SEMATECH

Roadmap for the Computer Integrated Manufacturing (CIM) Application Framework

SEMATECH
Technology Transfer 95052825A-ENG

Roadmap for the Computer Integrated Manufacturing (CIM) Application Framework

Technology Transfer # **95052825A-ENG**
SEMATECH
May 31, 1995

Abstract: This technology transfer reports on the first year of a joint project between the National Institute of Standards and Technology (NIST) and SEMATECH. It includes a roadmap for promoting adoption of the SEMATECH Computer Integrated Manufacturing (CIM) Framework as an industry standard. The CIM Framework is an object-oriented software infrastructure that creates a common environment for integrating applications and sharing information in production environments. The roadmap consists of the following stages: specification; consensus; standardization; and testing/certification. Each stage is described in detail in the document.

Keywords: Computer Software, CIM, Object Oriented Systems, Frameworks, Specifications

Authors: S.L. Stewart, James A. St. Pierre

Approvals: Carla Jobe, Project Leader
Glenn Hollowell, Project Manager
Gary Gettel, Director, Manufacturing Systems Development
Dan McGowan, Technical Information Transfer Team Leader

Table of Contents

List of Figures

List of Tables

Acknowledgements

The authors wish to thank the following persons at the National Institute of Standards and Technology (NIST) for their assistance in developing this report: Neil Christopher, Elizabeth Fong, Barbara Goldstein, Greg Koeser, Tom Kramer, Michael McCaleb, Michael McLay, Steve Osella, and Evan Wallace.

The authors also wish to ackowledge the following persons for valuable discussions: John Barkley, Ed Barkmeyer, Kevin Brady, Tony Cincotta, Barbara Cuthill, Martha Gray, Shirley Hurwitz, Arnold Johnson, and Tom Rhodes.

Also, Joe Chandler provided valuable technical support.

1 EXECUTIVE SUMMARY

The document describes results of the first year of a joint project between the National Institute of Standards and Technology (NIST) and SEMATECH, and lays out a roadmap for adoption and use of the SEMATECH Computer Integrated Manufacturing (CIM) Framework. This roadmap includes the following steps:

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

This report also makes the following prioritized recommendations:

- 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 (OMG) (and its Manufacturing Special Interest Group) to promulgate the specification.
- Reach consensus on a 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 Common Object Request Broker Architecture (CORBA) classes, including 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.

The report makes several other technical recommendations. Background material and supporting documentation are available separately from the project manager.

2 INTRODUCTION

This report covers the first year of a joint project between 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.

This report includes a roadmap to adoption, use, standardization, testing, and certification of the CIM Framework developed by SEMATECH. Although recommendations in this report have been presented to SEMATECH and often are based on SEMATECH information, they represent NIST's vision for the future direction of the joint project.

Certain commercial products are identified in this report. However, such identifications are for clarity and do not imply recommendation or endorsement by NIST.

3 STANDARDIZATION

3.1 Background

SEMATECH developed the CIM Application Framework based on work by Texas Instruments (TI), a member company, in the U.S. Defense Department's Microelectronics Manufacturing Science and Technology (MMST) project. The CIM Framework's goals are to promote integration on the shop floor, reduce costs, and increase reuse through object-oriented (OO) technology. The CIM Framework is compatible with the Object Management Group's (OMG's) 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 its current version, the specification presents the interfaces in terms of Harel state charts and Rumbaugh diagrams and in conventional language.

3.2 Roadmap

A roadmap for standardizing the CIM Framework goes through several stages. The analogy to a roadmap is only loosely true, because the stages overlap and most activities must occur in parallel. However, the emphasis and level of effort will shift as participants progress through this process. The principal stages—specification, consensus, standardization, and testing/certification—are detailed below.

3.2.1 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) development team at SEMATECH. Version 1.0 was published by SEMATECH on March 31, 1994, and has been revised twice since then. The current version, *Computer Integrated Manufacturing (CIM) Application Framework 1.2*, (Technology Transfer #93061697E-ENG) is available from SEMATECH, while version 2.0 is under development.

This project took the electronic version of the original specification and converted it into a Hypertext Markup Language (HTML) document suitable for online browsing, thus making it readable on the World Wide Web. HTML is a specialization of the Standard Generalized Markup Language (SGML) (IS0 8879) used by Web servers to format compound documents. The project subsequently updated the online document to reflect version 1.2. Doing so 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 of SEMATECH were instrumental in making this conversion possible.

The authors recommend that this process be extended 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 hard copy version for printing, (2) an HTML version for on-line browsing, and (3) an extract of the formal portions of the specification for computer processing. This concept is called single source specification management.

3.2.2 Consensus

To achieve SEMATECH's objectives, it is not sufficient to produce a specification, even one of technical excellence. There also must be widespread agreement among 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 already has involved users' groups from its member companies in developing the specification. It also is contacting independent suppliers and providing orientation and training about the CIM Framework in scheduled classes and public conferences. The authors believe that this process of awareness, involvement, and training is absolutely essential to the success of the CIM Framework, and recommend that it be continued and expanded to the limits of the resources available.

3.2.3 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, by groups of interested parties, or by companies through widely used products.

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

In semiconductor manufacturing, Semiconductor Equipment and Materials International (SEMI) 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 is formed to develop a specific technological area where consensus on standards is essential to creating a viable market for the new technology. An important example for the CIM Framework

is the OMG, organized in 1989 to develop an Object Management Architecture (OMA) and 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 IDL. These IDL interface specifications comprise the computer processable 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 1994 and provides the chairman, Fred Waskiewicz. This CIM Framework joint project has actively assisted this process by providing support for the work of the MfgSIG.

The authors believe the CIM Framework specification is sufficiently general to serve as a basis for an OO manufacturing framework to meet OMG's needs. The MfgSIG already is studying this issue. In part, it is believed 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 specific to semiconductors or even electronic manufacturing.

Another benefit of the electronic version of the specification was illustrated by SEMATECH's Paul McGuire, who generated a complete, spreadsheet-based cross-reference to the class definitions. This product is useful for analyzing and checking the class definitions, and is available from SEMATECH as *Computer Integrated Manufacturing (CIM) Application Framework Specification 1.2 Interactive Browser Spreadsheet*, Technology Transfer #95042783A-ENG.

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 international ones, such as the ISO technical committees. Third, through its more than 500 members, OMG provides access to a major segment of the OO information technology supplier community. This will help project participants 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 this roadmap to standardization, the authors see three or four potential liaisons:

1. With SEMI for semiconductor-specific standards
2. With IEC/TC93 (Design Automation) for more general electronic industry standards

3. With ISO/TC184/SC5 (Manufacturing Automation/Architecture and Communications) for more general manufacturing standards
4. Possibly with subcommittees of the ISO/IEC Joint Technical Committee 1 (JTC1), if there are information technology standards that go beyond manufacturing.

SEMATECH and NIST together have extensive working relationships with all these parties. As the specification evolves, the joint project should evaluate opportunities for establishing such liaisons as the mechanism for formal, international standardization. Great leverage can be obtained through these partnerships.

3.2.4 Testing and Certification

The CIM Framework will not succeed 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 to test 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; the results are discussed in detail in Section 4 below.

3.3 Significant Issues

This section briefly identifies several issues that can have an important influence on the success of the CIM Framework. In some cases, specific recommendations are made. However, it is believed that all of these issues warrant continued attention for the life of the project.

3.3.1 Supplier Involvement and Support

Earlier, the authors 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 expected cost and productivity benefits. As much as 70% of the cost of semiconductor manufacturing software (especially integration costs) are generated by the manufacturer's in-house software groups. So, the semiconductor manufacturer is typically both a user and a supplier. When the manufacturer contracts for systems integration, a third-party integration company 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 the joint project needs to identify and work with trade associations and consortia to reach as many suppliers as possible. At the same time, the project needs to continue educating user companies as both users and internal suppliers. Often, these may be two quite separate sub-organizations.

3.3.2 Formal Specification

The authors very much support the use of formal description techniques (FDTs) 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. However, 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 offer 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, their semantics also must be captured. Lawrence Eng of SEMATECH is making a valuable contribution by exploring VDM++ as a semantic specification technique, and this project plans to leverage that work into the development of test implementations of applications.

In version 1.2, the semantics are captured in a combination of 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 problems will be encountered when the CIM Framework is integrated 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 definitions of signatures covered by IDL. In order to use both, some way must be found 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.

3.3.3 Evolution and Maintenance of the Specification

Version 1.2 of the specification essentially is a traditional paper document. 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 discussed in Section 3.2.1, a further step should be taken to make the electronic form itself the master version while allowing equivalent versions of the specification to 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:

1. **Partition the specification.** Chapter 5 is more than half the document and contains all of the formal specifications. This part changes most rapidly and would be the most useful in machine-processable form. Separating it from the rest of the document would facilitate maintenance.

2. **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.
3. **Use IDL *modules* to control name scope.** IDL includes the concept of modules to control the scope of names (identifiers). These modules are useful as a software development tool, and avoid potential problems of name collision when changes are made to other parts of the specification. Modules can be made to align to components.
4. **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.

3.3.4 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. The authors have no complete solution to this problem, but note that if it is possible to subdivide the specification to meet the needs of different manufacturing domains (as discussed in Section 3.2.3), then the same technique might be used to partition ownership. In any event, effective control of the content of the CIM Framework remains with those who are willing to exert the effort to make it a success.

3.3.5 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 widespread acceptance of the CIM Framework.

3.3.6 Integrating Related Efforts

Many other projects are underway to apply information technology to manufacturing or to the larger enterprise of which manufacturing is a major component. Some of these projects are identified in the *Appendix on Related Activities*, a draft of Chapter 5 of the MfgSIG white paper. 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 for this project to try to identify such projects and find ways to leverage its own work, as well as avoid duplication of effort.

3.4 Technical Recommendations

The current specification is based on the original Smalltalk work done for MMST. In making the specification language independent, a number of Smalltalk idiosyncrasies have yet to be resolved. These have been reported to the MES build team and will not be repeated here. However, in studying the specification and other IDL specifications, e.g. the National Industrial Information Infrastructure Protocols (NIIIP) and several proposals to OMG, the authors have come to a number of conclusions about a style of writing IDL. Below are presented some of the most important in a series of recommendations about how to write such specifications.

3.4.1 Parameters

1. 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.
2. 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, e.g. 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.
3. 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.

3.4.2 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, the authors 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 _gets or not. And the same rule can be applied even if the method is computational, if no nonstandard exceptions can occur. No new exceptions need to be defined, but the standard exceptions must be processed correctly.

In the case where truly nonstandard exceptions are possible, the authors 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.

3.4.3 Returning Values

Defining exceptions is one of the weaker points of the current specification. The authors 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, object methods can be cataloged pragmatically into four categories based on how exceptions are used:

- **No nonstandard exception.** 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) must 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.

4 TESTING AND CERTIFICATION

4.1 Introduction

Definitions of two special terms used in this section are appropriate. These are as follows:

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 acceptance by the industry—primarily the suppliers, since they are most directly affected by the program. The suppliers should be involved from the very beginning of certification program definition in order to ensure 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 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.

4.1.1 Defining Interoperability Goals

It is reasonable to ask, "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. 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 "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.

For example, even perfect POSIX conformance still does not guarantee that an application will compile the first time on a different 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 whether the primary goal is to promote and provide some level of reuse or to provide interoperability. The authors recommend that interoperability should be the main focus, and that reuse be considered a valuable side effect of the specification. Enforcing good OO 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 through the use of OO technology.

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

4.2 Business Case for Certification Testing

One of the first steps in designing a test plan for a specification is deciding 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 requires several steps, the four main 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. For example, customers (semiconductor manufacturers) may not want to pay directly, but might support paying premium prices for certified products.

- Suppliers pay. Suppliers and customers benefit from a strong certification program, because of increased market opportunity, reduced internal costs, reduced support costs, and better customer satisfaction.
- A public organization pays for development of a certification program, as with Structured Query Language (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.
- An independent company develops the test suite. As an example, Perenial, Inc. developed the C programming language test suite. There was no direct payment to Perenial, but NIST (the certification authority in this case) agreed to recognize Perenial's C test suite for conformance testing.

The consortium 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 the SEMATECH to explore other funding sources to maintain the program.

4.2.1 Examples of Related Industry Programs

4.2.1.1 CAD Framework Initiative

The CAD Framework Initiative (CFI), based in Austin, TX, 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 are performed. Certification services are priced to encourage smaller company participation, as well as reward sponsor membership in the CFI consortium. Table 1 shows products and prices.

Table 1 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 and 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 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 must 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 also is 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 ensure that 100% of a product's intersection with the standard PI has been tested by the suite, at least 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, and 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 ensure 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.

More information is available from Don Cottrell, cottrell@cfi.org, (512) 338-3379.

4.2.1.2 Novell NetWare

The program of Novell NetWare, Provo, UT, has been in place for several years and boasts more than 3,000 certified products. The program offers two brand marks as follows:

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 2,500 servers and workstations running 24 hours a day.

Novell also has a Certification Alliance designed to help developers integrate certification with their product development 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.

4.2.2 Selling Certification

Certification is a marketable service and must be promoted like any product or service. For example, Taligent recently announced its new certification and branding program at COMDEX/Fall 94 for its CommonPoint application system (formerly known as TalAE).

"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 for the target audience. For example, the approach used with a company that supplies tools to semiconductor manufacturers may be somewhat different from that used with information systems (IS) people within a semiconductor company. Suppliers wish to sell more of their products and services, while IS people want to be able to integrate new tools quickly into their manufacturing line. In fact, suppliers may be fearful of open standards, since they are intended to prevent a supplier from holding a customer hostage within a proprietary environment. This attitude must be understood in order to effectively convince 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 test suites, and increasing the potential market. Other benefits include:

- Easier entry into previously closed shops by being able to introduce a small piece rather than a complete solution; for example, plug-and-play provides a mechanism for a supplier to more easily get its foot in the door. In general, it is easier to convince a customer to try a small piece of your solution; rather than start customers at ground zero with your products, you can introduce them slowly to help them migrate to your tools.
- In companies that 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 so with multiple suppliers.
- 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 effectively shortens the development cycle and introduces 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 provide insight and experience in quality issues that can be reapplied in other company development processes.
- Customer-support costs are lower because of better design paradigms inherent in the standards, and 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. Means of disseminating this information include press releases, trade journal articles, Usenet news groups, the World Wide Web, or a formal registry.

  For example, Novell circulates a test bulletin (via its 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. CFI also circulates press releases to appropriate channels, and has a ready-made network of key industry contacts by virtue of its status as a consortium.

Developing test suites can be incredibly expensive. For example, Table 2 contains some rough estimates of the effort expended developing certification test suites for some common languages at the API level (where a test suites is a collection of individual test cases).

Table 2 Certification Test Suite Effort Levels

Specification	Estimated Effort (in 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, one 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, and there are 3,000 tests in the POSIX test suite. Dividing the latter by the former, it can be estimated that there will be 12 tests per procedure call. In the CIM Framework Specification, there are approximately 770 unique methods. Using the POSIX model, the number of tests per method can be estimated as 12; multiplying that by 770 methods yields 9,240 tests for the CIM Framework. That is approximately three times the number of tests required for POSIX, which would require an

estimated 27 to 30 person-years of effort to code the test suite for API-level testing of the CIM Framework. See section 4.4 below for recommendations to address the high cost of testing.

4.3 Methodology for Certification Testing

Another key question to answer when defining a certification testing program is, "Who will perform the test service?" First, one must consider the three main phases that define the execution of a certification program. They are:

4.3.1 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)

4.3.2 Testing Phase

- Execution of testing according to procedures
- Generation of reports

4.3.3 Certification Phase

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

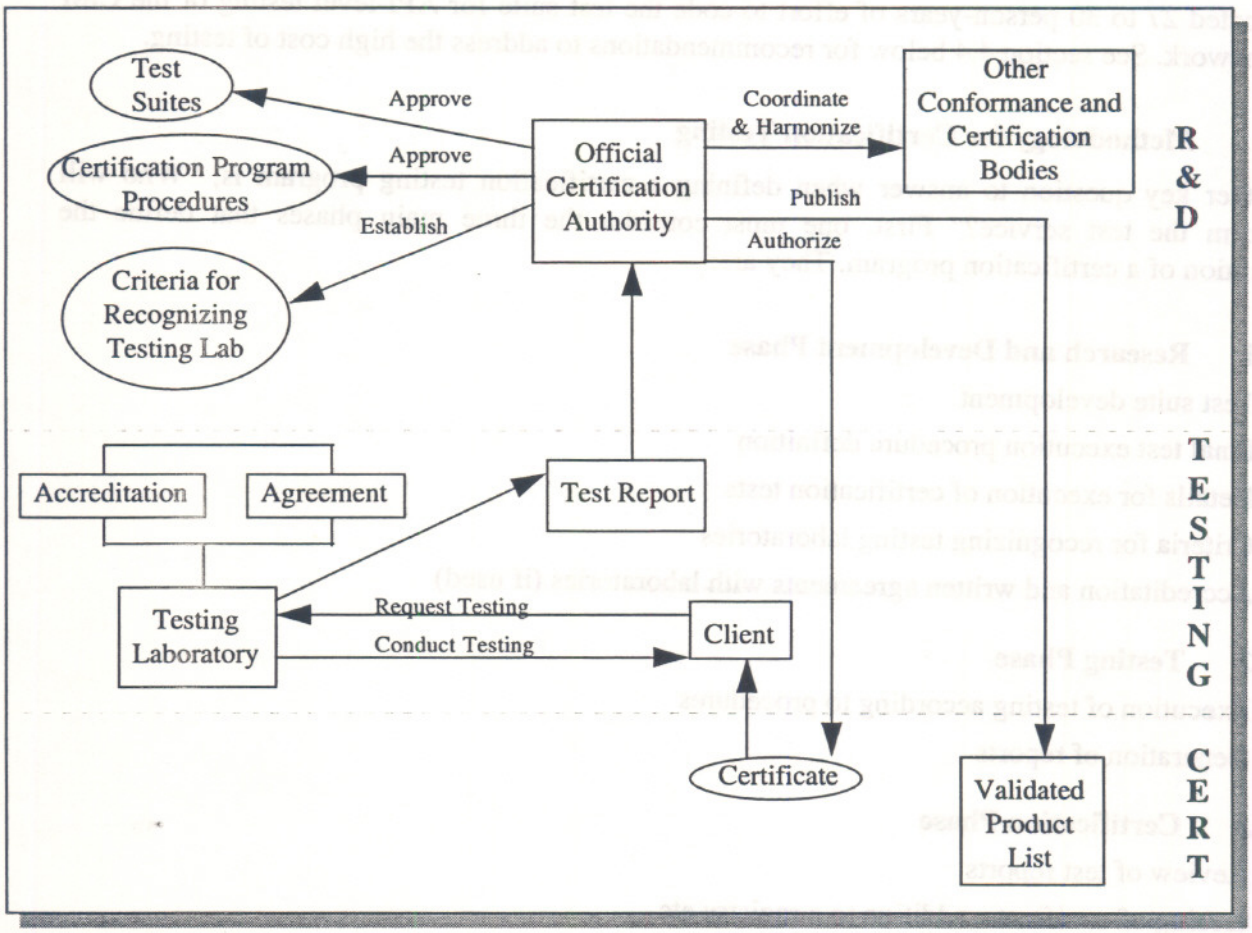

Figure 1 Certification Program Components

Figure 1 shows all possible tasks that might be involved in a certification program. Note that a given methodology may not incorporate every task in Figure 1. Also 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 its product. The official certifying body reviews the results of the supplier's testing and delivers the certificate indicating the results. 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 then would 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. An issue with this approach is that the laboratory needs to maintain competency and be reaccredited periodically, which can be costly. An advantage is that 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 instance might be if a foreign certification body such as the National Computing Centre (NCC) in the United Kingdom were to enter 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 be used only with a very mature standard.

In some cases the government performs the testing service, as with SQL. However, this approach requires an approved and funded program.

4.4 Certificate Contents

The certificate itself should contain at least the following information:

- Procedures followed (may include a detailed test plan)
- Versions (and models numbers) of all relevant software and hardware components used during testing
- Profiles of tests performed (e.g., CIM Framework components tested)
- Organization performing the testing
- Organization auditing the testing (if applicable)
- Evidence or reference to related standards conformance (as required)
- Overall pass/fail status

4.5 Summary of Recommendations on Testing

The authors recommend the following on testing:

- Suppliers should 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 conformant 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.

- It is strongly recommended 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 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, it is strongly recommend that the *backplane* approach be adopted. The backplane approach allows for minimal impact to the suppliers, and fits with the notion of framework services being provided to all applications and becoming absorbed into the operating system.
- The high cost of deploying a certification program should be highlighted to SEMATECH management so 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 dollars, however, is not the best approach to analyze the effort required. A better metric is the person-months required to generate the test suite and develop the certification program (see examples above).
- Methods for automatically generating tests should be explored further. Research projects and companies 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.
- Suppliers should be involved in test suite development to establish a consensus for the program. Also recommended is 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.
- A self-certification program, audited by SEMATECH, is recommended. 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.
- The project should leverage 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 does not indicate 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.

- Component dependencies, as indicated by the McGuire spreadsheet, should be included in the specification. This would allow suppliers to determine the minimum configurations required for interoperability.
- Assertions for the specification should be captured in a formal document that defines what is to be tested and how. This is closely related to exploring automated methods of test generation from the formal specification.
- SEMATECH, with help from NIST, should reach agreement with a group of semiconductor suppliers on the test suite. The test suite then should be developed by a small group of experts, preferably those most familiar with the CIM Framework. However, as large a group as possible should reach consensus on the acceptability of the test suite. This buy-in is required for the success of the program.

4.6 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 "back doors" that 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. This is due to the fact 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.

Customer Service
SEMATECH Technology Transfer
2706 Montopolis Drive
Austin, TX 78741

internet: info@SEMATECH.org
fax: (512) 356-3081
phone: (512) 356-SEMA