C programming. Perl is easier to program than C, but like C it can be written in a very concise and cryptic form. An advantage of Perl is that it is interpreted rather than compiled. Like Java, it eliminates pointers, which greatly simplifies the language. It should be noted that as with most of the other languages mentioned in this report, Perl runs on most major platforms; however, it does not provide the same type of Web client based execution environment that Java does.

Visual Basic is primarily a Windows tool used to develop user interfaces and small applications. A recent development from Microsoft is their Visual Basic Script that allows developers to link and automate a wide variety of objects in Web pages, including ActiveX controls and Java applets. One of Microsoft's WWW pages describe ActiveX as follows [8]: "ActiveX is a term used to refer to a broad range of client/server technologies from Microsoft that are designed to increase the dynamic designs of a Web site." There is also work ongoing to integrate Microsoft's ActiveX technology with Java. This will allow developers to write Java applets that interface to ActiveX controls. Currently ActiveX components require Microsoft's Internet Explorer 3.0 Beta 1.

The use of C/C++ was also considered, since C programs will run on virtually any computer. The main drawback to C and C++ is that there is currently no capability to download and automatically run C or C++ programs in a controlled environment as there is with Java.

Again it is important to point out that the benefits of using the testing architecture described in this report are heavily dependent on the portability of Java. If Java fails to realize its promise of providing platform-independence for applications, then this concept of providing highly portable conformance testing will be somewhat less fruitful than predicted. However, it should be noted that Java is similar enough to C that there are tools which will translate Java to C; therefore, it should be relatively easy to convert tests developed in Java to C, if so desired.

In addition to the advantages of Java listed above, there is much ongoing work with Java and its interface to CORBA objects. Also, as noted earlier, there is a significant industry interest in Java. One likely result of this will be a large pool of knowledgeable Java programmers as well as general industry support for the language. As with some of the other languages we discussed, Java can be used to develop user interfaces that will run across heterogeneous platforms. Having an easily accessible, highly portable, and extensible conformance testing environment will provide significant cost savings to the SEMATECH member companies. Such an environment will also help semiconductor suppliers develop higher quality products that are conformant to the CIM Framework. Because of the many potential benefits to the SEMATECH member companies, Java and CORBA have been selected as the primary technologies for this project. Although other languages may be used in the implementation of this project, the primary goal is to evaluate the feasibility of using Java and CORBA technologies to provide a conformance testing environment for the SEMATECH CIM Framework.

2.4 Possible Future Work

Combining automatic test generation tools with the Java/CORBA conformance testing environment described in this report would be a useful follow-on project to this work. This initial work is focused on the conformance testing environment for developers of applications which conform to the CIM Framework Specification. This work could be extended to include mechanisms for certifying, through the use of digital signatures, for example, that official versions of the test suite were run against a given object under test. The test could also generate a signature for the object being tested so that in the future one could be easily verify whether the application a customer is running is indeed the one that has passed certification. Having an automated mechanism for certifying that an application is conformant will save the money normally required to pay for certifying officials to personally verify the execution and performance of a given implementation.

It should be noted that there is a lot of activity ongoing to allow various programming languages to run in an environment similar to the Java environment. For example the Grail environment for Python, Microsoft's efforts with the ActiveX objects, and IBM's Rexx [9], NetRexx, and Object Rexx languages are other industry examples. Future work may involve the integration of several of these and other languages/environments in a conformance testing and certification environment, which draws on the strengths of each technology.

3. Single-Source Specification Management

3.1 Overview

Following up on last year's report, one task for this year has been exploring the problems of maintaining complex documents, specifically specifications, in a single format with multiple uses and multiple distribution formats. We introduced the notion of Single-Source Specification Management (S3M). To some people this may seem to be a dull—or even unimportant—problem, but through many years of working with information-intensive computing projects we have learned that it is a difficult and critical issue. The rise of desktop computing power has brought many new tools for document preparation, but there has not been a corresponding advance in managing the resulting complex documents. The S3M task is looking at pragmatic solutions to the problem of maintaining complex documents with multiple distribution formats while using combinations of readily available software. We are trying to come up with the best partial solution that can be applied now rather than an ideal solution based on research software or a theoretical solution based on untested ideas.

3.2 Document Formats

The first question is to understand the different kinds of document formats and see how each serves specific needs in the overall document management process. We have identified three kinds of document formats:

Working formats

The file storage formats native to specific word processors or editors (collectively referred to as *tools*). They are usually **proprietary** to a specific tool and version of the tool. As a result, these formats are usually named for the tool/version or the associated file extension. The idea of a working format is used by other tools, like presentation graphics, spreadsheets, and simple database applications.

Some examples are: MS Word 6.0 (DOC), WordPerfect 6.1 (WPD), Framemaker 5 (fm or doc).

Exchange formats

These file formats are designed for transferring documents from one tool to another. Their success depends on the prevalence and adequacy of their implementation. A candidate exchange format must be widely adopted (become standard) and well implemented (correctly transfer most of the document's structure and information) to be useful.

Two limited examples are: Rich Text Format (RTF) and ascii text (TXT).

Distribution formats

These are final output formats, that is, electronic formats that are suitable for printing or browsing. Hypertext Markup Language (HTML) [10] is the most important example for publishing on the Web, but ascii text is also widely used over the network because it is simple and, until the spread of HTML, it was the only format that could be read on any computing platform.

The best known examples for producing paper copies are the printer driving formats, like Adobe PostScript (PS) and Hewlett-Packard PCL. Adobe has also defined Portable Document Format (PDF), a more modern format adapted to electronic browsing but capable of being printed. Its predecessor, PostScript, was designed for printing, and adaptable to browsing only with special viewers, *e.g.* GhostView.

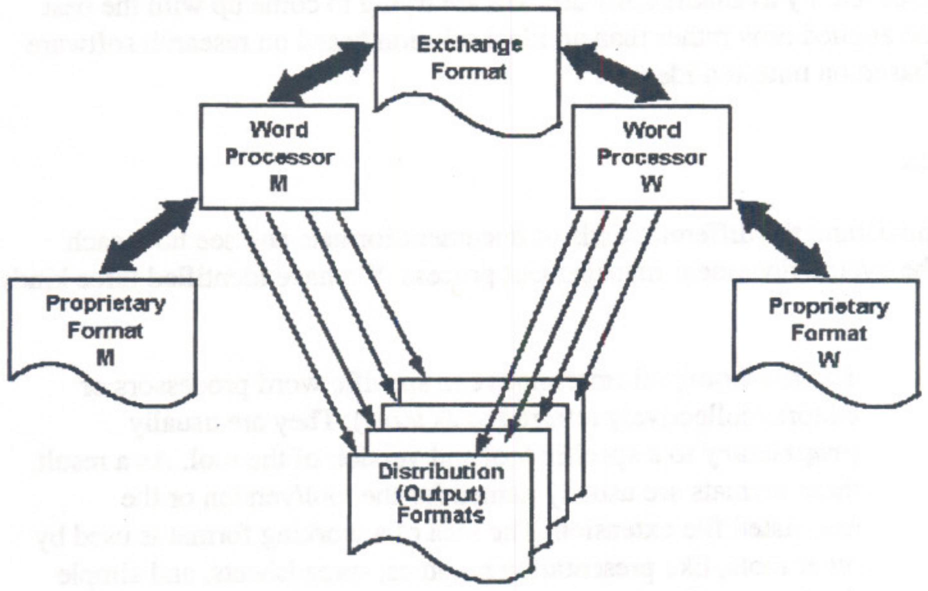

Figure 2. Electronic Document Formats

With the right tools, any format can be pressed into service in a different class. Normally a word processing tool has printing functionality; so, its working format could be viewed as an output format if one has the right word processor. Likewise, a working format serves as an exchange format for any group of people using the same word processor. Distribution formats, for printing, are largely limited to the output function because the transformation from working format or exchange format is not easily reversible in practice. This irreversibility property has some potential advantages in the protection of intellectual property rights, but right now this is largely a side-effect.

HTML and ascii text are special cases. They are equally useful as working, exchange, and electronic distribution formats. Ascii text has been used these ways for many years with text editors and simple printer drivers. Much more recently, we have seen word processing tools that either work directly in HTML or can import and export it. As the capability of handling HTML becomes more wide-spread in commercial word processors, HTML will become more useful as an exchange format. As an example, this document was prepared using an HTML editor, SoftQuad's Hotmetal, so that it would be immediately available for publication on the Web. Then the HTML version was read into Microsoft's Word/Internet Assistant to produce the hardcopy version. This demonstrates the point of the task: a single source in HTML provided both electronic and hardcopy results.

There is an important case of working formats serving as exchange formats. In highly competitive markets, like DOS and Windows, it is an advantage to be able to import competing working formats. So, it is common for these word processors to make some attempt to read (import) other working formats, most especially their own earlier versions. They are most successful, of course, at reading their own earlier versions, although with problems more often than a user would like. When we look at reading competitors' formats, we find that the degree of success is often quite low for anything more than the simplest of documents. As one would expect, a new version of a word processing tool is limited to the existing versions of the competitors' formats. So, there is a never ending round of updating.

3.3 Electronic Distribution Formats

Any electronic documents created for others to read should be presented in one or more of a small number of formats that are nearly universally readable. This can, and should, be done regardless of the document processing application that was used to edit the documents originally. These distribution (or output) formats are, in order of desirability:

1. Hypertext Markup Language (HTM/HTML)—good control of appearance and page layout but can be affected by browser settings. It includes hypertext (links to other Web documents) and image capability and is universally readable with all browsers, if standard HTML is followed.
2. Portable Document Format (PDF)—excellent (nearly perfect) control of appearance and page layout. Hyperlinks are not yet practical. Graphics in the document are perfect. And readers are freely available for PC's, Mac's, and Suns (and other platforms).
3. Ascii Text (TXT)—appearance is single font and size while page layout is minimal. There are no hyperlinks and no graphics but it can be read and printed universally. For simple documents that do not merit a lot of work on their appearance, text is often the easiest and best alternative. If you feel comfortable sending the message through email, without feeling that you have lost something important in the format, then text is fine for posting on the Web.
4. PostScript (PS)—Some systems, like Mac's and Suns, produce PS output almost exclusively, and PS printers can always print it, but if you are outside that environment PS can be problematic. With special software, PS can be viewed directly from a workstation, but it is most practical when used to produce hardcopy with a PS printer.

Note that HTML and ascii text files are easily indexed by search engines whereas PDF and PS files are not.

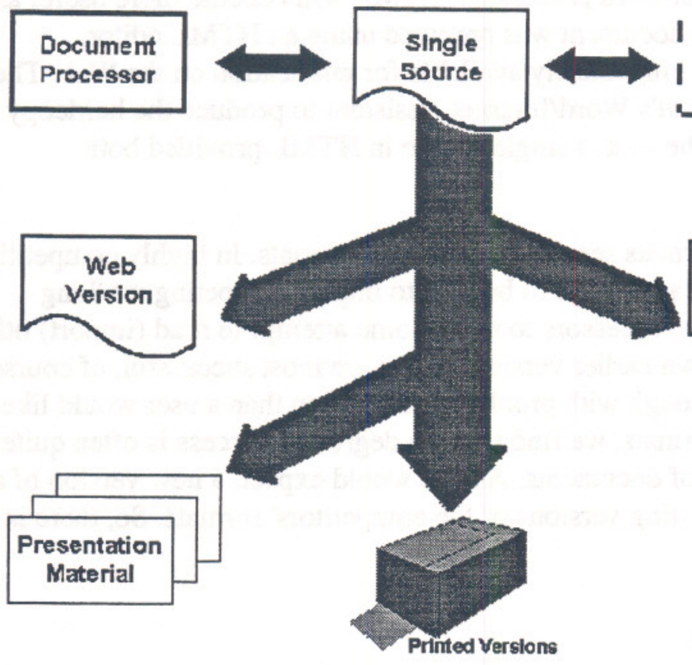

Figure 3. Using a Single Source With Multiple Distribution Formats

Which ones should be used? If an HTML document is available, that is the first choice. If ascii text is satisfactory for the document, then it is sufficient. If a PS file is available, we recommend that it should also be made available in PDF (easy to do). Other documents, which cannot be easily converted to HTML, should be distributed in PDF. For complex documents in HTML, which involve many linked pages, it may be worthwhile to consider a supplemental PDF version that can be viewed or printed as a single document. It may also still be desirable to produce a PS version when you produce a PDF version, for those people in the Mac/Sun environment who are not using Acrobat. This is not difficult to do. Adobe's PDF with the Acrobat Reader is far superior to Adobe's PS with GhostView. One can create PDF from PS or create native PDF files from any Windows PC application. When distributing HTML documents in another format, it is useful if the URL's are preserved in the text, for example, in brackets.

3.4 The Exchange Question

The question of document exchange can be stated precisely in terms of the requirements that must be met to convert a document from one working format to another with minimal loss of content or structure. This is an important question because we often need to revise a document prepared by someone with a different word processor. The question can be answered in any of three ways:

Exchange Formats	These are formats that can be read and written by many word processors. This is a classic solution used in many computing

applications where a transformation of information must be made among many different applications. Each application is designed to read and write its proprietary format and one common (exchange) format. If the exchange format is open and in a publicly available specification, then anyone can write tools to work with that format. Such tools are not limited to other word processors but can include tools of all kinds to index, search, reorganize, and much more.

Import/Export This a the capability to read or write other widely used working formats. For this to be a solution to the exchange question, each word processor (and each version) would have to be able to read most other working formats. some users could be dealing with three different word processors, each with a current version and a previous version with distinct working formats. For importing to be a complete solution, each of the six tools would have to implement one writer (for its own format) and six readers for all the formats. That makes 42 transformations that must be programmed. This is in contrast to 24 transformations that must be programmed for the exchange format solution (two readers and two writers for each of the six tools).

Conversion There are a number of commercial and freeware products specifically designed to convert from one format to another. Some try to handle a large variety of formats. These are usually commercial products and are often limited to the formats associated with one platform. Others are designed to handle certain commonly found conversions between two formats. Freeware and commercial products are both common. The same approach is found for other files formats, especially graphics formats.

3.5 Format Conversion

The goal of format conversion is to put documents into a single common format that can serve all the diverse needs discussed above. We conducted a number of tests with Microsoft Word 6.0c (including the Internet Assistant), Novell (now Corel) WordPerfect 6.1 (including the Interactive Publisher), and several HTML editors. One of the authors (SLS) also participated in another series of tests to select a common format for his division. Those tests included Adobe Framemaker 5.0 in addition to the two word processors already identified. All three could create HTML documents, but only Word with the Internet Assistant could also import HTML. This capability of reading HTML documents is sufficiently important that it is the basis for the near-term recommendation. In addition, this recommendation is compatible with the current SEMATECH word processing environment.

We understand that the latest versions of Word and WordPerfect for Windows 95 will provide more features for electronic publication with HTML, but they were not available for testing in our environment during the study period. We expect this technology to advance rapidly; therefore, many products may provide the necessary features in the future. Another possible approach is to define a new Standard Generalized Markup Language (SGML) document type definition (DTD). SGML is an international standard (ISO 8879) for defining document structure. HTML is a special case of an SGML DTD; so, this amounts to designing an alternative to HTML. These are all areas for future study and evaluation.

As a result of those tests, we have three levels of recommendation:

1. **Near-term:** Use Word with the Internet Assistant and stay within the HTML template as much as possible.
2. **Middle-term:** Use HTML once the definition of HTML is rich enough to support the Specification and related documents and there are tools to support it.
3. **Long-term:** Explore an SGML document type definition tailored to the needs of the CIM Framework project.

The definition of HTML and the quality of tools that can use HTML are expanding so rapidly that there may be no need to go to the more sophisticated solution suggested in level 3. If the Specification needs more content tags than HTML provides, then defining a new DTD based on HTML is an alternative.

We have been amazed at how rapidly HTML features have spread to conventional word processing and how many stand-alone HTML tools have been introduced in the last year. In our first report, we could only recommend exploring HTML. Now we believe that level 2, using HTML as the format for the single source is very nearly feasible. We recommend that developments in this area of technology be tracked closely. The potential for improved document management particularly for Web documents is great.

The CIM Framework Specification, like many technical documents, relies heavily on graphics. The standard format for Web publication is GIF. There are now many tools for converting other graphics formats into GIF, but the problem is that the graphics are often embedded in the original electronic text. Based on our experience, we recommend that all graphics be separated from the text and managed as independent files that are linked to the base document. All graphics are linked when using HTML with the use of image (IMG) or anchor (A) tags. By separating out the graphics, the documents will be ready for full use of HTML. We have found that GIF files can be linked to Word files as easily as to HTML files.

While we have not made any specific recommendations about the important issue of configuration management and version control of complex documents, SEMATECH's decision to adopt Lotus Notes to support this function, as well as others, is a sign of good practice. We do feel that it would be impractical to manage such an extensive and distributed project without networked, interactive tools.

4. Conclusions

Two tasks were covered in this report. The first studied the potential for using Java in conformance testing, and the second task covered preparation and dissemination of specification documents in both electronic and paper formats from a single source.

The key benefit of Java is its ability to run on heterogeneous computer platforms. This capability has been long sought in the computing industry, and its feasibility has been demonstrated several times with platform-independent interpreted languages and semicompilers. Wide acceptance of Java will make this approach practical. The use of Java and CORBA for developing a conformance testing environment is only one way that the SEMATECH member companies may benefit from this distributed object paradigm. Although there are still technical issues—such as, performance and security—related to Java, the trend towards adoption by the industry is currently the biggest advantage that Java has over other object-oriented languages.

We recommend that SEMATECH continue to explore automatic test generation techniques. This is apart from the proposal for Java-based testing but would complement it nicely and should be done in conjunction with it.

With the extraordinary acceptance of HTML and the Web by the technical community, and even the public, we feel that it is more important than ever to consider the adoption of HTML as a standard publication format, and even as an exchange and working format. This can be done with conventional word processing tools with HTML capability or with specialized HTML-based tools. The reorientation to Web publication and HTML will require some reorganization of the Specification text. The plan to partition the text so that individual focus teams can work on components of the Specification independently will also help make the text more manageable on the Web. We strongly support that approach.

References

All Web references (URL's) were verified for existence in July, 1996.

[1] S. L. Stewart and James A. St. Pierre, *Roadmap for the Computer-Integrated Manufacturing Application Framework*, NISTIR 5679 (1995) also SEMATECH 95052825A-ENG. Web reference [http://www.cme.nist.gov/msid/pubs/stew95a/]

[2] Lawrence Eng, *et al.*, *Computer Integrated Manufacturing (CIM) Application Framework Specification 1.3*, SEMATECH Technology Transfer #93061697F-ENG, Austin TX (1996). Web reference [http://www.sematech.org/public/cim-framework/home.htm]

[3] Java Web reference [http://java.sun.com/allabout.html]

[4] Netscape Java Web reference [http://home.mcom.com/comprod/products/navigator/version_3.0/enhance/index.html]

[5] Python Web reference [http://www.python.org]

[6] Tcl/Tk Web reference [http://www.sunlabs.com:80/research/tcl/]

[7] Perl Web reference [http://www-cgi.cs.cmu.edu/htbin/perl-man]

[8] Microsoft ActiveX Web reference [http://198.107.140.2/wwlive/html/activex.html]

[9] IBM Rexx Web reference [http://rexx.hursley.ibm.com/rexx/rexx.htm]

[10] World Wide Web Consortium Web reference [http://www.w3.org/pub/WWW/MarkUp/]