SEMI E81-0600
PROVISIONAL SPECIFICATION FOR CIM FRAMEWORK DOMAIN ARCHITECTURE

This provisional specification was technically approved by the Global Information & Control Committee and is the direct responsibility of the North American Information & Control Committee. Current edition approved by the North American Regional Standards Committee on March 2, 2000. Initially available at www.semi.org April 1999; to be published June 2000. Originally published June 1999.

SUPERSEDED

1 Purpose

1.1 This document is an overview of the structure and contents of a suite of documents representing an application framework for the Computer Integrated Manufacturing (CIM) systems as used in semiconductor factories. A framework is a software infrastructure that creates a common environment for integrating applications and sharing information in a given domain. The purpose of this framework is to establish an industry standard architecture for complex manufacturing systems, leading to an open, multisupplier CIM system environment. The framework described in this specification is called the CIM Framework.

2 Scope

2.1 The intent of this document is to describe the Manufacturing Execution Systems (MES) domain that is the subject of the CIM Framework and to provide a reference for concepts that are common to the set of documents that specify the CIM Framework. The *Provisional Specification for CIM Framework Domain Architecture* defines the structure, relationships and interworkings of the components that together comprise the CIM Framework. This architecture defines the partitioning of the CIM Framework components and the responsibilities of each of those components. It also specifies the common abstractions for manufacturing jobs, material, and factory resources that are used consistently throughout the CIM Framework as unifying themes.

2.2 The CIM Framework Domain Architecture does not address the dependencies on computing technologies needed to implement these components. These aspects apply more to the realization of the components as software artifacts than to their functionality in terms of semiconductor manufacturing concepts. The technical aspects of the CIM Framework architecture are captured in a separate document, SEMI E96, Guide for CIM Framework Technical Architecture.

2.3 This specification does not purport to address safety issues, if any, associated with its use. It is the responsibility of the users of this specification to establish appropriate safety and health practices and determine the applicability of regulatory limitations prior to use.

3 Limitations

3.1 The CIM Framework Specification must continue to evolve to meet the needs of a competitive and vital industry. The content of this framework represents a significant amount of real development experience from a number of commercial software suppliers and their customers. These specifications reflect the product architectures of those companies, as well as the requirements of their customers.

3.2 As a SEMI Provisional Standard, the Specification for CIM Framework Domain Architecture has specific deficiencies that must be addressed before it may be upgraded to full SEMI Standard status. These deficiencies are:

- Ensuring consistency with the details of subsequent related specifications that are based on this domain architecture.
- Evolving from coarse-grained component partitions to fine-grained components that provide substitutability of smaller components.
- Expanding interfaces to include build-time configuration functions.
- Providing fully validated models using the standard Unified Modeling Language (UML) notation.
- Aligning the CIM Framework representation of equipment and interfaces for interactions with equipment automation software with emerging standards in areas such as Object-Based Equipment Model (OBEM) and Automated Material Handling Systems (AMHS).
- Modifying the CIM Framework use of the “in” parameter mode and operation return value to include also the “out” and “inout” modes to better accommodate implementations based on Microsoft DCOM and IDL enhancements for pass-by-value of objects.

Documents Document Sharing Hub
Full document details: SEMI E81-0600

- Adjusting the functional partitioning of the Domain Architecture to reflect the final positioning of sub-components in the anticipated revisions of the other CIM Framework specifications.

4 Referenced Standards

4.1 *SEMI Standards*

SEMI E5 — SEMI Equipment Communications Standard 2 Message Content (SECS-II)

SEMI E10 — Standard for Definition and Measurement of Equipment Reliability, Availability, and Maintainability (RAM)

SEMI E30 — Generic Model for Communications and Control of Manufacturing Equipment (GEM)

SEMI E32 — Material Movement Management (MMM)

SEMI E42 — Recipe Management Standard: Concepts, Behavior, and Message Services

SEMI E58 — Automated Reliability, Availability, and Maintainability Standard (ARAMS): Concepts, Behavior, and Services

SEMI E86 — Provisional Specification for CIM Framework Factory Labor Component

SEMI E93 — Provisional Specification for CIM Framework Advanced Process Control Component

SEMI E96 — Guide for CIM Framework Technical Architecture

SEMI E97 — Provisional Specification for CIM Framework Global Declarations and Abstract Interfaces

SEMI E102 — Provisional Specification for CIM Framework Material Transport and Storage Component

4.2 *OMG Documents*[1]

CORBA — Common Object Request Broker Architecture, Version 2.3.1 (OMG Document formal/99-10-07).

MfgDTF — Manufacturing Domain Task Force Roadmap, Version 3.1 (OMG Document mfg/98-06-11).

OMA — Object Management Architecture Guide, Version 3.0 (OMG Document ab/97-05-05).

UML — UML Notation Guide, Version 1.1 (OMG Document ad/97-08-05).

Workflow — Joint Workflow RFP Revised Submission (OMG Document bom/98-06-07).

1 Object Management Group, 492 Old Connecticut Path, Framingham, MA 01701, USA

4.3 *SEMATECH Documents*[2]

CIMArch — Computer Integrated Manufacturing (CIM) Framework Architecture Concepts, Principles, and Guidelines, Version 1.0 (SEMATECH-Technology Transfer #97103379A-ENG).

CIMFW — Computer Integrated Manufacturing (CIM) Application Framework 2.0 (SEMATECH Technology Transfer #93061697J-ENG).

4.4 *Other References*

ALBUS — J.S. Albus and A.M. Meystel, *A reference model architecture for design and implementation of intelligent control in large and complex systems*, International Journal of Intelligent Control and Systems vol. 1, no.1 p.15–30, World Scientific: Singapore, March 1996.[3]

ANSI — ANSI Standard ANSI/ISA-S88.01-1995, Batch Control Part 1: Models and Terminology[4]

COM+ — http://www.microsoft.com/msj/1197/complus.htm; http://www.microsoft.com/com/[5]

DCOM — http://www.microsoft.com/windows/downloads/bin/nts/dcom_architecture.exe.[5]

JAVA — http://www.javasoft.com.[6]

WfMC — http://www.wfmc.org.[7]

NOTE 1: As listed or revised, all documents cited shall be the latest publications of adopted standards.

5 Terminology

5.1 *Abbreviations and Acronyms*

5.1.1 *AMHS* — Automated Material Handling System

5.1.2 *APC* — Advanced Process Control

5.1.3 *APCFI* — Advanced Process Control Framework Initiative

5.1.4 *API* — Application Programming Interface

5.1.5 *CIM* — Computer Integrated Manufacturing

5.1.6 *MES* — Manufacturing Execution System

2 SEMATECH, 2706 Montopolis Dr., Austin, TX 78741, USA

3 World Scientific Publishing Co., 1060 Main St., River Edge, NJ 07661, USA

4 American National Standards Institute, 11 West 42nd St., New York, NY 10036, USA

5 Microsoft, 10500 NE 8th St., Ste. 1300, Bellevue, WA 98004, USA

6 Sun Microsystems Inc., 901 San Antonio Road, Palo Alto, CA 94303, USA

7 Workflow Management Coalition Office, 2 Crown Walk, Winchester, Hampshire, S022 5XE, United Kingdom

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E81-0600

5.1.7 *MMMS* — Material Movement Management Services (SEMI)

5.1.8 *OBEM* — Object Based Equipment Model

5.1.9 *OMA* — Object Management Architecture

5.1.10 *PFC* — Process Flow Context

5.1.11 *PFI* — Process Flow Iterator

5.1.12 *RFP* — Request for Proposal

5.1.13 *RMS* — Recipe Management System

5.1.14 *UI* — User Interface

5.1.15 *WIP* — Work In Process

5.2 *Definitions*

5.2.1 *abstract interface* — an interface defined outside any component that generalizes common features of the CIM Framework. The abstract interfaces are intended for use in multiple components via interface inheritance mechanisms.

5.2.2 *application* — 1. One or more programs consisting of a collection of interoperating objects which provide domain specific functionality to an end user or other applications. 2. Functionality provided by one or more programs consisting of a collection of interoperating objects.

5.2.3 *application framework* — a framework that constitutes an application or a set of applications for a domain area.

5.2.4 *application interface* — the interface provided by an application or application program.

5.2.5 *application object* — an object implementing an application interface.

5.2.6 *architecture* — the structure of the components of a program/system, their interrelationships, and principles and guidelines governing their design and evolution over time.

5.2.7 *attribute* — an identifiable association between an object and a value. An attribute may have functions to set and retrieve its value.

5.2.8 *behavior* — the effects of performing a requested service including its results.

5.2.9 *binding* — a specific choice of platform technologies and other implementation-specific criteria.

5.2.10 *class* — the shared common structure and common behavior of a set of objects. Class often implies an implementation of the common structure and behavior while interface represents a specification of those common features.

5.2.11 *client* — an object that uses the services of another object by operating upon it or referencing its state.

5.2.12 *collection* — an object containing references to (collections of) other objects with services for managing them and providing access to them as a related group of objects.

5.2.13 *component* — a reusable package of encapsulated objects and/or other components with well-specified interfaces. The component is the element of standardization and substitutability in the CIM Framework.

5.2.14 *Computer Integrated Manufacturing (CIM)* — an approach that leverages the information handling capability of computers to manage manufacturing information and support or automate the execution of manufacturing operations.

5.2.15 *conformance* — adherence to a standard or specification in the implementation of a product, process, or service.

5.2.16 *conformance requirement* — identification in the specification of behavior and/or capabilities required by an implementation for it to conform to that specification.

5.2.17 *conforming implementation* — an implementation that satisfies all relevant specified conformance requirements.

5.2.18 *distributed system* — an integrated collection of several processing and memory components whose distribution is transparent to the user so that the system appears to be local.

5.2.19 *domain interface* — an interface specific to an application subject area.

5.2.20 *domain object* — an object implementing a domain interface.

5.2.21 *events* — an asynchronous message denoting the occurrence of some incident of importance. For example, state change or new object created.

5.2.22 *event channel* — the intermediate object that forwards published events to interested subscribers.

5.2.23 *exception* — an infrastructure mechanism used to notify a calling client of an operation that an unusual condition occurred in carrying out the operation.

5.2.24 *extensibility* — the ability to extend or specialize existing components and add new object classes or components while preserving architectural integrity and component conformance to standards.

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E81-0600

5.2.25 *framework* — a collection of classes or components that provide a set of services and functionality for a particular domain.

5.2.26 *implementation* — the internal view of a class, object or module, including any non-public behavior. The specific code and functionality that implements an interface.

5.2.27 *infrastructure* — the services, facilities, and communications mechanisms that support the collaboration between and lifecycle of distributed objects.

5.2.28 *inheritance* — a relationship among classes wherein one class (a subclass) shares the structure or behavior defined in one or more other classes (superclass). A subclass typically specializes its superclasses by augmenting or redefining existing structure and behavior.

5.2.29 *instance* — a software entity that has state, behavior and identity. The terms instance and object are interchangeable. An object is an instance of an interface if it provides the operations, signatures and semantics specified by that interface. An object is an instance of an implementation if its behavior is provided by that implementation.

5.2.30 *interface* — the external view of a class, object, or module that emphasizes its abstraction while hiding its structure and internal behavior. An interface definition ideally includes the semantics.

5.2.31 *interface inheritance* — the construction of an interface by incremental modification of other interfaces (see implementation inheritance). The CIM Framework specifies interface inheritance but not implementation inheritance.

5.2.32 *interoperability* — the ability for two applications or the parts of an application to cooperate. In the CIM Framework, interoperability requires that application components be able to share data, invoke each others' behavior (services), exchange events, and publish service exceptions.

5.2.33 *job* — some system level operation whose execution may be requested by an entity whose responsibility it is to manage jobs. The job concept is analogous to operations performed on the "factory floor" in a physical factory. There, operators are requested to perform operations (jobs) requested by their managing supervisors or some other managing source. A job often spans a significant amount of time and multiple resources within the system. In the CIM Framework, the job construct is intended for specialization to enable specific job supervisors and jobs to provide system solutions.

5.2.34 *lifecycle* — the life of an object, including creation, deletion, copy, and equivalence.

5.2.35 *method* — an operation upon an object defined as part of the declaration of a class. In general, the terms message, method and operation can be used interchangeably. Technically, a method is defined within a class and an operation is defined within the IDL. An operation is implemented by a method.

5.2.36 *object* — an identifiable encapsulated entity that implements one or more services that can be requested by a client. An instance of a class.

5.2.37 *object services* — interfaces for general services that are likely to be used in any program based on distributed objects.

5.2.38 *Object Management Group (OMG)* — an international consortium dedicated to the development of open specifications for distributed, heterogeneous, object-oriented systems.

5.2.39 *operation* — an operation is an entity, identified by an operation identifier that denotes a service that can be requested. An operation has a signature that describes the legitimate values of request parameters and returned results, including any exceptions.

5.2.40 *persistent object* — an object that can survive the process or thread that created it. A persistent object exists until it is explicitly deleted.

5.2.41 *process definition* — information characterizing manufacturing processes including an estimate for the time a process resource will be engaged in the process; process resource settings; and the process capabilities required for the process.

5.2.42 *process flow* — the part of a product specification that defines the sequence of process steps for the manufacturing of a specific product. The data structure for representing a process flow is the directed graph; specifically, a tree structure. The nodes of the tree are called process flow nodes (see below). Services are required to navigate the process flow.

5.2.43 *process flow context* — navigational information pertaining to a product's progress as it traverses its context process flow.

5.2.44 *process step* — the smallest unit of processing activity that can be defined in a process flow. One or more process steps are sequenced to define an operation set.

5.2.45 *recipe* — the pre-planned and reusable portion of the set of instructions, settings and parameters that determine how a job is to be performed. For example, recipes are used to describe Process Steps and are typically contained within a Product Specification.

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E81-0600

They determine the processing environment seen by a manufactured product (e.g., wafer). Processing recipes may be subject to change between product runs or processing cycles.

5.2.46 *sub-component* — a component that is fully contained within a larger component. The interfaces of the sub-component may be exposed or hidden by the encapsulating component.

5.2.47 *substitutability* — the ability to replace a given component from one supplier with a functionally equivalent component from another supplier without impacting the other components or its clients in the system.

5.2.48 *type* — a declaration that describes the common properties and behavior for a collection of objects. Types classify objects according to a common interface; classes classify objects according to a common implementation.

6 Overview

6.1 This section provides background information that will help readers get the most from the content of this specification.

6.2 *Intended Audience*

6.2.1 The framework specification is intended to address the needs of the following CIM technologists:

- Technical CIM managers.
- System architects and engineers.
- Application developers and integrators.
- Standards developers.

6.2.2 These groups may be found in a variety of organizations, including semiconductor manufacturers, software product suppliers, system integrators, equipment suppliers, standards organizations, universities, national laboratories, and other research organizations.

6.2.3 *Technical CIM Managers*

6.2.3.1 Technical CIM managers are responsible for managing the development, delivery, and integration of complex manufacturing software applications. They can use the CIM Framework specification to plan and organize the development activities and guide component testing and validation. Moreover, those who buy some of their software from external sources can use it as a purchasing guide when discussing system architecture and integration requirements with potential suppliers.

6.2.4 *System Architects and Engineers*

6.2.4.1 System architects and engineers are responsible for overall system design, including selection of industry standards for computing and communications infrastructure, software development processes, product roadmaps, and related topics. They can make extensive use of the CIM Framework as a starting point for many of their activities, including the

- partitioning and allocation of application functions to specific modules,
- definition of the boundary between the distributed system infrastructure and the rest of the system, and
- specification of open interfaces between the portions of the system they are designing and the external environment.

6.2.4.2 They can also use the CIM Framework specifications to define a strategic system roadmap for migration to an open, distributed system environment.

6.2.5 *Application Developers and Integrators*

6.2.5.1 Application developers and integrators must produce, install, and support software applications for semiconductor manufacturing. The CIM Framework specification, in conjunction with a specific framework "binding" (i.e., target computer system hardware and software technologies), represents a set of detailed design requirements for the application developer. At a minimum, the CIM Framework defines the scope and boundaries of the essential standard components of a manufacturing execution system, and can be used principally as an interface specification. The object models can also be used in the internal design of new applications and/or legacy integration "wrappers," accelerating the development process even further. Finally, the specification can form the basis for creating an independent set of tests necessary to verify conformance.

6.2.6 *Standards Developers*

6.2.6.1 Developers of CIM software standards are responsible for specifying the public interfaces and shared information models that allow the many software products found in a modern semiconductor factory to work together. They can use the CIM Framework as an open source of information for establishing precise definitions for the many items in a factory that must be represented in multiple suppliers' products, including

- standards for partitioning and communicating with complex equipment,
- product and raw material attributes and relationships,

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E81-0600

- process definitions and routings,
- equipment data sampling schemes and storage schema,
- personnel qualifications,
- and many others.

6.3 *CIM Framework Foundational Concepts*

6.3.1 This section provides an explanation of the basic concepts of framework, component, and sub-component as used in the CIM Framework Domain Architecture.

6.3.2 *Framework*

6.3.2.1 A framework is a software infrastructure that provides a common environment for the development and integration of applications and sharing of information in a given problem domain. The CIM Framework is a particular type of framework based on an object-oriented model of semiconductor wafer manufacturing. It specifies manufacturing objects and object interaction protocols that enable building semiconductor CIM applications from a framework of compatible, substitutable application components.

6.3.2.2 The heart of the framework is a set of semiconductor manufacturing abstractions (e.g., Wafer, Specification, Machine) and services (e.g., get wafer location, set specification parameter, get machine utilization) that are typically embodied in applications (e.g., material management, specification management, machine control). The implementations of these abstractions are delivered on distributed computer platforms (e.g., workstations, servers) which use standard software system technologies (e.g., communications, database, and user interface). The current CIM Framework is specifically targeted at manufacturing information management and control for both the planning and operational phases of semiconductor wafer fabrication.

6.3.3 *Component Architecture*

6.3.3.1 The CIM Framework specifies software functions that are common across MES applications and serve to integrate those MES applications into a coherent system. The CIM Framework software architecture is based on components. Components are software building blocks—"chunks" of functionality that make up software applications. By specifying standard interfaces and behavior of common MES components, manufacturers can assemble systems from components from multiple suppliers and they can evolve those systems by extending the common components and by substituting old components with improved components that implement the same interfaces and behavior in improved and extended ways. The CIM Framework defines a manufacturing execution system architecture whose components can be assembled in many ways and driven by many business processes and operational policies.

6.3.3.2 Figure 1 shows the CIM Framework architecture as a layered system, with the CIM Framework covering the middle layer of that system. Figure 2 details the layers, showing the components and their interaction and extension. The following subsections provide an introduction to the CIM Framework architecture.

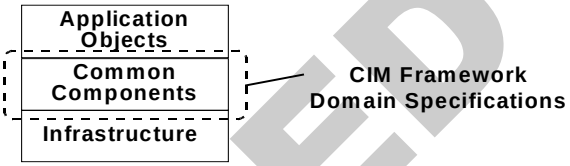

Figure 1
CIM Framework Layering

6.3.3.3 *Infrastructure*

6.3.3.3.1 The infrastructure provides the distributed computing environment for the application. These services include operating system, communications, data storage, user interface, event distribution, exception management, etc. The CIM Framework assumes infrastructure services and facilities defined by the Object Management Group®'s (OMG) Object Management Architecture (OMA) or by the Microsoft® DCOM and COM+ architecture and it can be mapped to other infrastructures such as those for Java™. SEMI E96 addresses the infrastructure layer.

6.3.3.4 *Common Components*

6.3.3.4.1 Common components are the functional entities common across MES applications. For example, material tracking, machine management, and scheduling applications all need a common, shareable concept of wafer groups (lots), machines, and process recipes. The common components provide a shared model for these entities, enabling quicker development and integration of material tracking, machine management, scheduling and other applications. They specify the data and behavior of these components required for interoperability between the applications.

6.3.3.5 *Application Objects*

6.3.3.5.1 The application objects provide the application functionality beyond the common components. These application objects provide application-specific data and behavior (such as the specific scheduling algorithms or the specific recipe management functions), building on the common component data and behavior that allows the application to interoperate with other applications. The application objects also

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E81-0600

define the business process workflows, business logic and user interfaces for the applications. They provide functionality that is often product- or site-specific. This functionality should not be included in an industry-wide standard for common components. Rather, it should be accommodated through the extendibility and reuse mechanisms of the common components.

6.3.3.6 *Component Granularity and Incremental Standard Conformance*

6.3.3.6.1 The CIM Framework components are the smallest elements of standardization of functional interface and behavior. The CIM Framework specifies relatively fine-grained components (in terms of their functional scope) as in the SEMATECH CIM Framework Specification Version 2.0 [CIMFW]. These components are larger than objects (their specification is in terms of an object model with typically three to five objects) but more fine-grained than traditional MES applications. However, the initial SEMI CIM Framework standards also identify components that are more coarse-grained, aligning with current MES product boundaries. These coarse-grained components contain fine-grained sub-components in their specifications, as in Figure 3 (typically two to four sub-components per coarse-grained component).

6.3.3.6.2 The coarse-grained components encapsulate the detail of the internal objects, relationships and sub-components by selectively exposing, hiding or abstracting some object methods and relationships. The coarse-grained components are specified with the detail of the sub-component and object interfaces and behavior, but standard conformance is in two levels; first-level conformance is to the interfaces of the coarse-grained components (not requiring exposure of the encapsulated detail), and second-level conformance is to the detail of the sub-components.

6.4 *CIM Framework Functional Scope*

6.4.1 The term Manufacturing Execution System (MES) represents an abstraction for a *collection* of software implementations. While there are examples of implementations that provide significant coverage of MES functionality, the industry trend is toward supplier focus on areas of core competency. In many cases this will result in a supplier offering for a subset of the MES domain, or a partitioned offering of separable products by a single source. Large, more monolithic implementations are gradually evolving toward this model of component packaging for smaller implementations. Ideally, MES scoping should correspond to natural boundaries that have emerged in representative products that border the "In MES"/"Outside MES" dividing line. It is that capability within MES scope that will be provided by the CIM Framework.

6.4.2 The following list identifies some criteria that may be used to help scope MES within the larger context of manufacturing enterprise systems often called Computer Integrated Manufacturing.

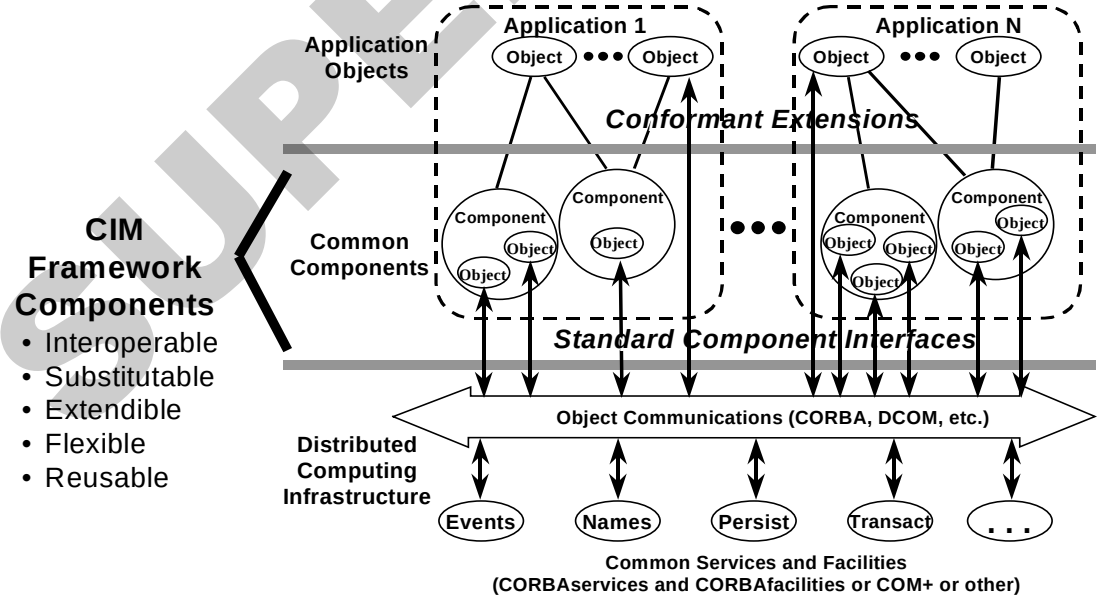

Figure 2
CIM Framework Component Architecture

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E81-0600

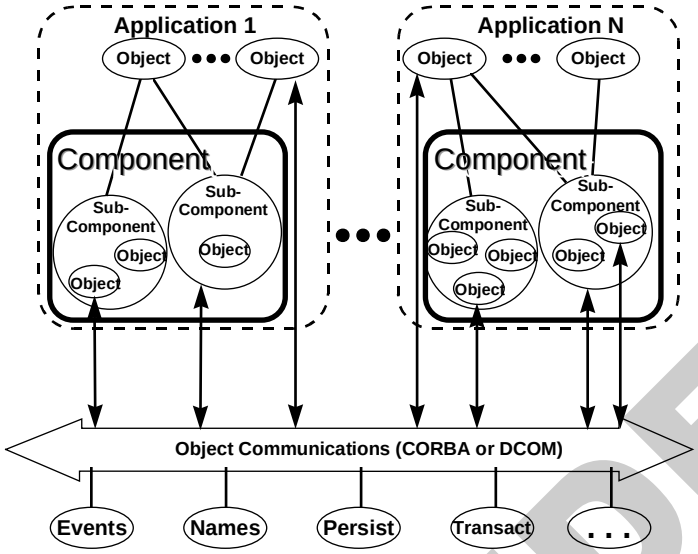

Figure 3
Components and Sub-Components

6.4.3 Thus, a component is "In MES" scope and within the CIM Framework scope, if

- it provides a job abstraction to manage work currently in progress across the manufacturing facility,
- it represents the convergence of product and process specifications, material, and manufacturing resources through execution of production jobs,
- it provides facility level planning and scheduling of manufacturing production activities,
- it provides access to historical data and reporting of occurrences that changed the state of the products, the production facility or its resources,
- it allows coordinated actions to control factory resources,
- it allows abstraction representations of production facilities and their resources,
- it enables automated update of manufacturing parameters (settings) through data collection and analysis of manufacturing processes, and
- it supports quality management through capture of key metrics (e.g., yield, throughput, cycle-time and utilization).

6.4.4 A component is "Outside MES" scope, thus outside CIM Framework scope, if

- it controls or manipulates the internal state or operation of a piece of manufacturing equipment,
- it deals with the business interactions between the manufacturing enterprise and external enterprises such as customers or suppliers,
- it manipulates the product or process definition with a focus on product design rather than execution of the manufacturing process,
- it focuses on the creation and manipulation of what-if models of factory or product state,
- it isn't directly concerned with transforming material from an initial (raw or partially completed) state to a more valuable product, and
- it is primarily used in support of laboratory analysis that is not directly integrated into the manufacturing process (e.g., off-line metrology).

6.4.5 Examples applying the MES definition and scoping criteria might be derived from the following high-level interactions.

6.4.5.1 *"In MES" Scope*

- A product request for production of goods (partial or finished) is offered to one facility which responds with a delivery commitment.

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E81-0600

- A production job is created, along with a grouping structure for the target product.
- A production job is split into subjobs and maybe merged again.
- A production job is rerouted to a different resource due to specific circumstances.
- A production job is assigned to a set of resources, allocated required material, and dispatched for execution.
- A change in state of product, resources or material resulting in a change to delivery commitments is reported.
- A change to a manufacturing process results in alteration of the execution of a production job.
- A production job is assigned to specific set of manufacturing machines due to their machine resource capabilities.
- A value for a process specification is changed due to gathered values of a influencing quality control process.
- Material is made available for use or is moved to a new physical location within the facility.
- Material is exchanged between different positional containers due to e.g., contamination control (cleaning of a positional container).
- A manufacturing machine is taken down for maintenance and becomes unavailable for job execution.
- A resource of a multiple resource machine is taken down and becomes unavailable, but not the whole machine.

6.4.5.2 *"Outside MES" Scope*

- A product is defined and engineered for production — Product and Process Engineering.
- A facility is qualified to produce a given product — Release for Production.
- An order is released for production — represented as a demand from Order Management System.
- A consumable is running out of stock — an order is released for delivery to the facility — Enterprise Resource Tracking.
- The facilities (or series of facilities) capable of producing the needed product are analyzed and the demand is matched with available supply (including capacity for future production) — Enterprise Planning System.
- A piece of manufacturing equipment obtains a recipe and enacts a manufacturing process on material — Equipment Automation.
- Material movement equipment controls the transport of material to a specified destination — Equipment Automation.
- Modeling data representing a hypothetical change in factory state is manipulated and analyzed to determine the effects of the changes — Modeling and Simulation.
- Material is packed for transport from the front-end facility to the back-end facility and a carrier is notified of shipping order — Transportation Logistics.
- A customer order is divided into two product requests involving two facilities to meet the requested delivery date — Release for Production.
- The layout of equipment locations within the factory is modified to accommodate a new tool — Factory Design.

6.4.6 Multiple levels of packaging framework functionality are supported by the specification so that a variety of applications from multiple suppliers with potentially intersecting capabilities can be accommodated. This means that, given a specific framework binding to a set of computer and software system technologies, an application can be instantiated and executed in this environment and will register itself as a set of well-known objects that provide a core set of framework specified services.

6.4.7 The CIM Framework specification defines a set of functional components designed to work together to form an integrated manufacturing system. The CIM Framework components cover the functionality of Manufacturing Execution Systems (MES). MES is a factory-wide function that drives material processing on equipment to produce products, and it manages all the resources to accomplish this.

6.4.8 Figure 4 shows the functional scope of the CIM Framework. The left-hand side represents engineering aspects of manufacturing systems, such as configuration, while the right-hand side indicates where MES falls within manufacturing operations. The CIM Framework covers operations of manufacturing execution to a larger degree and configuration to a lesser degree.

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E81-0600

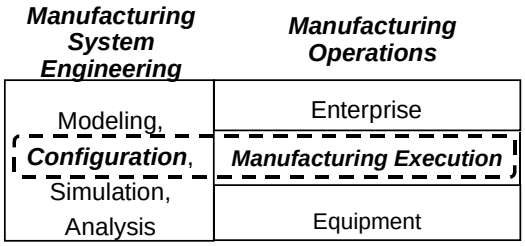

Figure 4
Boundaries of the CIM Framework: Manufacturing Execution

6.4.9 The CIM Framework specifies the MES functional responsibilities for semiconductor wafer fabrication. It does not include responsibilities for semiconductor packaging, assembly and test, but it was designed to accommodate extensions into this phase of semiconductor manufacturing.

6.4.10 Using the definition of component as found in Section 6.3.3, the components of the CIM Framework provide the following functionality found within MES configuration and execution:

- Product Management: manages product material and material groups.
- Durables Management: manages durable materials such as reticles and material carriers.
- Consumables Management: manages consumable materials such as gases and chemicals.
- Specification: manages product specifications and process flows.
- Factory Operations: drives efficient use of factory resources to manufacture products while meeting overall factory objectives.
- Scheduling: supports factory operations and other functions to optimize use of resources in manufacturing products and meeting factory objectives.
- Equipment Tracking and Maintenance: tracks equipment state and usage and manages equipment maintenance.
- Production Machine: manages execution of manufacturing process steps on process and metrology equipment.
- Recipe Management: provides factory-wide management and use of processing recipes.
- Advanced Process Control: executes run-to-run process control and fault detection strategies.
- Material Transport and Storage: moves and stores material.
- Factory Labor: manages labor resources and qualifications.
- Factory Services: provides support functions common across components, such as access security, document management/change control, and history storage and retrieval.

6.4.11 Section 7 lists the functional responsibilities of each of these components by MES configuration (called build-time) and execution (called run-time). The separation of the CIM Framework into components reflects anticipated boundaries of MES products, enabling integration of products from multiple suppliers into a coherent, integrated MES.

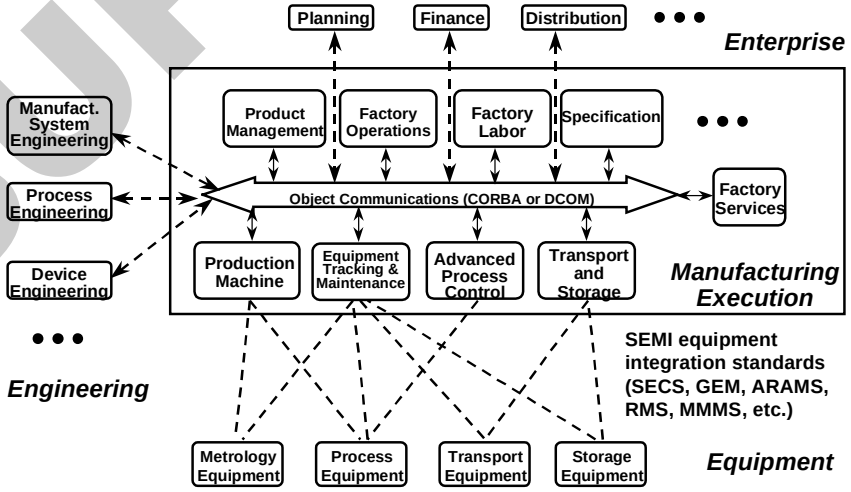

Figure 5
MES Component Integration

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E81-0600

6.4.12 As Figure 5 shows, the CIM Framework specifies MES interfaces enabling integration between MES components, that is integration within the MES. The CIM Framework does not directly address integration between the MES and external functions. External functions, such as enterprise functions and engineering functions, can interact with the MES using the CIM Framework-specified intra-MES interfaces and communications mechanisms. Interaction between MES functions and manufacturing equipment can use existing SEMI equipment integration standards such as SECS II (SEMI E5), GEM (SEMI E30), ARAMS (SEMI E58), RMS (SEMI E42), and MMMS (SEMI E32). Equipment can also implement CIM Framework components such as Production Machine and Equipment Tracking and Maintenance within the equipment, integrating with the MES (the equipment host) through CIM Framework component interfaces.

6.4.13 The Object Management Group's Manufacturing Domain Task Force is adopting standards for Enterprise, MES, Machine Control and Engineering applications (MfgDTF). Other related activity within the OMG includes its efforts regarding workflow (Workflow) and component architectures. SEMI tracks these through a liaison with the OMG.

6.4.14 The CIM Framework focuses on the common software functions that serve to integrate MES. It specifies components for factory resources, materials, specifications and factory control, and it establishes a functional and technical architecture so the components can be assembled to meet many functional needs and driven by company-specific business process, workflows and operation rules. The CIM Framework enables, but does not define, workflow and business logic to drive component behavior, nor does it define user interfaces (including decision support and reporting). Workflow, business logic and user interface are considered use-specific and thus outside an industry consensus standard. In addition, the CIM Framework specifies application interfaces to the MES functions, but does not specify the implementation of those functions.

6.4.15 *What Is in the CIM Framework Specification*

6.4.15.1 This section provides additional information about the CIM Framework suite of documents. A brief description of each document is given below.

6.4.15.2 *Specification for CIM Framework Domain Architecture (this document)*

- Contains introductory information and an overview of the CIM Framework specifications.
- Specifies the functional partitioning and functional responsibilities of CIM Framework components.
- Specifies common architecture patterns that functionally integrate CIM Framework components.
- Provides graphical and textual specification notation conventions for all CIM Framework components.
- Provides citations for related specifications or documents referenced within the CIM Framework.

6.4.15.3 *SEMI E96, Guide for CIM Framework Technical Architecture*

- documents the CIM Framework component interoperability architecture based on distributed computing standards including OMG CORBA® and Microsoft DCOM.

6.4.15.4 Additional documents detail the CIM Framework specifications. The document structure will reflect the functional partitioning of the domain architecture. Except for reorganization, the details will remain substantially the same as contained within the CIM Framework Blue Ballots with some requested extensions. The structure of these documents has been influenced by SEMATECH's CIM Framework Specification [CIMFW]. These companion documents represent the actual specification; whereas this document serves primarily as their introduction.

7 Requirements

7.1 This section specifies the functional requirements of CIM Framework components. It establishes a functional partitioning of CIM Framework components and it defines functional architectures for resource tracking and maintenance and for factory control. It also establishes the graphical and textual notations required to specify the CIM Framework components. The list below is a short overview of this section.

- Global and Abstract Definitions Group
- Factory Services Group
- Product Management Component
- Durables Management Component
- Consumables Management Component
- Specification Component
- Factory Operations Component
- Scheduling Component
- Equipment Tracking and Maintenance Component
- Production Machine Component
- Recipe Management Component

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E81-0600

- Advanced Process Control Component
- Material Transport and Storage Component
- Factory Labor Component

7.2 *Functional Partitioning*

7.2.1 This section overviews the functional responsibilities of the CIM Framework domain components. Details of the responsibilities are specified in the individual CIM Framework domain component specifications.

7.2.2 The domain component responsibilities cover two types of functions:

- Build-time, or configuration aspect of manufacturing systems engineering in Figure 4, which defines and configures the specific entities in the component, such as the specific machines and their state models or the wafer carriers and their initial locations.
- Run-time, or the execution aspect of manufacturing operations in Figure 4, which executes the component behaviors, maintains inter-component relationships, maintains component state and history and provides run-time access to component state and behavior.

7.2.3 The initial CIM Framework domain standards do not address the build-time functions of the components. These will be added as the standards move from provisional status to full standards. However, this functional partitioning section does identify build-time responsibilities, permitting these build-time functions to be properly allocated to the correct component as the CIM Framework evolves.

7.2.4 The components partition CIM Framework functionality. The components are designed to be exclusive: no functionality exists in more than one component. The only exception is that there are some cross-system functions that are used, specialized or implemented by most or all of the other components. These cross-component domain functions are in separate groups, called Global and Abstract Definitions Group and Factory Services Group.

7.2.5 The list of responsibilities for a complete MES solution is far more than the responsibilities of the CIM Framework components. The CIM Framework focuses on those functions that serve to integrate applications. Suppliers and users will provide functional responsibilities in addition to the standard responsibilities. The CIM Framework component responsibilities are also limited by focusing the scope of the CIM Framework on the run-time needs for wafer fabrication MES but to not cover the full scope of CIM solutions for the full semiconductor manufacturing process.

7.2.6 Sections 7.2.7 through 7.2.20 present the responsibilities for each of the CIM Framework components and the groups. Example terminology specific to semiconductor manufacturing is provided for clarity. It does not preclude application specialization for other industries.

7.2.7 *Responsibilities of Global and Abstract Definitions Group*

7.2.7.1 The Global and Abstract Definitions group (defined in SEMI E97) specifies the definitions of abstract data types and abstract interfaces used throughout the CIM Framework. These specifications are separated into a distinct group to enable them to be specified once and then logically included or inherited wherever they are subsequently needed.

7.2.7.2 *Global Definition Responsibilities*

- Provide type definitions for common data structures to ensure consistent representation. These items include data types for common concepts such as coordinates, priorities, timestamps, and sequences of basic data types.
- Provide type definitions for representations of historical occurrences.
- Provide definitions for common exceptions used consistently throughout the CIM Framework.

7.2.7.3 *Abstract Interface Responsibilities*

- Material architecture: the functions common to identifying, grouping, moving, locating and tracking material in the factory. The architecture is specialized for product material, durables and consumables. See Section 7.3.2.
- Factory resource architecture: the functions common to defining, organizing, tracking usage of and maintaining factory resources including equipment, sensors, durables, and people. See Section 7.3.3.
- Job architecture: the functions common to creating, executing and managing work in the factory. The job architecture is specialized for material processing jobs, material transport jobs, resource maintenance jobs and factory jobs that drive product material through their process flows. See Section 7.3.4.

7.2.8 *Responsibilities of Factory Services Group*

7.2.8.1 The Factory Services group (which is not actually a component) specifies domain facilities to support all domain components. The components can

SUPERSEDED

Documents
Document Sharing Hub
Full document details: SEMI E81-0600

separately implement these facilities for their own use, they can share a common facility across components, or use a combination of separate and shared implementations.

7.2.8.2 *Services in Factory Services Group*

- Document management and version management: functions common to creating, storing, retrieving, and changing (under change control and version management) documents such as process recipes, process flow specifications, and maintenance specifications.
- Event broker: support for components to publish, subscribe to and filter events.
- Access security control: support for components to limit access of system functions to authorized users and applications.
- Component management: functions for CIM Framework component registration, start-up, shutdown, etc.
- History: supports the ability of factory objects to configure and maintain histories (time-based sequences of data and events) and to support access to those histories.
- Data collection: the functions common to data collectors, including equipment, sensors, and manual data entry.

7.2.9 *Responsibilities of Product Management Component*

7.2.9.1 The Product Management component captures the state of all product material (work in progress). For wafer fabrication, this includes wafer and die-level tracking to support multiple products on a single wafer and to track known-good-die through wafer probe. The component also tracks engineering wafers and test wafers, even though they may not become "product." The component implements the common material architecture functions so product material can be identified, grouped (product groups, lots, process groups, transport groups), moved and located in the factory.

7.2.9.2 *Build-time Responsibilities of Product Management Component*

- Configure inventory regions.
- Configure product tracking histories, including what data to store and data archival mechanisms.

7.2.9.3 *Run-time Responsibilities of Product Management Component*

- Record and support access to current and historical product information, including:
  - Where material is (and has been) in its process flow.

    NOTE 2: The decision to advance material along its process flow is in the Factory Operations component.
  - Product state (processing, on hold, scrapped, etc.).
  - Material type (product, engineering, filler, etc.).
  - User-defined product information.
  - Genealogy, batching, allocation to product requests, the number of times reworked, etc.
  - Historical information on what equipment processed the material, when, with what process flow specification, recipe and durables, and by what operator.
  - Relations and history to related information in other components, including material location, material container, process flow specification, inventory region, process run data, recipe, etc.
- Notify other components of material information changes.
- Support creation, maintenance and historical recording of product groups and relationships among product groups (such as lots, transport groups and process batches) including splits and joins (and resulting product genealogy), experiments (planned future splits and processing changes), hold, rework, etc.

  NOTE 3: The decision to split, join, batch, move, rework, etc. is a responsibility outside of the Product Management component. Product Management records the results of the decisions.
- Record and support access to material location at multiple levels of detail: an area, a specific machine, a specific load port on a machine, a specific zone in a stocker, a specific wafer slot in a carrier, or a specific wafer coordinate of a die.
  - Including reference to material containers (cassettes, pods, etc.) holding the product (the containers are managed by the Durables Management component).

    NOTE 4: Material location is served by the Product Management component and the Material Transport and Storage component. The Material Transport and Storage component is responsible for physically moving material and recording its location, and the

SUPERSEDED

Documents Document Sharing Hub
Full document details: SEMI E81-0600