Roadmap for the Computer Integrated Manufacturing Application Framework

by

S. L. Stewart

James A. St. Pierre

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

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Executive Summary

The final report for the first year of a joint project between the National Institute of Standards and Technology (NIST) and SEMATECH lays out a roadmap for adoption and use of the SEMATECH CIM Framework that includes:

- Developing a Specification
- Reaching Consensus
- Standardization
- Testing and Certification

This report makes the following recommendations, in priority order:

- Adopt a single source electronic specification management approach
- Increase supplier involvement in both specification and certification development
- Give reference implementation high priority
- Use the Object Management Group (and its Manufacturing Special Interest Group) for promulgating the specification
- Reach consensus on certification business model and methodology
- Address the high cost of certification
- Develop usage scenarios to clarify the implementation and use of the CIM Framework
- Reconcile the scope of the specification with CORBAfacilities and CORBAservices and with other related projects
- Do not make certification dependent on access to suppliers' source code
- Focus on interoperability
- Expand use of formal description techniques in the specification
- Explore automatic test generation

In addition, there are numerous technical recommendations throughout the report.

1. Introduction

This is the final report for the first 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. The results of two statements of work are covered in this report: (1) Generalization, Standardization, and Promotion, and (2) Conformance Testing and Certification. The work was carried out by a team at NIST led by the authors and including Neil Christopher, Elizabeth Fong, Barbara Goldstein, Greg Koeser, Tom Kramer, Michael McCaleb, Michael McLay, Steve Osella, and Evan Wallace. We also want to acknowledge valuable discussions with John Barkley, Ed Barkmeyer, Kevin Brady, Tony Cincotta, Barbara Cuthill, Martha Gray, Shirley Hurwitz, Arnold Johnson, and Tom Rhodes. Valuable technical support was provided by Joe Chandler.

This report includes a roadmap to adoption, use, standardization, testing, and certification of the Computer Integrated Manufacturing Application Framework (CIM Framework) developed by SEMATECH. Recommendations included in this report have been presented to SEMATECH, and are often based on SEMATECH information, but they are recommendations from the NIST project for the future direction of the joint project. Their publication in this report should not be taken as endorsement or acceptance by SEMATECH.

Background material and supporting documentation are contained in appendices, which are available separately.

Certain commercial products are identified in this paper. These identifications are for clarity of presentation only. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the products are among the best available for the purposes they serve.

This work was supported jointly by SEMATECH contract #34008401 and by NIST under its Scientific and Technical Research and Services budget.

2. Standardization

2.1. Background

SEMATECH developed a framework for CIM applications, the CIM Application Framework, based on work by Texas Instruments (TI), a member company, in the Microelectronics Manufacturing Science and Technology (MMST) project. The goals are to promote integration on the shop floor, reduce costs, and increase reuse through object-oriented technology. The CIM Framework is based on the Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA). In particular the specification of the framework uses the OMG interface definition language (IDL) to define the classes (interfaces) of the framework. In the current specification, version 1.1, there is further specification of the interfaces in terms of Harel state charts and Rumbaugh diagrams, as well as English narrative.

2.2. Roadmap

A roadmap to standardization for the CIM Framework goes through several stages. The analogy to a roadmap is only loosely true because the stages are overlapping and most of the activities need to be carried out in parallel. However, the emphasis and level of effort will shift as we progress through this process. The principal stages are: specification, consensus, standardization, and testing/certification. Each of these stages is expanded below.

2.2.1. Specification

The logical first step in developing a standard framework is to create a specification. For the CIM Application Framework, this process started with the MMST project at Texas Instruments and is now being carried out by the Manufacturing Execution Systems (MES) Build Team at SEMATECH. The first version was published by SEMATECH on March 31, 1994, as *Collaborative Manufacturing System Computer Integrated Manufacturing (CIM) Application Framework Specification 1.0*. This was revised, and version 1.1 was published on August 31, 1994. A second revision (1.2) is soon to be published, and version 2.0 is under development.

Our project took the electronic version of the original specification and converted it into a HTML document suitable for online browsing, that is we made it a World Wide Web readable document. HTML, short for Hypertext Markup Language, is the specialization of SGML (Standard Generalized Markup Language, ISO 8879) used by Web servers for formatting compound documents. We subsequently updated the online version to 1.1. In so doing, we demonstrated the feasibility of making the specification available in browsable, electronic form without having to distribute the original electronic document. John O'Connor and Fred Waskiewicz, from SEMATECH, were instrumental in making this conversion possible.

We recommend that this process be carried even further by planning for a future version of the specification in which one electronic source can be used to create three different forms of the specification: (1) the hardcopy version for printing, (2) an HTML version for online browsing, and (3) an extract of the formal portions of the specification for computer processing. We refer to this concept as single source specification management.

2.2.2. Consensus

To achieve SEMATECH's objectives, it is not sufficient to produce a specification, even one of technical excellence. There must also be widespread agreement among both the suppliers and users of manufacturing software that applications should be based on the specification. This consensus is a necessary step in the road to adoption and success.

SEMATECH has already involved the users' groups from its member companies in the process of developing the specification. It is also contacting independent suppliers and providing orientation and training about the CIM Framework in scheduled classes and public conferences. We believe that this process of awareness, involvement, and training is absolutely essential to the success of the CIM Framework, and we recommend that it be continued and expanded to the limits of the resources available.

2.2.3. Standardization

Standardization is the next step beyond consensus; it records the consensus in a well-defined and public way. Standards can be promulgated by national and international standards bodies or by groups of interested parties or by companies through widely used products.

The American National Standards Institute, a non-governmental organization, is the U.S. national standards body; however, many of its standards are developed by accredited standards development organizations (SDO), like the Institute for Electrical and Electronics Engineers (IEEE). At the international level there are several standards bodies, for example, the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). These bodies develop standards through technical committees of volunteer experts, but the final adoption is by ballot of the member countries.

For semiconductor manufacturing, the Semiconductor Equipment and Materials International (SEMI) organization conducts an international standards program for its members, which can play a significant role in standardizing the semiconductor-specific portions of the CIM Framework.

Recently, there has been increased use of other kinds of organizations to develop standards in information technology where the pace of technical development is faster than traditional standards-making procedures can accommodate. Typically a consortium or similar organization will be formed to develop a specific technological area where consensus on standards is essential to creating the market for the new technology. One important example for the CIM Framework is the Object Management Group (OMG). It was organized in 1989 to develop an Object Management Architecture (OMA) and a Common Object Request Broker Architecture (CORBA).

SEMATECH is a corporate member of OMG and has committed to using CORBA as the basis for binding CIM Framework-conformant applications to a computing infrastructure. The formal syntax for the interfaces is specified in CORBA Interface Definition Language (IDL). It is these IDL interface specifications that comprise the computer processible portions that would be extracted from the single source specification recommended above. These same specifications can be the basis for submission of the CIM Framework to the OMG Technical Committee (TC) as part of a vertical market CORBA common facility for manufacturing.

The mechanism for submission is already available in the Manufacturing Special Interest Group (MfgSIG) of the OMG/TC. SEMATECH was instrumental in revitalizing this SIG in August of 1994 and provides the chairman, Fred Waskiewicz. Our project has actively assisted this process by providing support for the work of the MfgSIG.

After studying the CIM Framework specification, we believe that much of it is sufficiently general to serve as a basis for an object-oriented manufacturing framework to meet OMG’s needs. The MfgSIG is already studying this issue. In part, we believe that this generalization is possible because the object class structure lends itself to more general superclasses with class specialization for more specific applications. Thus many, if not most, of the higher-level classes in the CIM Framework are not semiconductor or even electronic manufacturing specific.

Another benefit of the electronic version of the specification was illustrated by Paul McGuire, of SEMATECH, who was able to generate a complete spreadsheet-based cross-reference to the class definitions. This is useful for analyzing and checking the class definitions.

Working with OMG has several advantages. First, it is necessary to participate in the OMG/TC in order to stay current with CORBA technology. Second, OMG has an active liaison process with other standards’ organizations, particularly the international ones, like the ISO technical committees. Third, through its more than 500 members, OMG provides access to a major segment of the object-oriented information technology supplier community. This will help develop involvement and support from more suppliers beyond the semiconductor manufacturing community.

The OMG liaison process is especially well suited to carrying out a complete standards program. Basically a liaison is established by an organization that participates in both OMG and the candidate standards group. As such, the liaison document must meet the requirements of both groups, but to date this has not been a difficult requirement.

In our roadmap to standardization, we see three or four potential liaisons: (1) with SEMI for semiconductor-specific standards, (2) with IEC/TC93 (Design Automation) for more general electronic standards, (3) with ISO/TC184/SC5 (Manufacturing Automation/Architecture and Communications) for more general manufacturing standards, and (4) possibly with subcommittees of the ISO/IEC Joint Technical Committee 1 (JTC1) if there are information technology standards that go beyond manufacturing.

Between SEMATECH and NIST, we have extensive working relationships with all these parties. As the specification evolves we should evaluate the opportunity for establishing such liaisons as the mechanism for formal, international standardization. SEMATECH can obtain great leverage through these partnerships.

2.2.4. Testing and Certification

The CIM Framework will not be a success if it is not used, and standardization is only one step toward raising users' confidence to that point. Another important step in the quality assurance process is to have some means of testing implementations for conformance to the standard and a certification process to attest to the results. This topic was a principal study area for this project, and the results are discussed in detail in Section 3 below.

2.3. Significant Issues

In this section we briefly identify a number of issues that can have an important influence on the success of the CIM Framework. In some cases there are specific recommendations. However, we believe that all these issues warrant continued attention for the life of the project.

2.3.1. Supplier Involvement and Support

Earlier we identified supplier involvement as critical to the success of the CIM Framework. Unless the suppliers of manufacturing software adopt the CIM Framework, this work may have great technical value, but it will not bring about the cost and productivity benefits expected. As much as 70% of the cost of semiconductor manufacturing software, and especially the integration costs, are generated by in-house software groups of the manufacturer. So, the semiconductor manufacturer is typically both a user and a supplier. When the manufacturer contracts out the systems integration, then a third party integration company also becomes part of the supplier chain.

This diversity of suppliers becomes even more complicated when the CIM Framework is generalized to a broader manufacturing community where each type of manufacturing may have its own segment-specific software supplier chain. This means that we need to identify and work with trade associations and consortia to reach as many suppliers as possible. At the same time SEMATECH needs to continue educating the user companies as both users and internal suppliers. Often these may be two separate sub-organizations.

2.3.2. Formal Specification

We very much support the use of formal description techniques (FDT) in the specification. The CIM Framework is intended to be a standard for software development, that is, for creating computer programs and their data. Computer programs are, in their own way, the ultimate in formal specification. But, they are too detailed for many kinds of human analysis. Therefore the goal of a framework or other high-level specification is to capture as much of the essence as possible, while suppressing the implementation details and maintaining the benefits of formal description.

The IDL portions of the specification are central to the success of the CIM Framework in the CORBA environment, and they are a good example of the benefits of FDT. However, IDL is only intended to capture the signatures of the method interfaces, essentially a syntactic specification. In order to fully characterize the methods, we need to capture their semantics as well. Lawrence Eng, at SEMATECH, is making a valuable contribution by exploring VDM++ as a semantic specification technique, and we plan to leverage that work into the development of test implementations of applications.

Currently in version 1.1, the semantics are captured in a combination of English narrative, Rumbaugh diagrams, and Harel state charts. While the last two are formal, they are essentially graphical or tabular and not easily converted for automatic computer manipulation. There are many FDTs to choose among, but we will encounter problems when we need to integrate our CIM Framework with the work of other groups that have independently chosen a different FDT. At present it does not seem likely that one FDT will become dominant; so this will likely remain a significant integration issue at the enterprise level.

A second potential problem is that the semantic techniques include the definitions of signatures covered by IDL. In order to use both, we must find some way to harmonize the mappings defined by CORBA for IDL with the mappings defined by the tools associated with the semantic FDT.

Despite these potential problems, the benefits of FDTs are sufficient to recommend their use.

2.3.3. Evolution and Maintenance of the Specification

The current version of the specification (1.1) and the next version (1.2) are essentially traditional paper documents. Even though modern electronic document preparation technology is used to maintain the master version, the form is still one of a carefully prepared paper publication. In this respect it is like almost every other formal standard.

As we discussed in Section 2.2.1 we propose going a step further by making the electronic form itself the master version while doing it in such a way that equivalent versions of the specification can be produced for specific needs with a high degree of automation. This does not negate the change control process in any way although it may make the updating of changes easier. There is often a significant ripple effect when one change forces changes elsewhere in the document. A carefully constructed electronic hypertext document can make this updating simpler, and in some cases automatic.

There are several other recommendations to make the specification more manageable and useful:

Partition the specification. Chapter 5 is more than half the document and contains all of the formal specifications. It will also be the part that changes most rapidly and the part most useful in machine-processible form. Separating it from the rest of the document would facilitate maintenance.

Develop usage scenarios. There are many places in the interface definitions where the intention of the designers is ambiguous to outside readers, particularly as to whether an interface is supposed to cause a change in state or to record that a state has changed. Detailed scenarios of how some of the important interfaces are intended to be used would resolve the question for those not directly involved in the formulation and evolution of the specification.

Use IDL *modules* to control name scope. IDL has the concept of modules to control the scope of names (identifiers). These modules are not only useful as a software development tool, but also avoid potential problems of name collision when changes are made to other parts of the specification. Modules can be made to align to components.

Revise the CIM Framework to use CORBAfacilities and CORBAservices. The OMG Common Facilities and Common Services have been renamed, but they are still being actively developed to extend the range of functions defined by CORBA. Certain parts of the specification need to be reviewed in light of the latest proposals in such areas as event management and time services.

2.3.4. Ownership of the Specification

As the CIM Framework follows this roadmap to standardization, other groups will want ownership, in some form, as part of the process of becoming a standard. We have no complete solution to this problem, but we note that if it is possible to subdivide the specification to meet the needs of different manufacturing domains, as discussed in Section 2.2.3, then the same technique might be used to partition ownership. In any event, the effective control of the content of the CIM Framework remains with those who are willing to exert the effort to make it a success.

2.3.5. Support for Different Platforms

The original TI MMST system was implemented in Smalltalk. The current specification is more platform independent because it is based on CORBA interoperability. However, not all object systems are CORBA conformant, and it remains to be seen if various interoperability proposals will be successful. In particular, the Common Object Model (COM) differs from CORBA in several important ways. COM has single inheritance and multiple interfaces per class, but CORBA supports multiple inheritance and only a single interface per class. COM inherits method signatures in subclasses but not the implementations of those methods. Both object models will evolve, and there are powerful reasons pushing for successful interoperability, but the Microsoft Windows family of platforms based on COM, is important for the widespread acceptance of the CIM Framework.

2.3.6. Integrating Related Efforts

There are many other projects underway in applying information technology to manufacturing or to the larger enterprise of which manufacturing is a major component. Some of these projects are identified in Chapter 5 of the MfgSIG white paper (in preparation). While these projects may call their work a framework, a reference architecture, or a reference model, there are often similarities to the CIM Framework in purpose and content. It is important that SEMATECH identify such projects and find ways of leveraging their own work, as well as avoiding duplication of effort.

2.4. Technical Recommendations

The current specification (version 1.1) is based on the original Smalltalk work done for MMST. In the process of making the specification language independent, a number of Smalltalk idiosyncrasies have yet to be resolved. We have reported these to the MES Build Team and will not repeat them here. However, in the process of studying the specification and other IDL specifications, for example, National Industrial Information Infrastructure Protocols (NIIIP) and several of the proposals to OMG, we have come to a number of conclusions about a style of writing IDL. Here we present some of the most important in a series of recommendations about how to write specifications of this kind.

2.4.1. Parameters

Avoid ‘inout’ Parameters. Only use 'inout' where it is absolutely necessary and it should never be necessary. Do not use it to avoid inventing another parameter name. In fact all parameters should be grouped into the 'in' parameters first followed by the 'out' parameters, if any. This should always be possible because the IDL is based on a message passing paradigm, where the 'in' parameters are the request message, and the 'out' parameters plus the method value are the response message.

Use meaningful, but brief, parameter names. IDL is primarily a syntax specification. The small amount of semantics available in IDL is contained in the agreed upon (standard) types, the structure of **typedefs**, and whatever connotation is provided by the choice of names. It is impossible to be formal, or even rigorous, in specifying semantics through names alone; so, do not try too hard by using long, convoluted name, for example, **top-leftmost-branch-of-the-call-tree-if-there-is-one** . On the other hand do not be deliberately obtuse by using names like **astring** or **value1**.

Use 'readonly' wherever possible. 'Readonly' has two possible uses depending on exactly how the IDL is meant. One possibility is that it specifies an attribute that must be set at create-time, *i.e.* is part of the essential identity of the object and cannot be changed, there is no _set method. If an error is made in one of these values at creation, the only recourse is to destroy the object and create anew. This is a very important feature and is analogous to the key field(s) in a relational database. The other possible use of 'readonly' (where multiple interfaces are allowed) is to restrict the attribute in question for that interface (analogously, a view in database terminology) to be retrievable but not modifiable. This can have value for both performance optimization and security controls. It should be noted that CORBA does not allow multiple interfaces at this time, but it is a subject of debate.

2.4.2. Exceptions

Almost any method may raise an exception of some sort. If nothing else there might have been a hardware, software, or communications failure while the method was executing. Section 4.14 of CORBA 1.2 (Chapter 4 of *The Common Object Request Broker: Architecture and Specification*, document 93-12-43 on the OMG server) covers the standard exceptions provided by CORBA. These exceptions cover most possible failure modes, and a conforming application will need to handle these exceptions.

For one important set of objects, we believe that all exceptions are handled by the standard exceptions. This is the set of data-centric objects that are made up of _get methods (or _get and _set if the attribute is not 'readonly'). The same argument applies to other retrieval methods whether they are simple _get's or not. And the same rule can be applied even if the method is computational, if no non-standard exceptions can occur. No new exceptions need to be defined but the standard exceptions must be processed correctly.

Now to the case where there are truly non-standard exceptions possible. We use the word ‘exception’ rather than ‘error’ because there are many more interesting cases where exceptions are a normal but alternative response. For example, suppose a program requests an agent-object to perform some service. In processing the request, the agent concludes that although it cannot respond exactly to the original request, there is an alternative that might do the job, but the response is very different in structure. Raising an exception is the method of choice in IDL for returning a significantly different signature.

2.4.3. Returning Values

Defining exceptions is one of the weaker points of the current specification. We believe that the specification can be significantly improved by reviewing the values returned by the methods and adding exceptions where needed. Based on the discussion in the previous section, we can catalog object methods pragmatically into four categories based on how exceptions are used:

No non-standard exceptions. Return result, or if the result is more complex, return a main result as the value of the method and other results as 'out' parameters. They should be 'out' only, not 'inout'.

Simple, two-valued exception. A success/fail response is common to many methods. Return a boolean, and the actual results as 'out' parameters. This assumes that no additional information needs to be returned in the false case.

More than two modes of return. If the results have the same signature in every case, or are a subset (possibly empty), of the 'normal' return, then return an **enum value** from an **enum type** defined for this method, or a set of methods that share the same possible modes of return. This situation arises often in the specification where the state of a process is being queried or set. The specification uses numerous Boolean methods to report each state separately. By combining all these state-reporting functions into a single method that returns an **enum value**, the specification is not only much simpler to read and understand, but easier to modify. Even a structured state can be returned by only two or three methods, one for each partition of the state structure.

The most complex case. Where the 'normal' response and the 'exceptional' responses can be quite different, the programmer (method designer) will need to create one or more user-defined exceptions. This is also one of the few cases where a 'void' return might be appropriate. Think very carefully before invoking this heavy-duty machinery.

3. Testing and Certification

3.1. Introduction

A few definitions are in order:

Conformance: To be in accordance with some specified standard or specification.

Certification: A procedure by which a third party gives written assurance that a product, process, or service conforms to specific requirements.

One of the most critical aspects of a certification program is having it be accepted by the industry, and primarily the suppliers since they are most directly affected by the program. The suppliers should be involved from the very beginning of definition of the certification program in order to ensure the its success.

The current level of ambiguity in the CIM Framework specification makes it impractical to develop certification tests at this time. However, it is recommended that the details of the certification program (business model and methodology) be defined as soon as possible. This report reviews various models for certification programs and makes recommendations for the approach to be taken with regard to the CIM Framework.

The CIM Framework specification is understood to be a work in progress and is evolving as expected with new levels of detail at each revision. The addition of formal definitions, to describe the behavior of the framework, will greatly assist the certification program development, in that the expected behavior will be more rigorously defined. In addition the development of a reference implementation is strongly recommended to aid in the successful development of a certification program.

3.1.1. Defining Interoperability Goals

It is reasonable to ask the question, "what is the point of certification?” It is not just assurance of some level of *quality*. Usually certification conjures up notions of compatibility, interoperability, and portability [1]. In industry today, interoperability tests often refer to the testing, via pairwise matching, of specific supplier applications. This is a very expensive proposition especially as the number of applications to be certified increases.

In some cases, the certification of the application program interfaces (API) themselves provides a high level of interoperability. POSIX is a case in point. There is no explicit interoperability certification involved in the POSIX certification. However, one of the results of POSIX certification is the ability for different Unix implementations to interoperate at certain levels. This is due to the fact that the POSIX standard itself provides good coverage of the domain for which it is intended.

Interoperability or compatibility are loose terms that suggest some kind of cooperation or harmony among unlike components of a system. These terms have been applied to features ranging from "is written in the same language" to "can read ASCII" to "plug-and-play.” Portability is often mentioned when defining interoperability goals, and it usually means the ability to move a program or piece of data around among different environments and still be able to use it with a minimum of effort, even though the program may be very unlike other components in design or function. "Usually, the tighter the integration required to satisfy interoperability requirements, the more cost and effort involved. Real-life compromises will reflect acceptable thresholds of pain for integrators." [1]

For example, even perfect POSIX conformance still will not guarantee that an application will compile the first time on a foreign system. But compared to the problems of Unix porting in the past, these problems are considered minor, and do not detract from the usefulness of the POSIX standard in promoting portability. This should be understood in the context of the CIM Framework also, even if the framework does not provide perfect interoperability, it will be successful if it provides a noticeable net decrease in integration costs.

In the current CIM Framework specification it is not clear as to whether the primary goal is to promote and provide some level of re-use or whether the main focus is on interoperability. Our recommendation would be that the main focus should be on interoperability, and that re-use be considered a valuable side-effect of the specification. Enforcing good OO (object-oriented) programming practices on developers is neither practical nor useful. If a supplier can provide a product that passes certification at the component interface level, then it is irrelevant how the product is actually implemented. The focus should be on the plug-and-play aspect of interoperability with respect to the semiconductor manufacturing floor. However, it is very useful to educate the developers about the tremendous benefits available via the use of object-oriented technology.

The reference implementation is critical in order to provide some initial CIM Framework services and simple applications, which suppliers’ products may be tested against. The key here is not only to develop a robust reference implementation, but to also develop detailed scenarios which exercise a wide range of application interaction.

3.2. Business Case for Certification Testing

One of the first things that should be decided when designing a test plan for a specification is who will pay for certification? Certification or the issuing of a certificate is actually the final step in the process. The creation of a full certification program will require several different steps, the main four ones being:

1. Test suite development
2. Testing service procedures and execution
   (may include accreditation procedures for third party labs)
3. Maintenance of the test suite
4. Administration and issuing of certificates

There are several potential methods for funding:

A consortium pays

Example: VHDL members put in funds and resources and contracted with a university to develop the conformance test suite. SEMATECH could fund the effort initially and plan to migrate support of the program to another organization.

Customers pay

Example: the customers (semiconductor manufacturers) may not want to pay directly but they may support paying premium prices for products which are certified.

Suppliers pay

Example: The suppliers as well as customers benefit from a strong certification program, because of increased market opportunity, reduced internal costs, reduced support costs, and better customer satisfaction.

Public organization pays for development of a certification program

Example: SQL, however, even if this is used to develop the initial program, there should be a long term plan to make the program self sufficient.

Independent company develops the test suite

Example: Perenial, Inc. developed the C programming language test suite. There was no direct payment to Perenial, however NIST (the certification authority in this case) had an agreement with Perenial to recognize their C test suite for conformance testing.

The consortial approach is recommended because it would spread the initial cost over most major users. SEMATECH could provide this funding, which could entitle member companies to discounted certification. The idea would be for SEMATECH to explore other funding sources to maintain the program.

3.2.1. Examples of Related Industry Programs

CFI (CAD Framework Initiative, Austin TX)

The CAD Framework Initiative is developing standards to solve interoperability issues in the electronic computer-aided design (ECAD) world. CFI is a consortium of ECAD vendors who help support certification and standards activities through membership fees.

Currently CFI has a certification program in place for the Design Representation (DR) standard for electrical connectivity. The DR is an API that allows client tools to extract netlist information from a circuit design in a uniform manner directly from commercial object-data repositories, regardless of the actual storage mechanisms of the objects, or underlying database format. The DR eliminates the need for translation of design information into flat files in order to move design data between unlike clusters of tools. It also eliminates the need for a tool vendor to support an arbitrary number of proprietary database interface mechanisms for access to this information.

Both server (framework) and client (tool) certification is performed. Certification services are priced to encourage smaller company participation, as well as reward sponsor membership in the CFI consortium:

Table 1 CFI Certification (Example Pricing)

CFI Certification (Example Pricing)		
Product Type	**Price (1)**	**Price (2)**
Framework DR	$15,000	N/A
Version Upgrade	$10,000	N/A
DR Tool	$10,000	$5,000
Version Upgrade	$5,000	$2,500

Price (1) Sponsor may deduct credits from member fees.
Price (2) Special "Emerging Company" price [sales less than $20 million].

A DR Toolkit is available for $6,000 (with a 25% discount for corporate members) to assist vendors with porting, as well as testing activities. It contains test data (both realistic and synthetic), example client implementations, a server reference implementation, helper code examples, implementation notes and advice, the complete DR standard, a binary instrumentation monitor for client applications, and a set of test suites. Note: the toolkit is included in the price of certification. This was done purposely to encourage suppliers to commit to the certification process.

CFI uses a self-certification model, sharing the testing burden with vendors, while making it easier for them to incorporate certification in their natural product cycle. This is combined with an audit provision to enforce conformance.

Server testing is based on a complete implementation of the standard. (Partial, or levels of, server compliance was rejected by CFI as confusing to the end customer.) The DR Toolkit contains all the test suites necessary for the standard; however, the vendor is obligated to provide all additional test cases for behaviors that reference the standard but are not covered by the standard suites.

Application testing is based on categories of client tools, each of which must support a minimal set of functionality of the standard in order to be considered useful (in terms of interaction with other tools, as well as portability of design data). Each category has a set of test cases supplied by CFI for the minimal set. The vendor is also obligated to create test cases for all remaining behaviors that intersect with the standard, but are not covered in the minimal requirements tests.

Binary instrumentation code enforces test coverage by monitoring all activity along the DR program interface (PI). Vendors are required to supply an unstripped version of the object module, whose symbols are analyzed for use by the DR PI. Then the test suite is run using a specially wrapped version of a server, which intercepts all calls to the PI, logs information about each one, and tallies them according to the actual addresses of the calls in the code. As each successive test is run, the call records accumulate. At the end, any calls to PI routines that were *not* exercised by the tests are reported. It is not sufficient for a product's test cases to have passed through the PIGetLibs() call, for example, it is necessary that *every* point in the code that has a call to PIGetLibs() has been exercised by the test cases (and passed). This monitor provides a way to insure that 100% of a product's intersection with the standard PI has been tested by the suite, at least right at the point of the API (if not all the secondary paths nearby). This technique skirts the typical arguments against test metrics, and obviates source-code inspection to pull out all the call sites. (Naturally, the monitor code is OS-specific and dependent on the application binary interface definition. In cases of multiple platform certification, the monitor need only be run on *one* of the platforms.)

The procedure requires vendors to complete a test plan, which CFI must approve, submit object versions (only) of the product being tested (to become part of CFI's interoperability lab), implement the test cases (not already available in the DR Toolkit), run the tests, return results for CFI approval. CFI performs all certifications in-house, and awards a brand mark for display on product packaging and advertising (according to certain guidelines). At any point, including any time after the process, CFI may perform an audit of the tests to insure ongoing conformance.

An Issue Tracking System is maintained with convenient E-mail and World Wide Web interfaces for logging problems, tracking them, and identifying fixes.

(Contact: Don Cottrell, cottrell@cfi.org, 512-338-3731)

Novell (Provo UT) NetWare

The Novell program has been in place for several years and boasts more than 3000 certified products. The program comes in two flavors:

(1) YES It Runs with NetWare

This brand mark results from a self-certification program for vendors that involves paying a fee and completing a survey detailing the product's compatibility with NetWare.

(2) YES NetWare Tested and Approved

This brand mark is reserved for products that have been tested by Novell Labs, a separate testing division of Novell, Inc. Novell claims that this rigorous process includes actual cross compatibility tests with every (appropriate) certified product to date, performed by 2500 servers and workstations running 24 hours a day.

Novell also has a Certification Alliance designed to help developers integrate certification with their product cycles, increase NetWare expertise, and address compatibility issues earlier. Membership carries a $35,000 annual fee, which includes training, a testing kit, site inspection, and a number of support incidents.

3.2.2. Selling Certification

Certification is a marketable service and must be promoted like any product or service. For example, Taligent recently announced their new certification and branding program at COMDEX/Fall'94 for its application system CommonPoint™ (formerly known as TalAE™) [2]. "We wanted the product name to connote the fundamental promise of the Taligent solution; a common foundation for next generation applications; a common platform for distributed computing; and a common user environment for people to work together," said Joe Guglielmi, Taligent chairman and CEO. "With CommonPoint we believe we've succeeded in choosing a name that appropriately represents this vision. And through the execution of our branding and certification program, we'll be able to ensure a consistent set of object-oriented API's are deployed across leading hardware and operating system architectures. This will provide developers and users with a common point for a new generation of computing." In addition, Taligent announced that it intends to make its APIs and the test suites available to OEMs and standards bodies. In January 1994, Taligent announced plans to submit its APIs to industry standards associations such as X/Open Company Ltd. and the OMG.

As in any marketing effort the benefits of using the product or service must be highlighted when presenting to the target audience. For example, the approach used with a company which supplies tools to semiconductor manufacturers may be somewhat different from the approach used with information systems (IS) people within a semiconductor manufacturer. Suppliers wish to sell more of their products and services, whereas IS people want to be able to integrate new tools quickly into their manufacturing line. It can be the case that suppliers are fearful of open standards. This is because open standards are intended to prevent a supplier from holding a customer hostage within a proprietary environment. This attitude needs to be understood in order to effectively convince the suppliers that the benefits of certification to an open standard outweigh any negative side effects. Some of the benefits are decreased time to market, reduced development costs, improved quality by leveraging off of test suites, and increasing the potential market, in addition to:

- Easier entry into previously *closed* shops, by virtue of being able to introduce a small piece rather than a complete solution, *i.e.*, plug-and-play provides a mechanism for a supplier to more easily get their foot in the door. In general it is easier to convince a customer to try a small piece of your solution; rather than starting at ground zero with your products, you can introduce them slowly to help a customer migrate to your tools.

- In companies which have decentralized purchasing, different manufacturing sites can easily purchase different tools to suit their specific requirements. Of course large companies may want to take advantage of volume purchase agreements, but the CIM Framework makes it much easier for them to do it with multiple vendors.

- Suppliers may be able to deliver products sooner, since the plug-and-play nature of the CIM Framework allows finer grain resolution on integrating new tools into the line. This has the result of effectively shortening the development cycle and introducing products in phases.

- Customers are more receptive to buying certified products.

- Certification can be touted by suppliers as a testimonial to the quality of their products, especially if the testing is done by an independent laboratory.

- Successful completion of the certification program can result in gaining insight and experience in quality issues that can be reapplied in other company development processes.

- Customer support costs are less because of:

- ▸ Better design paradigms inherent in the standards
- ▸ Rigorous testing required for certification

- Inclusion of a supplier’s products in various announcements or listings of conformant applications, provides additional marketing which can generate new sales for a supplier. Some of the means of disseminating this information are: press releases, trade journals, Usenet news groups, the World Wide Web, or a formal registry.

For example, Novell circulates a test bulletin (via their YES source book, NetWare support encyclopedia, NetWire, Reseller News, and the IMSP index) to inform the industry about a product's Tested and Approved status. CFI maintains current lists of certified products, as well as interactive demonstrations on the World Wide Web. They also circulate press releases to appropriate channels, and have a ready-made network of key industry contacts by virtue of their position as a consortium [1].

Developing test suites can be incredibly expensive. For example, here are some rough estimates of the effort expended developing certification test suites for some common languages, at the API level (where a test suite is a collection of individual test cases).

Table 2 Certification Test Suite Effort Levels

Specification	**Estimated effort (Person Years) for certification test suite development**
FORTRAN	9-10
VHDL	9-10
COBOL	10-15
POSIX	10-11
C	4-5
SQL	10+

Note that this only addresses API level testing and does not include any interoperability testing of various implementations of the above specifications working with multiple implementations.

Because of the similarity between POSIX procedure calls and CIM Framework methods, we can estimate the effort needed to develop certification for the CIM Framework. The number of procedure calls in POSIX can be compared to the number of methods in the CIM Framework. There are approximately 250 procedure calls in the POSIX standard. There are 3000 tests in the POSIX test suite. From this we can estimate 3000/250 = 12 tests per procedure call. In the CIM Framework specification there are approximately 770 unique methods. If we use the POSIX model we can estimate the number of tests per method as 12, then 12x770 = 9240 tests for the CIM Framework. That is approximately three times the number of tests required for POSIX, which would give an estimate of 3 x (9 to 10) person-years or between 27 and 30 person-years of effort to code the test suite for API level testing of the CIM Framework. See section 3.4 below for recommendations to address the high cost of testing.

3.3. Methodology for Certification Testing

Another key question to be answered when defining a certification testing program is, "who will perform the test service?" First let us consider the three main phases which define the execution of a certification program. They are:

1. Research and development phase

- Test suite development
- Final test execution procedure definition
  - Details for execution of certification tests
- Criteria for recognizing testing laboratories
  - Accreditation and written agreements with laboratories (if used)

2. Testing phase

- Execution of testing according to procedures
- Generation of reports

3. Certification phase

- Review of test reports
- Issuing of certificate, addition to a registry *etc.*

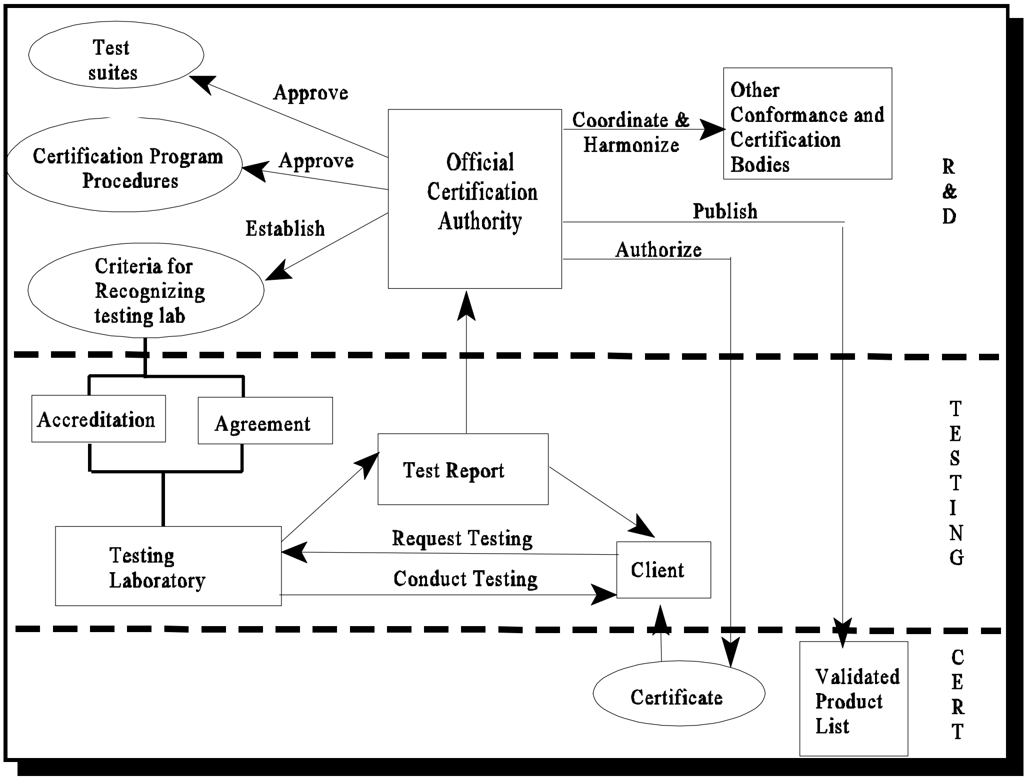

Figure 1 *Certification program components*

Figure 1 shows many of the tasks that might be involved in a certification program. Note that a given methodology may not incorporate every task in Figure 1. Note also that some tasks may be provided by more than one organization. For example, in the self-certification methodology described below, the testing laboratory task can be performed either by the supplier, or by the official certifying body, in the event of an audit.

One methodology sometimes referred to as *self-certification* requires the supplier to provide a test plan to the official certifying body. The certifying body reviews the test plan and recommends changes if necessary. Changes to the test plan are made until the certifying body approves the test plan. The supplier then executes the test plan against their product. The official certifying body reviews the results of the suppliers testing and delivers the certificate indicating the results. (See certificate contents). The official certifying body reserves the right (at any time and for any reason) to audit the test, or to require the performance of the tests in the presence of a representative of the certifying body. This is the recommended method because of the minimal cost and the benefits to suppliers. This model is based on the CFI DR certification program. By using the provided test suites during development suppliers can improve both their quality and time to market. This aspect increases the appeal of the certification program to the suppliers.

Another methodology would use the services of an accredited testing laboratory, i.e., a laboratory accredited through a formal laboratory accreditation program such as National Voluntary Laboratory Accreditation Program (NVLAP). The accredited laboratory would then execute the testing phase. This might involve a representative of the accredited laboratory visiting the supplier to supervise the execution of the appropriate tests. The accredited lab then sends the reports to the official certifying body. One of the issues with this approach is that the laboratory needs to maintain competency and be re-accredited periodically which can be costly. One of the advantages of this approach is that the testing is performed by an independent third party.

Yet another interesting methodology might involve an agreement or contract between two official certifying bodies. For example, official certifying body A might recognize the certification by another official certifying body B as acceptable for issuance of A's certificate. An example might be for a foreign certification body such as the NCC (National Computing Centre) in the United Kingdom entering into an agreement with a U.S. certifying body. This approach could save a great deal of money since the testing phase and part of the certification is already done. Typically this method could only be used on a very mature standard.

In some cases the government performs the testing service, for example, SQL; however, this of course requires an approved and funded program.

The Certificate itself should contain at least the following information:

- Procedures followed (may include a detailed test plan)
- Versions (and model numbers) of all relevant software and hardware components used during testing.
- Profiles of tests performed (For example, what CIM Framework components were tested?)
- Organization performing the testing
- Organization auditing the testing (if applicable)
- Evidence or reference to related standards conformance (as required)
- Overall Pass/Fail status

3.4. Summary of Recommendations on Testing

We strongly recommend that the suppliers be involved as soon as possible in the definition of the certification and conformance testing program. Without the suppliers' buy-in the program cannot succeed. This must take into account certifying legacy applications and fully compliant implementations. Giving the suppliers the certification test suites provides an incentive for them to take part in certification. The test suite would be expensive for any one of them to develop alone but will be extremely valuable to all of their quality assurance processes.

We strongly recommend that any requirement to have access to a supplier's source code be abandoned. Other certification programs have found that suppliers are extremely unwilling to provide access to their source code, since in most cases this is how they distinguish themselves in a competitive marketplace. All supplier product testing for the purposes of certification should be considered from a black box perspective, *i.e.*, without reference to the source code, only from the interfaces defined by the IDL in the specification.

There has been some discussion as to whether the CIM Framework architecture should be based on a backplane or stereo component metaphor. This decision cannot, and should not, be made on the basis of what is *better* for certification. A certification program can be built around any architecture. However, we strongly recommend that the *backplane* approach be adopted. The backplane approach allows for minimal impact to the suppliers, and it fits with the notion of framework services being provided to all applications and becoming absorbed into the operating system.

We recommend that the high cost of deploying a certification program be highlighted to SEMATECH management so that they are aware of what is involved in such an endeavor. The proper resources must be applied to the problem in order to have any reasonable effect. Generally, for a complex standard the cost will be in the millions of dollars. Measuring the cost in terms of dollars however is not the best approach to analyze the effort required. A better metric is the person-years required to generate the test suite and develop the certification program (see Table 2 above).

We recommend that methods for automatically generating tests be explored further. There are research projects and companies that are focusing on this question. Interactive Development Environments (IDE), for example, has developed a tool which reads in IEEE Standard 1175 STL (Semantic Transfer Language) and automatically creates tests. Other research is being done on ADL (Assertion Definition Language). Any automation of the test generation could provide tremendous savings to SEMATECH in developing a certification program.

We recommend supplier involvement in test suite development to establish a consensus for the program. We also recommend investigating universities as a technical resource for manual and automatic test generation. This would also transfer knowledge of the CIM Framework to the next generation of engineers.

We recommend a self-certification program which is audited by SEMATECH. There are many advantages to the self-certification program; for example, it involves the suppliers and has benefits for them such as improvements in the quality assurance process. The CFI model is a recommended model, with the possible exception of binary instrumentation. Also, an agency other than SEMATECH may be better suited for administering the certification program in the long run.

We recommend leveraging off of existing standards, for example, requiring all communication to be CORBA conformant will greatly enhance the interoperability of the specification. Currently the specification discusses CORBA but never indicates that it is a requirement. The certification plan could then require CORBA conformance prior to CIM Framework certification. Also, pushing any common facilities into other standards efforts would off-load some effort from SEMATECH. An example of this might be that event management could be provided by CORBAfacilities.

We recommend that the component dependencies, as indicated by the McGuire spreadsheet, be included in the specification. This would allow suppliers to determine the minimum configurations required for interoperability.

We recommend that a document be generated which contains the assertions for the specification. This provides a formal document which defines what is to be tested and how. This is closely related to exploring automated methods of test generation from the formal specification.

We recommend that SEMATECH, with our help, and a group of semiconductor suppliers reach agreement on the test suite, which is then developed by a small group. In other words, the test suite is best developed by a small group of experts, preferably those most familiar with the CIM

Framework, but it is recommended that as large a group as is feasible reach consensus on the acceptability of the test suite. This *buy-in* is required for the success of the program.

3.5. Certification of Extensions to the CIM Framework

In general suppliers who add extensions to the CIM Framework should be responsible for supplying the certification tests for their extensions; these tests would be subject to approval by the official certifying body. There has been some discussion regarding the need for certification to detect the use of so-called *backdoors* which extensions might use. This may be valuable during initial validation programs to determine the level of conformance of legacy applications, but it is not clear whether this is useful during the final certification testing. The reason being that if an implementation passes certification testing at the component interface level it should be certified (at the API or interface level), regardless of whether it is coupled via non-framework interfaces in a given supplier's configuration. As long as the component interfaces are exposed properly then other components can be properly connected to them as indicated in the specification.

Works Cited

[1] David Mallis, *Final Draft Report on Compliance and Certification*, contractor report (unpublished) for contract 43NANB510468, National Institute of Standards and Technology, Gaithersburg MD 20899, 1995.

[2] Taligent, Inc., press release from COMDEX/Fall'94 in Las Vegas, Nevada, on November 14, 1994.

Provenance:

Version 6, July 13, 1995, to modify in light of Barbara Goldstein’s suggestions. Last revision before printing. WERB approved as NISTIR 5679. This version will also be posted to the Web in PDF and PS formats. Web versions named roadmap.pdf, .ps, and .htm according to format.

Version 5 updated May 24, 1995, to correct Cottrell phone number and add Works Cited section instead of the appendices for NIST publication.

Version 5, May 4, 1995: included all changes from Weber (A number of his comments were questions for the future).

Version 4, April 3, 1995: including all the changes and corrections from Stewart, St. Pierre, Field, and Weber (on first half).

Version 3, March 31, 1995: including Stewart’s changes.

Version 2, March 24, 1995: including St. Pierre’s changes.

Version 1, March 17, 1995: Final draft.