4.1.32.4 Table 14 lists the requirements for fundamental compliance to ARAMS and the section references to ARAMS or (where applicable) the standards where these capabilities are defined. These requirements shall all be satisfied for equipment to be termed ARAMS-compliant.

Table 14 Table 14 Fundamental ARAMS Requirements

Requirement	*Reference*
Event Notification	GEM 4.2.1.1 or ERS
Clock Services	GEM 4.10 or CTMC 8.7
Read-Only Data Access	GEM 4.2.2 or OSS 12.1
User-Configurable Data Access	GEM 4.5 or OSS 12.1
Alarm/Exception Management	GEM 4.3 or EMS
ARAMS State Model	ARAMS 8.3
ARAMS State TransitionNotification	ARAMS 12, 16.1
ARAMS Substate Codes	ARAMS 9
ARAMS Status Data	ARAMS 11.2
ARAMS Constant Data	ARAMS 11.3, 11.4
ARAMS Event Report Data	ARAMS 12, GEM 4.2.1.1, GEM 4.2.1.2.
Host State Change Request	ARAMS 15.1, 15.2, 15.5
Estimation of Powerdown Time	ARAMS 16.2
ARAMS Behavioral Requirements	ARAMS 16 (all except 16.3.2, 16.7.4, 16.8.2)

4.2 *Additional Capabilities* — The following capabilities are not required for fundamental compliance to ARAMS. They are recommended for equipment that provide a local operator interface and are not expected from equipment such as individual process modules of a cluster tool.

14.1.15 *User-Configurable Powerup State* — Provision of the user-programmable variable PowerupState, which allows the user to select either STANDBY or UNSCHEDULED DOWNTIME as the powerup state following powerdowns that occurred during STANDBY. (See Section 16.3.2.)

14.1.16 *User-Configurable Fault Recovery to Manufacturing* — Provision of user-programmable variables, PrdRecovery and SbyRecovery, that allows the user to enable or disable automatic equipment-initiated recovery from faults that clear without user intervention. (See Section 16.7.4.)

14.1.17 *Accumulator Data* — Support for the variables defined in Section 11.5, together with access to this data by the operator (Section 14.1) and the host, with support for the ResetAccumulators service. (See Sections 11.5 and 15.6.)

14.1.18 *User-Generated ARAMS Substate Table(s)* — The ability to accept one or more sets of user-generated ARAMS Substate Codes, as defined in ARAMS Substate Tables, that may be selected by the user in a request to change to a new state or substate. This includes support of the TableSend and TableRequest services and methods for operator access. (See Sections 10.3, 14.2, and 15.3.)

14.1.19 *Equipment-Generated ARAMS Substate Table(s)* — Provision of a set of equipment-defined ARAMS Substate Codes that can be selected by the user in a request to change to a new ARAMS state or substate. This includes support of the TableRequest service. (See Sections 10.3 and 15.4.)

14.1.20 *User-Generated Symptom Table(s)* — The ability to accept one or more user-defined ARAMS Symptom Table, allows the operator to select a symptom when requesting a transition to a new ARAMS state, and the associated data. This includes support for the TableRequest service and methods for operator access. (See Sections 10.4, 14.3, and 15.3.)

14.1.21 *Human Interface Requirements* — Operator access to ARAMS tables as defined in Sections 14.2 through 14.4. Where color is used, conformance to color code in Section 14.5.

14.1.22 *Equipment-Selected Substates* — Support for the user-configurable variable SubstateSelect (Section 11.4.5) allowing the user to enable and disable the capability to select substates of PRODUCTIVE and STANDBY. Documentation of the prioritization of substate selection. (See Section 16.6).

14.1.23 *User-Configurable Fault Detection in ENGINEERING* — Support for Transitions 12 and 13 (Section 16.7.4) and the user-configurable variable EngInterrupt (Section 11.4.1) allowing the user to enable and disable Transitions 12 and 13.

14.1.24 *User-Configurable Fault Recovery to ENGINEERING* — Support for automatic recovery (Transition 13, Sections 16.8.1 and 16.8.2) and the user-configurable variable EngRecovery (Section 11.4.2) allowing the user to enable and disable automatic recovery in Transition 13.

4.2.1.1 Table 15 provides the section references for additional ARAMS capabilities.

Table 15 Table 15 Section References for Additional ARAMS Capabilities

Capability	*Section Reference*
User-Configurable Powerup State	ARAMS 11.4.4, 16.3.2
User-Configurable Fault Recovery	ARAMS 11.4.2, 11.4.5, 11.4.6, 16.8.2
Accumulator Data	ARAMS 11.5, 15.5
User-Generated ARAMS Substate Table(s)	ARAMS 10.1, 10.2, 10.3, 14.3, 15.3
Equipment-Generated ARAMS Substate Table(s)	ARAMS 10.1, 10.2, 10.3, 15.4
User-Generated ARAMS Symptom Table(s)	ARAMS 10.1, 10.2, 10.4, 14.3, 14.4, 15.3
Human Interface Requirements	ARAMS 14.2, 14.3, 14.4, 14.5
Equipment-Selected Substates	ARAMS 11.4.7, 16.6
User-Configurable Fault Detection in ENGINEERING	ARAMS 11.4.1, 16.7, 16.7.4
User-Configurable Fault Recovery to ENGINEERING	ARAMS 11.4.2, 16.8
Human Interface Requirements	ARAMS 14, 14.1, 14.2.

14.1.25 *Human Interface Requirements* — Operator access to ARAMS data and state change, defined in Sections 14, 14.1, and 14.2.

14.2 *Requirements for Compliance* — Table 16 provides a checklist for ARAMS compliance.

Table 16 Table 16 ARAMS Compliance Statement

Fundamental ARAMS Requirements	*Implemented*	*ARAMS Compliant (See NOTE 1.)*
Event Notification	❑ Yes ❑ No	❑ Yes ❑ No
Clock Services	❑ Yes ❑ No	❑ Yes ❑ No
Read-Only Data Access	❑ Yes ❑ No	❑ Yes ❑ No
User-Configurable Data Access	❑ Yes ❑ No	❑ Yes ❑ No
Alarm/Exception Management	❑ Yes ❑ No	❑ Yes ❑ No
ARAMS State Model	❑ Yes ❑ No	❑ Yes ❑ No
ARAMS State Transition Notification	❑ Yes ❑ No	❑ Yes ❑ No
ARAMS Substate Codes	❑ Yes ❑ No	❑ Yes ❑ No
ARAMS Status Data	❑ Yes ❑ No	❑ Yes ❑ No
ARAMS Constant Data	❑ Yes ❑ No	❑ Yes ❑ No
ARAMS Event Report Data	❑ Yes ❑ No	❑ Yes ❑ No
Host State Change Request	❑ Yes ❑ No	❑ Yes ❑ No
Estimation of Powerdown Time	❑ Yes ❑ No	❑ Yes ❑ No
ARAMS Behavioral Requirements	❑ Yes ❑ No	❑ Yes ❑ No
Additional Capabilities	*Implemented*	*ARAMS Compliant (See NOTE 2.)*
User-Configurable Powerup State	❑ Yes ❑ No	❑ Yes ❑ No
User-Configurable Fault Recovery to Manufacturing	❑ Yes ❑ No	❑ Yes ❑ No
Accumulator Data	❑ Yes ❑ No	❑ Yes ❑ No
User-Generated ARAMS Substate Table(s)	❑ Yes ❑ No	❑ Yes ❑ No
Equipment-Generated ARAMS Substate Table(s)	❑ Yes ❑ No	❑ Yes ❑ No
User-Generated ARAMS Symptom Table(s)	❑ Yes ❑ No	❑ Yes ❑ No
Equipment-Selected Substates	❑ Yes ❑ No	❑ Yes ❑ No
User-Configurable Fault Detection in ENGINEERING	❑ Yes ❑ No	❑ Yes ❑ No
User-Configurable Fault Recovery to ENGINEERING	❑ Yes ❑ No	❑ Yes ❑ No
Human Interface Requirements	❑ Yes ❑ No	❑ Yes ❑ No

NOTE 1: Do not mark YES unless all fundamental ARAMS requirements are implemented.

NOTE 2: Additional capabilities may not be marked ARAMS-compliant unless all fundamental ARAMS requirements are implemented.

15 ARAMS States for Multi-Module Equipment

4.3 The preceding sections define how ARAMS is to be supported by simple equipment. Simple equipment includes equipment with at most one process chamber, and a single process capability, where individual modules are not treated separately from the equipment.

NOTE 22: Process capability, in this context, refers to the factory's manufacturing process. Equipment with more than one process capability may be used in different ways at different steps, typically through different process recipes. Such equipment may be "available" for one process but not for another. This type of complexity is neither addressed nor affected by SEMI E10 or by ARAMS.

4.4 This section addresses the application of ARAMS to complex equipment, including modular equipment where individual modules may be in different ARAMS states/substates. Complex equipment includes cluster tools and any other type of equipment that is organized into separate subsystems that can be addressed individually. In this case, it is advantageous for each module or subsystem to be given its own ARAMS state model. In addition, the overall equipment system itself has an ARAMS state model. This situation is illustrated in Figure 6.

4.5 The complexities of possible interactions between ARAMS states of the individual modules and the ARAMS state of the cluster tool as a whole are beyond the scope of this document. The following approach is recommended:

- Each module complies to fundamental requirements for the ARAMS state model, data variables, and message services.
- The integrated cluster tool complies to requirements for the ARAMS state model, data variables, and message services.
- The set of all the ARAMS models above are simultaneously active, as represented in Figure 6.
- The relationships between the ARAMS state for the cluster tool and the ARAMS states for the individual modules are user-configurable wherever possible. (Certain relationships between the cluster and critical modules, such as central wafer handler or central load lock, may not be configurable: e.g., if the critical module is down, the cluster is down.)

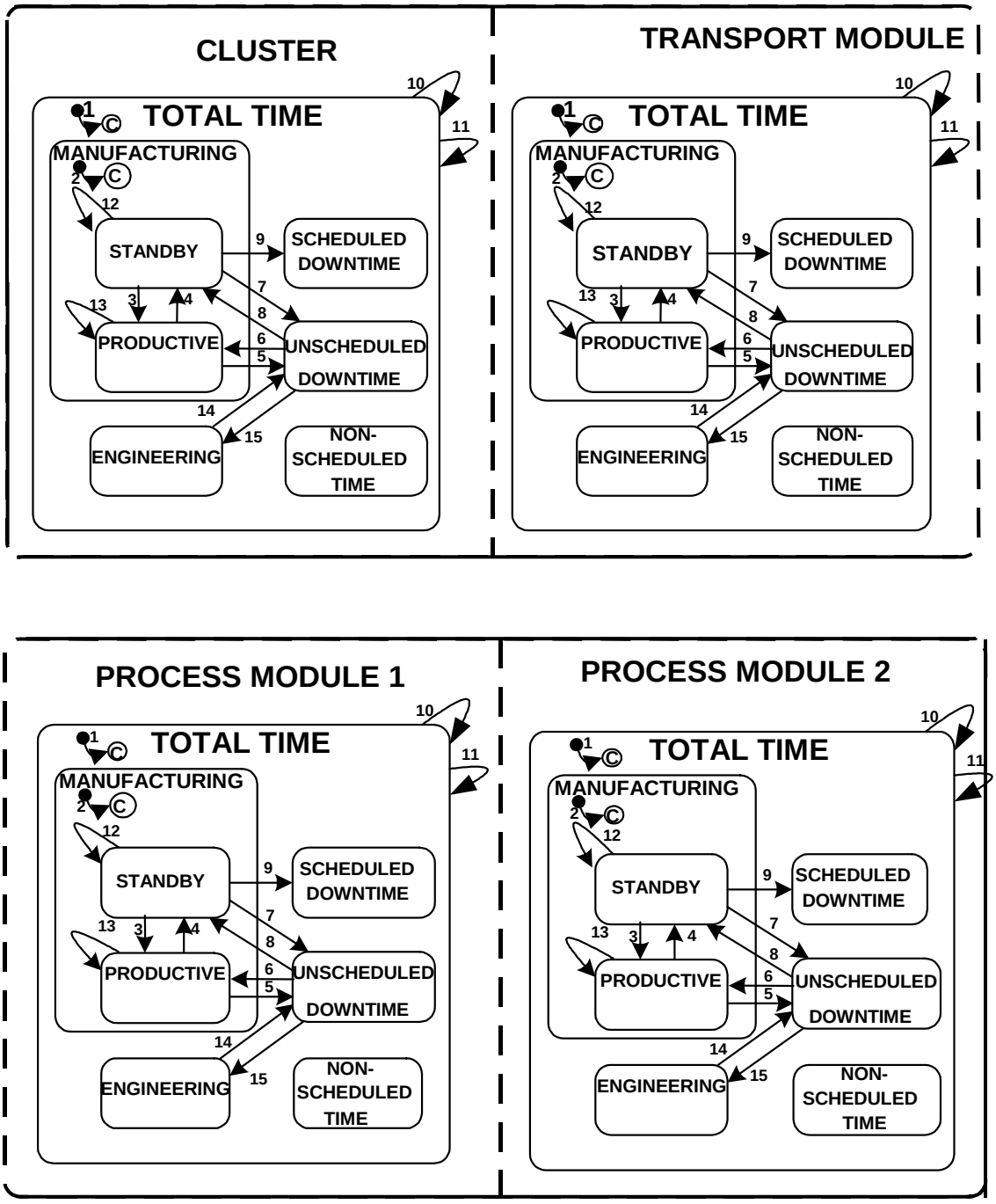

Figure 6
ARAMS Model for Cluster

NOTICE: SEMI makes no warranties or representations as to the suitability of the standards set forth herein for any particular application. The determination of the suitability of the standard is solely the responsibility of the user. Users are cautioned to refer to manufacturer's instructions, product labels, product data sheets, and other relevant literature respecting any materials mentioned herein. These standards are subject to change without notice.

The user's attention is called to the possibility that compliance with this standard may require use of copyrighted material or of an invention covered by patent rights. By publication of this standard, SEMI takes no position respecting the validity of any patent rights or copyrights asserted in connection with any item mentioned in this standard. Users of this standard are expressly advised that determination of any such patent rights or copyrights, and the risk of infringement of such rights, are entirely their own responsibility.

RELATED INFORMATION 1

NOTE: This related information is not an official part of SEMI E58 and is not intended to modify or supersede the official standard. Rather, these notes are auxiliary information provided as background or examples of possible application and are included as reference material. The standard should be referred to in all cases. SEMI makes no warranties or representations as to the suitability of the material set forth herein for any particular application. The determination of the suitability of the material is solely the responsibility of the user.

This section describes different applications of ARAMS, including an example of a method for estimating the time of powerdown as well as various scenarios.

R1-1 Estimating Powerdown Time

R1-1.1 This section describes a method for estimating the time at which a loss of power, reset, or re-boot occurs. Estimation of this time, allows the accumulation of time-in-state for the previous ARAMS state to be maintained.

R1-1.1.1 The equipment maintains a date/time value, PowerdownTime, that is used to estimate the time when a loss of power occurred (see Section 11.2.12). In this example, the equipment provides an additional user-configurable variable, UpdatePeriod, that defines the number of seconds in an update period. UpdatePeriod is used to set an interval timer. PowerdownTime is updated with the current date and time at the end of each timed interval.

R1-1.1.2 Whenever the equipment enters the initialization state, whether through powerup, reset, or re-boot, it uses the value found in PowerdownTime as the estimate of when the powerdown occurred. If a state change occurs as the result of a powerdown, then the time that has elapsed since the point of powerdown is used to update the appropriate time-in-state accumulator for the previous ARAMS state.

R1-1.1.3 For an example of the sequence of events and actions following a powerup, see Section R1-2.

R1-1.1.4 Figure R1-1 provides a simple logic flowchart as an illustration of the update process.

R1-1.2 *UpdatePeriod* — The length of the time interval, in seconds, for updating PowerdownTime. Form: positive integer.

R1-2 Powerup Scenario

R1-2.1 This scenario is analogous to a timing diagram. It illustrates the actions that occur at different times in sequence, showing the effect of a powerdown that occurs while the equipment was in the PRODUCTIVE state. In this example, accumulators defined in Section 11.5 are supported. UpdatePeriod is defined in the prior section and is set to a value of 30 seconds.

R1-2.2 In this example, the equipment received instructions for processing prior to the start of the timing and waited briefly for the material to arrive. Timing shown in this example begins when it receives the expected material and transitions from STANDBY to PRODUCTIVE at time = t_0. After a sequence of its normal periodic updates ($t_i = t_{i-1}$ + UpdatePeriod) of the estimated time of powerdown, it loses power while still in the PRODUCTIVE state. Following powerup, it determines the new ARAMS state, sets the required variables, and reports the state change to the host.

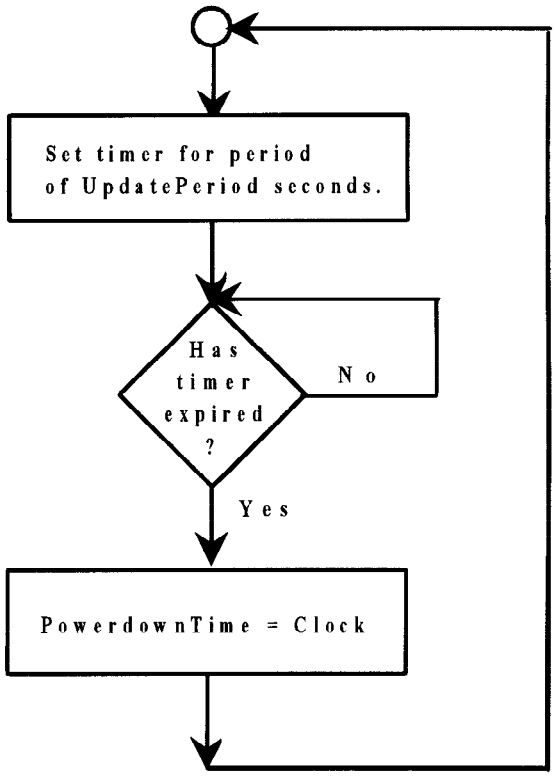

Figure R1-1
Periodic Update Logic

R1-2.3 The notation "=" indicates the value of a variable, while ":=" is used to indicate the act of setting a variable to a specific value. The notation "←" is used to indicate a message sent from equipment to host, and the notation "→" is used to indicate a message sent from the host to equipment.

Table 17 Table R1-1 Powerup Scenario

Time	*Action*	*Comment*
	ARAMSState = "2000" ARAMSTimestamp = t_{-1} PrdState = "1100"	Initial conditions prior to t = t_0: Equipment is in STANDBY PRODUCTION substate is "regular"
t_0	ARAMSTimestamp:= Clock PrevARAMSState:= ARAMSState ARAMSState:= PrdState	Requirements for PRODUCTIVE are satisfied. Equipment transitions automatically.
$t_1 = t_0 + 30$	PowerdownTime:= Clock	Periodic update of estimated powerdown time.
$t_2 = t_1 + 30$	PowerdownTime:= Clock	Periodic update of estimated powerdown time.
$t_3 = t_2 + 30$	PowerdownTime:= Clock	Periodic update of estimated powerdown time.
$t_4 = t_3 + 30$	PowerdownTime:= Clock	Periodic update of estimated powerdown time.
$t_5 = t_4 + 30$	PowerdownTime:= Clock	Periodic update of estimated powerdown time.
$t_6 = t_5 + 30$	PowerdownTime:= Clock	Periodic update of estimated powerdown time.
t_7		Powerdown
t_8	LastPowerdown:= PowerdownTime PowerdownTime:= Clock Elapsed time before powerdown = LastPowerdown – ARAMSTimestamp ($t_6 – t_0$) ARAMSTimestamp:= LastPowerdown (t_6) PrevARAMSState:= ARAMSState ("1100") ARAMSState:= "5000" ("UDT") PrdTime:= PrdTime + elapsed time ($t_6 – t_0$)	Powerup: Transition 1 Equipment system initialization ARAMS housekeeping Transition complete.
t_9	← ARAMS State Change Event Report "A" ← ARAMS Transition Event Report "B"	Equipment sends event notification to host.

R1-3 Equipment-Initiated Transition

R1-3.1 This scenario illustrates the events and activities that occur as the result of an equipment-initiated transition from PRODUCTIVE to UNSCHEDULED DOWNTIME, for equipment supporting the accumulators in Section 11.4.

Table 18 Table R1-2 Equipment-Initiated Transition

Comment	*Host*	*Equipment*	*Comment*
		ARAMSState = "1100" ("PRD/Regular Production")	Initial conditions.
			Equipment detects mechanical fault in wafer handler.
			Equipment suspends processing, transitions to UNSCHEDULED DOWNTIME, and notifies user.
		Elapsed time: = Clock – ARAMS ARAMSTimestamp:= Clock PrevARAMSState:= ARAMSState ("1100") ARAMSState:= "5000" ("UDT") DowntimeAlarm:= alarm/exception identifier DowntimeAlarmText := alarm/exception description DowntimeData:= additional information Prdtime:= Prdtime + elapsed time	

		← ARAMS State Change Event (Report “A”)	
		← ARAMS Transition Event (Report “B”)	

R1-4 Operator-Initiated Transition

R1-4.1 This scenario illustrates the events and activities that occur as the result of an operator-initiated transition from PRODUCTIVE to UNSCHEDULED DOWNTIME.

Table 19 Table R1-3 Operator-Initiated Transition

Comment	*Host*	*Equipment*	*Comment*
		ARAMSState = “1100” (“PRD/Regular Production”)	Initial conditions.
			Operator aborts process. Equipment halts processing.
		PrevARAMSState:=ARAMSState (“1100”) ARAMSState:= “5000” ARAMSTimestamp:= Clock SymptomID:= corresponding symptom identifier. SymptomText:= corresponding symptom description.	Operator selects new ARAMS state of UNSCHEDULED DOWNTIME (“5000”). Operator selects description of a symptom.
		Elapsed time:= Clock – ARAMSTimestamp ARAMSTimestamp:= Clock ARAMSState:= “5000” (“UDT”) DowntimeAlarm:= alarm/exception identifier DowntimeAlarmText:= alarm/exception description DowntimeData:= additional information PrdTime:= PrdTime + elapsed time	
		← ARAMS State Change Event (Report “A”)	
		← ARAMS Transition Event (Report “C”)	

R1-5 Host-Initiated Transition

R1-5.1 This scenario illustrates the events and activities that occur as the result of a host-initiated transition from its current substate of STANDBY to a different substate of STANDBY.

Table 20 Table R1-4 Host-Initiated Transition

Comment	*Host*	*Equipment*	*Comment*
		ARAMSState = “2800” (“SBY/No host”)	Initial conditions.
Host sends new ARAMS substate code.	ARAMSStateChange.req → (ARAMSCode = “2400”)		
		Elapsed time:= Clock Ð ARAMSTimestamp PrevARAMSState:= ARAMSState (“2800”) ARAMSState:= “2400” (“SBY”)	NOTE – ARAMSTimestamp is unchanged.
		← ARAMSStateChange.rsp	Equipment acknowledges the new state change.
		← ARAMS State Change Event (Report “A”) ← ARAMS Transition Event (Report “C”)	Equipment notifies host of ARAMS state change.

RELATED INFORMATION 2

NOTE: This related information is not an official part of SEMI E58 and is not intended to modify or supersede the official standard. Rather, these notes are auxiliary information provided as background or examples of possible application and are included as reference material. The standard should be referred to in all cases. SEMI makes no warranties or representations as to the suitability of the material set forth herein for any particular application. The determination of the suitability of the material is solely the responsibility of the user.

This section discusses issues that may arise through implementations of ARAMS.

R2-1 User-Initiated Transitions to UNSCHEDULED DOWNTIME

R2-1.1 Although substates of UNSCHEDULED DOWNTIME include (unplanned) change of consumables, out-of-spec input material, and facilities-related downtime, transitions to UNSCHEDULED DOWNTIME are generally regarded as resulting from an equipment fault. Operators have sometimes been known to log the equipment into UNSCHEDULED DOWNTIME as a way of freeing time, even though the equipment is operating completely within specifications.

R2-1.2 The equipment may use the status (read-only) ARAMSInfo to record information, such as Operator ID, that could be used in its defense.

R2-1.3 Additionally, equipment is encouraged to maintain an internal log of events. While the host may or may not request ARAMSInfo in an event report, the equipment's log could be used to record the information and would be accessible to field service personnel for analysis. Recording information such as the date, day of week, time of day, identifier of the operator (where known) could be an effective protection against abuse.

SEMI E58.1-0697 (Reapproved 0703)
SECS-II PROTOCOL FOR AUTOMATED RELIABILITY, AVAILABILITY, AND MAINTAINABILITY STANDARD (ARAMS): CONCEPTS, BEHAVIOR, AND SERVICES

This standard 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 April 11, 2003. Initially available at www.semi.org May 2003; to be published July 2003. Originally published June 1997.

1 Purpose

1.1 This document maps the services and data of the Automated Reliability, Availability, and Maintainability Standard (ARAMS) to SECS-II streams and functions and data definitions.

2 Scope

2.1 This document applies to all implementations of ARAMS that use the SECS-II message protocol (SEMI E5).

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

3 Referenced Standards

3.1 *SEMI Standards*

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

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

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

NOTICE: Unless otherwise indicated, all documents cited shall be the latest published versions.

4 Services Mapping

4.1 Table 1 shows the specific SECS-II streams and functions that shall be used for SECS-II implementations of the services defined in ARAMS.

4.2 The ARAMSStateChange service may be mapped either to S2,F41/F42 or to S2,F49/F50 or both. The equipment supplier is required to document the mapping used.

Table 1 Services Mapping Table

Message Name	*Stream, Function*	*SECS-II Message Name*
ARAMSStateChange	S2,F41/F42 S2,F49/F50	Host Command Send/Acknowledge Enhanced Remote Command/Acknowledge
ResetAccumulators	S2,F41/F42 S2,F49/F50	Host Command Send/Acknowledge Enhanced Remote Command/Acknowledge
TableSend	S13,F13/F14	Table Data Send/Acknowledge
TableRequest	S13,F15/F16	Table Data Request/Table Data

5 Service Parameter Mapping

5.1 Table 2 shows the mapping between service parameters defined by ARAMS and the data items defined by SEMI E5. An additional mapping for the ARAMSStateChange and ResetAccumulators services is shown in Table 4 below.

Table 2 Service Parameter Mapping Table

Parameter Name	*SECS-II Data Item*
AttrData	ATTRDATA
AttrName	ATTRID
ColHdr	COLHDR
ErrorCode	ERRCODE
ErrorText	ERRTEXT
ObjSpec	OBJSPEC
RequestStatus	HCACK
TableAck	TBLACK
TableCmd	TBLCMD
TableElem	TBLELT
TableID	TBLID
TableType	TBLTYP

5.2 Table 3 shows the data items in SECS-II messages that do not have a corresponding service parameter.

Table 3 Additional Data Item Requirement Table

Function	*SECS-II Data Item*
Used to satisfy SECS-II conventions for linking a multi block inquiry with subsequent multilock message.	DATAID
Used by S2F41 and S2F49 to indicate the ARAMSStateChange Service.	RCMD="ARAMSStateChange"
Used by S2F41 and S2F49 to indicate the ResetAccumulators Service.	RCMD="ResetAccumulators"

NOTE: The text strings specified in Table 3 for RCMD shall be recognized by the equipment, whether the equipment is or is not case-sensitive.

5.3 Table 4 provides the parameter mapping for the ARAMSStateChange and the S2,F41 and S2,F49 messages.

Table 4 Service Parameter Mapping (S2F41 and S2F49)

Parameter Name	*CPNAME*	*CPVAL/CEPVAL (Form)*
ARAMSCode	"ARAMSCode"	Text, 4 characters
ObjSpec	"ObjSpec"	Text. Conforms to data item OBJSPEC. NOTE: in S2,F49, ObjSpec maps directly to the data item OBJSPEC.
SymptomID	"SympID"	Unsigned integer
SymptomText	"SympText"	Text, 0–80 characters

NOTE: The text strings specified in Table 4 for CPNAME shall be recognized by the equipment, whether the equipment is or is not case-sensitive.

NOTICE: SEMI makes no warranties or representations as to the suitability of the standards set forth herein for any particular application. The determination of the suitability of the standard is solely the responsibility of the user. Users are cautioned to refer to manufacturer's instructions, product labels, product data sheets, and other relevant literature respecting any materials mentioned herein. These standards are subject to change without notice.

The user's attention is called to the possibility that compliance with this standard may require use of copyrighted material or of an invention covered by patent rights. By publication of this standard, SEMI takes no position respecting the validity of any patent rights or copyrights asserted in connection with any item mentioned in this standard. Users of this standard are expressly advised that determination of any such patent rights or copyrights, and the risk of infringement of such rights, are entirely their own responsibility.

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.

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.

- 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

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.

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.

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,

- 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.

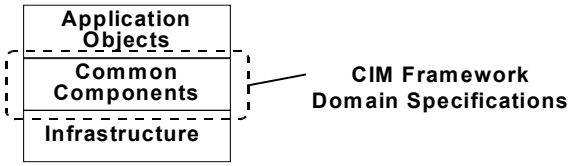

Figure 1
CIM Framework Layering

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

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.

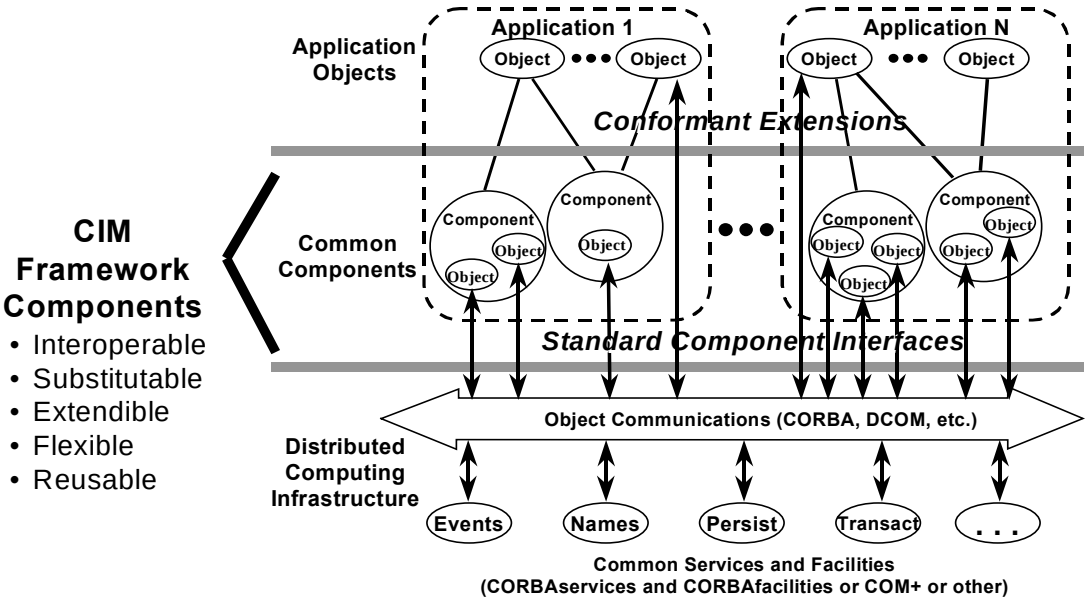

Figure 2
CIM Framework Component Architecture

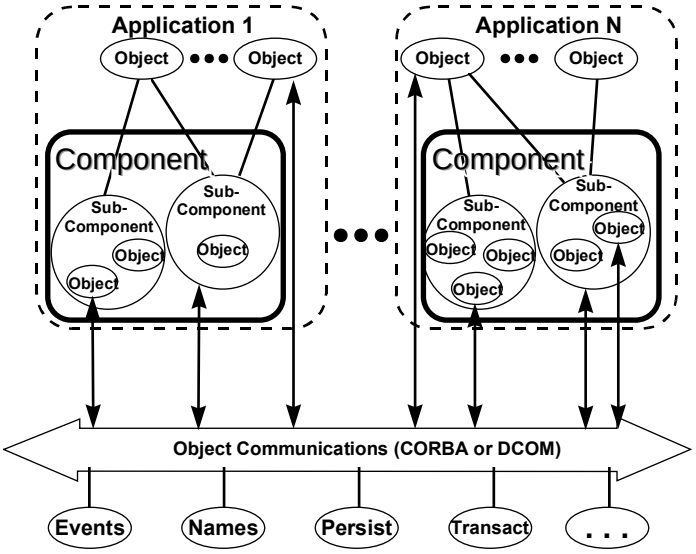

Figure 3
Components and Sub-Components

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.

- 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.

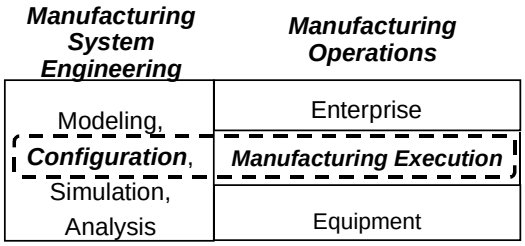

Figure 4
Boundaries of the CIM Framework: Manufacturing Execution

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.

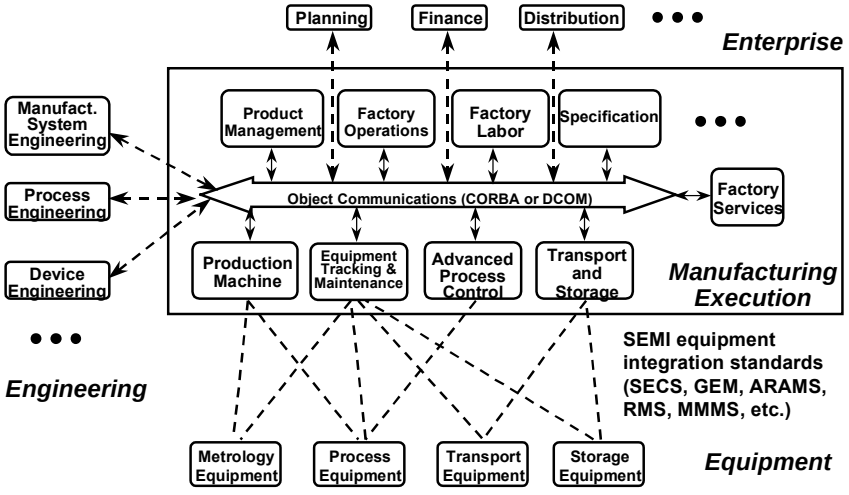

Figure 5
MES Component Integration

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

- 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

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

Product Management component is responsible for keeping its material location consistent with Material Transport and Storage material location information.

- Account for and track material loss.
- Support inventory regions: monitor, report and capture history of material movement through physical and logical regions of the factory.

7.2.10 *Responsibilities of Durables Management Component*

7.2.10.1 The Durables Management component manages the durable resources used in wafer manufacturing, including material containers (cassettes, SMIF pods, etc.), reticles, fixtures, test probes, etc. The component implements the generic resource tracking and maintenance architecture specialized for durables. The component also implements the common material architecture behavior so durables can be identified, grouped and located in the factory.

7.2.10.2 *Build-time Responsibilities of Durables Management Component*

- Define the specific durables in the factory (the inventory of carriers, reticles, etc.).
- Define inventory regions that track the use and movement of various durable types in a factory (for example, the reticles assigned to and existing in a lithography bay).
- Define relationships between durables and the process steps that use them.
- Define maintenance tasks for durables and what triggers them.

7.2.10.3 *Run-time Responsibilities of Durables Management Component*

- Track durable attributes (location, usage, contamination exposure, etc.) and state (available, not available, in use, etc.).
- Maintain relations between durables and other CIM Framework objects (such as product material contained in a wafer carrier, the reticles in a reticle carrier, the process step a reticle is used for, etc.).

  NOTE 5: The Material Transport and Storage component has some responsibility for managing material containers, such as to move and allocate empty pods, gas purge and clean pods, etc.

- Maintain, report and record history of durable attributes, state and relations.
- Support assignment of durables to factory jobs (e.g. supporting the scheduling of a reticle for use in a production machine job) and monitor and record job progress (from the perspective of the durable's role).
- Monitor durable maintenance triggers (such as excessive exposure to contaminants) and recommend durable maintenance jobs.
- Execute triggered durable maintenance jobs.
  - Change durable capabilities.
  - Execute durable maintenance jobs (pod cleaning, reticle inspection and repair, etc.) and report job progress.
  - Record durable maintenance triggers and job results in resource maintenance histories.
- Collect and report durable utilization and effectiveness measures.

7.2.11 *Responsibilities of Consumables Management Component*

7.2.11.1 The Consumables Management component manages consumable materials, including gases, chemicals, etc. The component implements the generic resource tracking and maintenance architecture to track consumable usage and quality (e.g., expired resist) and to enable maintenance based on replenishment levels and quality expiration. The component also implements the common material architecture behavior so consumables can be identified, grouped and located in the factory.

7.2.11.2 *Build-time Responsibilities of Consumables Management Component*

- Define the specific consumables in the factory (the types of gases, chemicals, etc.).
- Define inventory regions that track the use and movement of consumable types in a factory (for example, the equipment fed by a common gas bottle and feeders).
- Define relationships between consumable types and the process steps that use them.
- Define maintenance tasks and triggers for testing the quality of and replacing consumables.

7.2.11.3 *Run-time Responsibilities of Consumables Management Component*

- Track consumable attributes and state (supplier's lot number, quantity available, usage rate, expiration date, etc.).
- Maintain relations between consumables and other CIM Framework objects (the product groups a particular consumable was used on, the equipment fed by a common gas bottle, etc.).

- Maintain, report and record history of consumable attributes, state and relations.
- Support assignment of consumables to factory jobs (e.g., the scheduling of a gas feed pressure for use in a production machine job) and monitor and record job progress (from the perspective of the consumable's role).
- Monitor consumable maintenance triggers (quality checks and replenishment levels) and recommend consumable maintenance jobs.
- Execute triggered consumable maintenance jobs.
- Change consumable capabilities.
- Execute consumable maintenance jobs and report job progress.
- Record consumable maintenance triggers and job results in resource maintenance histories.
- Collect and report consumable utilization and effectiveness measures.

7.2.12 *Responsibilities of Specification Component*

7.2.12.1 The Specification component supports definition and use of process specifications, product specifications and bill of materials. Process specifications define process flows (the process steps to manufacture a product, along with supporting material movement, advance process control, equipment maintenance and other tasks). Product specifications relate product types to the process flows to build them.

7.2.12.2 The CIM Framework does not specify a fully-detailed representation of process flows. At run-time, it views a specification as a current operation (the Process Flow Context) and a view of potential next steps (the Process Flow Iterator). Although a complete MES solution must build and provide process flow details, the CIM Framework standard does not require this detail for component integration.

7.2.12.3 Recipes, process capabilities, maintenance specifications, document management and version control services are the responsibilities of other components. The Specification component references and uses these services.

7.2.12.4 *Build-time Responsibilities of Specification Component*

- Build process operation specifications.
  - Specify a processing step as a relationship between a combination of recipes, data collection plans, advanced process control strategies, process capabilities, processing constraints (timing dependencies, contamination constraints, qualified machines, etc.), operator qualifications, durables, consumables and other elements.

    NOTE 6: At run-time, a process operation specification is fully bound to create a process operation (a specific machine is selected, recipe parameters are set, specific support resources are named, etc.).
- Build manufacturing process flows.
  - Specify a process flow as a combination of material processing steps, material movement tasks, equipment qualification and maintenance tasks and other tasks.

    NOTE 7: Process flows are primarily a combination process operation specifications, but the process flow should allow inclusion of explicit steps for material movement, metrology, equipment calibration, etc.
  - Define process flow traversal logic (rework decisions, metrology sampling decisions, alternate path decisions, parallel assembly routes, etc.).
    - preconditions for beginning a step.
    - postconditions for completing a step.
    - logic to enable selecting appropriate next steps.
  - Support recursive (nested) composition of simple process flows (partial flows or mini-flows) into complex process flows.
- Support a library of configurable, reusable process operation specifications and process flows.
- Build product specifications.
  - The process flow to build the product.
  - Product-specific run-time parameters.
    - Reticle and other required resources.
    - Quality/metrology sampling plans.
    - Required manufacturing technologies.

      NOTE 8: This is not a full product specification. It is only those elements needed to select and initialize the manufacturing process.
- Define and maintain relationships between process flow specifications and product specifications which use them.
- Use document and version management for specification change control.

7.2.12.5 *Run-time Responsibilities of Specification Component*

- Create a fully-bound process operation from a process operation specification.
- Create a Process Flow Context (PFC) from a product specification (create the initial process flow for a product).

  NOTE 9: The PFC is the "pointer" marking the current step in a process flow.

- Transition the PFC from the current process operation to the next process operation in the flow (move along the process flow).

  NOTE 10: The decision to move is the responsibility of other components. The Specification component records the decision.

- Create a PFC from an existing PFC (for example, to support splitting of a product group).
- Create a Process Flow Iterator (PFI) from a product specification (create an initial process flow iterator for a product).

  NOTE 11: The PFI is a process flow navigation aid to look ahead or look behind at steps in a process flow without actually incrementing the PFC. This allows scheduling and other functions to determine what steps are possible and to use processing history in determining what steps to take next.

- Create a PFI from an existing PFI (for example, to support evaluating among alternatives).
- Use the PFI to traverse forward and backward through a process flow.
  - Using the results of executing process flow traversal logic (the factory job control or process job control does the actual execution of traversal logic to decide among alternate paths).
- Provide query support for the contents and relationships of process operation specifications and process operations.
- Maintain relationships of active process flows to the factory jobs, production machine jobs, and transport jobs that are implementing the process flow.
- Maintain relationships of active process flows (process flow contexts) to product groups (e.g. process groups, transport groups, etc.) and product group histories for recording the results of executing process flow steps.
- Maintain relationships of process flow specifications to process capabilities and process resources.
- Support effectivity changes to process flows and process operation specifications (if the process flow version changes for the current product group on the flow, be responsive to those changes).

7.2.13 *Responsibilities of Factory Operations Component*

7.2.13.1 The Factory Operations component provides support for maximizing factory effectiveness through efficient use of factory resources to satisfy product demand and planned objectives. Examples of overall factory effectiveness measures are on-time delivery, profit margin, profit rate, cost per wafer, yield, overall equipment effectiveness, cycle time, etc. The Factory Operations component also provides the interface to the enterprise planning systems to commit the factory to fulfill product requests and to communicate factory capacity and production capability to enterprise planning. The product requests also provide a placeholder for order information (such as quantity, due date, priority, etc.) and are part of the link to material tracking at the enterprise level. Given the factory-wide scope of Factory Operations, this component owns the overall factory resource model of machines, areas, inventory regions, etc.

7.2.13.2 Factory Operations may not have exclusive responsibility for maximizing all factory objectives. The Production Machine component maximizes the use of process resources (chambers, internal storage, production batching, etc.), the Material Transport and Storage component maximizes the use of transport and storage resources (transport vehicles, transport paths, storage locations, etc.), the Equipment Tracking and Maintenance component maximizes equipment availability and capability, etc. The business logic and decision-making methods may be distributed to improve overall enterprise effectiveness (Enterprise functions), overall factory effectiveness (Factory Operations functions) and overall equipment effectiveness (Production Machine, Material Transport and Storage and other functions). The CIM Framework provides flexibility, enabling the spectrum from centralized, enterprise-wide or factory-wide operations to distributed, resource-focused operations with coordination to achieve factory-wide effectiveness. The CIM Framework Job Architecture is critical to enabling this flexibility (see Section 7.3.4).

7.2.13.3 *Build-time Responsibilities of Factory Operations Component*

- Define the factory model of resources (machines, people, durables, etc.), areas (bays, lines, cells,

etc.), and inventory regions (see Section 7.3.3 for a description of the factory model of resources).

- Define the (initial or nominal) capabilities of the factory and area resources, including factory capacity, area capacity and process capability (detailed capabilities of the machines, people, etc. are the responsibility of other components).

7.2.13.4 *Run-time Responsibilities of Factory Operations Component*

- Dynamically receive product requests (from the enterprise or internally) and map them to product work-in-progress and planned material release (lot starts).
  - Associate material to a process flow (in the Specification component) for the product.
  - Consolidate product demand in the factory.
  - Request Factory Jobs or modify existing Factory Jobs to drive material through its process flows, in response to product requests.
    - New Factory Jobs for new material release,
    - Modify existing Factory Jobs (and associated material model and assigned process flow) for assigning new or modified product requests to material in process.
  - Set or modify product material quantity and due date(s) and other scheduling support information (such as priority and planned profit margin).
- Execute Factory Jobs to drive material through its process flow, using scheduling and dispatching services of the Scheduling component and requesting work from Production Machine, Material Transport and Storage, Equipment Tracking and Maintenance, and Factory Labor components.
  - Based on factory state (material process state, material due date and priority, material location, equipment state, equipment process capability, operator availability and qualification, etc.) and where the material is in its process flow, request and track Production machine jobs, transport jobs, maintenance jobs and other jobs that implement and enable the process steps in the process flow.
  - Based on success, failure or partial failure of production machine jobs and transport jobs, and based on metrology results and other information, determine the next activities for material and factory resources. This may include splitting or joining product groups, scrapping material, downgrading or changing the product type of material, and other actions responsive to job performance and resource availability. Once split, scrap and other decisions are made, Factory Operations may use scheduling utilities to select the next activities for the material and resources.
- Record and report factory job status and decisions in factory job histories.
- Report status and progress on enterprise product requests.
- Manage assignment of human resources to other factory resources.

7.2.14 *Responsibilities of Scheduling Component*

7.2.14.1 The Scheduling component supports Factory Operations, Material Movement, Production Machine and other components by ordering, in time, jobs that process material on equipment, move material, maintain equipment, etc. The scheduler uses knowledge of product demand, equipment and material state, process flows, throughput bottlenecks, operational policy and constraints, and other information to recommend jobs that maximize effective utilization of factory resources to satisfy product demand and planned objectives.

7.2.14.2 The CIM Framework specifies a dispatching function to support Factory Operations. Dispatching provides an answer to questions of the form, “What is next for this material or resource?” The answer may be based on any number of current or future constraints and objectives, but the dispatcher does not typically define a sequence of jobs projected into the future. Scheduling functions in the Scheduling component provide this future job sequencing function. Separate scheduler and dispatcher utilities can also provide resource-focused utilities, such as scheduling of process resources within a production machine or scheduling of material transport and storage resources and internal material movements within the factory-wide material transport and storage system.

7.2.14.3 *Build-time Responsibilities of Scheduling Component*

- Configure scheduling and dispatching decision mechanisms (for example, define scheduling goals and constraints, rules for resolving conflicting constraints, etc.).

7.2.14.4 *Run-time Responsibilities of Scheduling Component*

- Monitor resource and material state and apply scheduling and dispatching decision mechanisms to

recommend the next task (dispatching) or a sequence of tasks (scheduling) for factory resources.

7.2.15 *Responsibilities of Equipment Tracking and Maintenance Component*

7.2.15.1 The Equipment Tracking and Maintenance component implements the Factory Resource architecture (see Section 7.3.3) specialized for production equipment (process and metrology tools and material handling and storage equipment).

7.2.15.2 Note that the machine is the software representation of physical equipment and should reflect the state of that equipment. The functionality can be implemented within the equipment, within the MES, or in a separate system such as a cell controller, station controller or equipment integration solution.

7.2.15.3 *Build-time Responsibilities of Equipment Tracking and Maintenance Component*

- Configure production machines as a combination of, or relationship to, process resources (such as process chambers) and support resources (loadports, add-on sensors, internal material buffers and transport, etc.).
- Configure material handling resources as a combination of transport machines and storage machines and relations to support resources (loadports, add-on sensors such as automatic material identification, etc.).
- Configure equipment state models compatible with SEMI E10 (at the machine level and/or at the machine resource level).
- Configure (initial or nominal) production machine capabilities.
- Configure the (initial) recipe namespaces that production machines use and manage.
- Configure the (initial) material locations that material transport equipment is able to reach and the locations it is authorized to reach.
- Define equipment maintenance tasks and what triggers them.
  - Tasks: repair, calibration, qualification, etc.
  - Triggers: failures, usage, elapsed time, need by another related maintenance task, etc.
- Use document management and version control for maintenance specification change control.

7.2.15.4 *Run-time Responsibilities of Equipment Tracking and Maintenance Component*

- Track the equipment state and update the machine state, accordingly.
  - Keep the machine state model current.
  - Report and record equipment and machine state changes, alarms and events in machine history.
  - Support query of machine state and history.
  - Update machine and machine resource process capabilities to reflect machine state and capability and machine assigned capabilities.
- Drive the equipment to a requested state.
- Collect and report machine performance according to SEMI E58 and other utilization and effectiveness measures.
- Monitor equipment maintenance triggers.
- Execute and monitor triggered maintenance jobs.
  - Update equipment state and resource capability (before and after maintenance).
  - Recommend maintenance jobs for scheduling.
  - Execute maintenance jobs and report job progress.
  - Record maintenance triggered and maintenance job results in equipment maintenance history.
- Manage the relationships between production machines that have specific process capabilities and process flows that need specific process capabilities.
- Manage the material locations that material transport equipment is capable of reaching and is authorized to reach.

7.2.16 *Responsibilities of Production Machine Component*

7.2.16.1 The Production Machine component supports execution of production machine jobs on process equipment and interaction with material handling equipment for material input/output at the machine. The functionality can be implemented within the equipment and/or in a separate system such as a cell controller, station controller or equipment integration solution.

7.2.16.2 *Build-time Responsibilities of Production Machine Component*

7.2.16.2.1 Note that the configuration of the machine is the responsibility of the Equipment Tracking and Maintenance component.

7.2.16.3 *Run-time Responsibilities of Production Machine Component*

- Initiate and execute production machine jobs to perform specific process operations (and recipes) on specific material (process groups). Here are the steps to initiating and executing production machine jobs:

  1) Validate the production machine job: compatibility and availability of the material in the process group, process capability of the equipment, proper machine state, valid and available recipe or process specification, necessary operator skills, etc.

  2) Equipment setup: recipe download and select, launch advanced process control (APC) calculation, make operator and APC-recommended adjustments, configure equipment and sensors according to data collection plans.

  3) Initiate processing.

  4) Monitor process operation:

  - Collect process run data from equipment, add-on sensors and operator entry according to data collection plans.
  - per machine and per process resource.
  - Store in process run histories.
  - Event management: capture, interpret, record and respond to equipment events, fault detection events and other process anomalies.
  - Monitor equipment operation and state (from the equipment tracking and maintenance component) as another source of information on process operation.
  - Generate MES-level (machine) events and alarms.
  - Track material within machine and record with process run histories and product histories.

  5) Complete processing.

  - Execute post-processing tasks for the process operation.
  - Generate job completed events and invoke "inform job completed" method on job requestor.

- Sequence steps between machines and machine resources (for that sequence logic that is not covered by embedded equipment control).
- Implement recipe control and recipe agent functionality as defined in SEMI E42 - Recipe Management Standard. Integrate with factory-level recipe namespace functionality in the CIM Framework Recipe Management component.
- Job accountability: account for and report material yield loss in material management component and equipment productivity loss in equipment tracking component.
- Interact with Material Transport and Storage component and loadport for material hand-off handshake protocols.
- Manage and monitor equipment when idle.

7.2.17 *Responsibilities of Recipe Management Component*

7.2.17.1 The Recipe Management component provides a capability for applying machine recipes across the factory. Recipes for multiple machines of the same process capability using the same recipe syntax are managed by specializing the recipes based on specific machine settings. The Production Machine component is responsible for merging the recipe with machine-specific data and run-time data (such as Advanced Process Control component recommendations for recipe settings) into an executable recipe and for downloading it to the equipment or selecting and modifying it in the equipment. The Recipe Management component also works with Recipe Executors and Recipe Agents (terminology from SEMI E42) implemented in the Production Machine component, in the equipment, or elsewhere.

7.2.17.2 *Build-time Responsibilities of Recipe Management Component*

- Build process recipes.
  - Generic.
  - Specific to a machine type (same process capability and recipe syntax).
  - Including identification of variable parameters in recipe.
  - Use document management and version control for change management and effectivity.
- Define, configure and initialize recipe namespaces.

  NOTE 12: Including this responsibility under "build time" does not imply that all recipes are built at MES system build time. Recipes may be built after the MES system is running.

7.2.17.3 *Run-time Responsibilities of Recipe Management Component*

- Work with Production Machine component and other recipe controllers to select, download and modify recipes.
- Implement resource effectivity control (manage when and to what material and machines to apply recipe changes).

7.2.18 *Responsibilities of Advanced Process Control Component*

7.2.18.1 The Advanced Process Control component (defined in SEMI E93) supports the Production Machine component execution of production machine jobs by optimizing machine-specific settings for the current material and machine state to achieve desired process effects. It also detects faults in processing and recommends an appropriate response to the Production Machine component. Advanced Process Control strategies can include statistical process control, model-based process control, multi-variate analysis, trace analysis, fault pattern matching, or any other appropriate mechanism. The internal design of the Advanced Process Control component allows integrating different types of analysis and computation mechanisms and their execution environments into a single control strategy.

7.2.18.2 *Build-time Responsibilities of Advanced Process Control Component*

- Define control strategies and associated sensor processing algorithms, fault detection algorithms and optimal process control algorithms.
- Define real-time control data structures to support control algorithms.
- Define data collection plans to acquire control data.
- Define control strategy selection logic.

7.2.18.3 *Run-time Responsibilities of Advanced Process Control Component*

- Select a control strategy and associated sensor processing and analysis algorithms to optimize processing and detect faults.
- Launch, coordinate and monitor the execution of data collection plans and control processing and analysis algorithms.
- Apply algorithm results to recommend process settings and detected fault response actions (recipe adjustment, equipment shutdown, maintenance, rework, etc.).
- Update control data with algorithm results.
- Update control data with equipment and product material data (current equipment state, material state, process operation, etc.).
- Record and report control strategy and algorithm results and recommendations in process run histories.

7.2.19 *Responsibilities of Material Transport and Storage Component*

7.2.19.1 The Material Transport and Storage component (defined in SEMI E102) supports active and manual material movement and active and passive material storage (active storage: stockers, smart racks, equipment buffers; passive storage: tables, racks). It executes Transport Jobs and interacts with the Production Machine components for material handoff to and from process equipment. The component tracks and reports on material location and material movement history. The equipment configuration, tracking and maintenance functions are the responsibility of the Equipment Tracking and Maintenance component. For complicated interbay and intrabay material transport and storage, there may be multiple instances of the component -- one for each material movement area -- that cooperate to provide factory-wide material movement functions.

7.2.19.2 *Build-time Responsibilities of Material Transport and Storage Component*

7.2.19.2.1 Note that configuration of the material transport and storage equipment and its capabilities is the responsibility of the Equipment Tracking and Maintenance component.

7.2.19.3 *Run-time Responsibilities of Material Transport and Storage Component*

- Support scheduling of material movement and processing by predicting time for material delivery to specific locations.
- Execute and monitor transport jobs to move material groups (transport groups) to specific locations.
    - Validate that the job can be done: the material is available, the destination is reachable and has available storage or loadport capacity, there are sufficient material movement resources (cars, storage space, etc.) to implement the job.
    - Implement the job by scheduling and enacting actions on internal material movement and storage resources (cars, turntables, stocker shelves, etc.).
    - Collect and record data on transport job execution and history.

- Capture, record, interpret and respond to equipment events and fault detection.
- Generate MES-level job status events and material location change events.

- Record and report on material locations and material movement histories for material in the system.
- Interact with Production Machine component and loadport for material hand-off handshake protocols.

7.2.20 *Responsibilities of Factory Labor Component*

7.2.20.1 The Factory Labor component (defined in SEMI E86) implements the generic Factory Resource architecture (see Section 7.3.3) applied to factory personnel resources. It represents personnel, their capabilities (skills) and skill maintenance (training), and their assignment to other factory resources (to machines and areas) and jobs. The Factory Labor component integrates with access security services (in the Factory Services group) to associate personnel with roles, identity authentication (e.g. system login passwords) and access authorization.

7.2.20.2 *Build-time Responsibilities of Factory Labor Component*

- Configure the Person resource model.
  - Each Person's name, identification, department, role, etc.
  - Each Person's initial skills and authorization to perform jobs and access data.
  - Each Person's initial assignment to other factory resources (such as specific machines or factory areas).
  - Each Person's medical qualifications.
- Define skill maintenance tasks and what triggers them.

7.2.20.3 *Run-time Responsibilities of Factory Labor Component*

- Track and report the Person skills, authorization and assignments and record these in Person histories.
- Support assignment of Persons to factory jobs and monitor and record job progress (from the perspective of the Person's role).
- Monitor skill maintenance triggers (such as expired skill certification) and recommend skill maintenance jobs.
- Execute triggered skill maintenance jobs.
  - Change Person capabilities (before and after training).
  - Execute and monitor skill maintenance jobs (training, certification testing) and report job progress.
  - Record skill maintenance triggers and job results in person skill maintenance histories.
- Collect and report person and skill performance (utilization and effectiveness).
- Maintain relationships between persons and system security for identity authentication and authorization.

7.3 *Common Architectural Patterns*

7.3.1 This section documents architecture patterns that serve to functionally integrate CIM Framework components. The material architecture defines functionality common to Product Management, Durables Management and Consumables Management components. The factory resource architecture defines relationships and common functionality of factory resources. The job architecture defines a factory-wide model of controlling factory operations.

7.3.2 *Material Architecture*

7.3.2.1 The Product Management, Durables Management and Consumables Management components all share a common architecture for identifying, grouping and locating materials. Figure 6 illustrates the common material interface architecture.

7.3.2.2 Material has an identification, history, location, inventory region and associations to material containers that may contain it. Material can also be in multiple material groups that physically (same location or carrier) or logically (same lot, same process batch, same product family, etc.) associate material. Specializations of material include products, durables and consumables. Specializations of material groups include product groups, lots, process groups and transport groups.

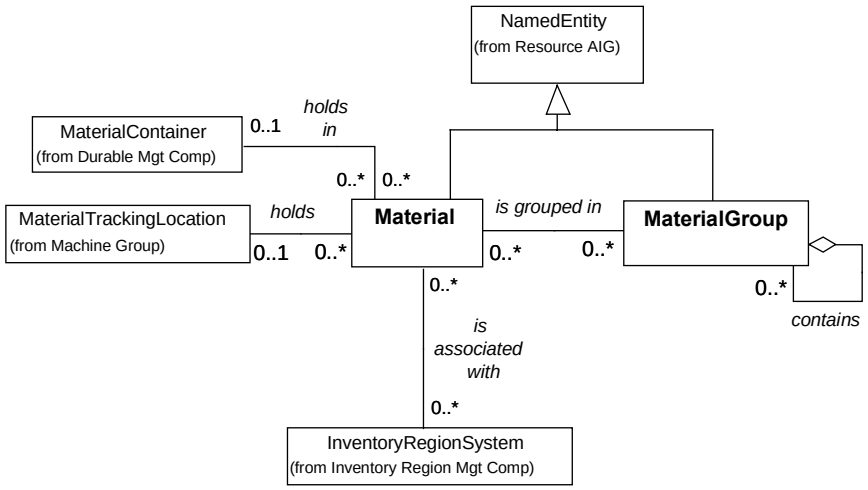

Figure 6
Material Interface Architecture

7.3.3 *Factory Resource Architecture*

7.3.3.1 Factory resources, such as machines, people and reticles, are the entities that participate in transforming material into products. The CIM Framework defines an overall domain architecture for factory resources that supports

- resource tracking and maintenance (resource usage counters, maintenance procedures, etc.),
- hierarchical organization (e.g., a factory is organized into areas which group machines and people),
- composition of resources (e.g., a machine is composed of chambers, load ports and sensors), and
- an object-oriented type structure (transport machines and production machines are types of machines; machines, ports, durables and people are support resources, etc.).

7.3.3.2 This section defines the CIM Framework resource architecture.

7.3.3.3 *Factory Resource Hierarchy*

7.3.3.3.1 A factory is usually viewed as a hierarchy of resources organized in groups, as in Figure 7. Each level or portion of the hierarchy has an associated task scope.

- An enterprise often includes multiple factories. The task of the enterprise is to deliver products to customers.
- A factory is made up of areas and bays with interbay material handling systems. The task of the factory is to manufacture product in response to product requests.
- Areas or bays are made up of processing and metrology equipment with intrabay material handling systems. The task of the areas or bays is to perform a group of process operations on groups of materials, move material to the processing equipment, and perform maintenance and other supporting tasks. For a simple factory, the area level is optional. For a complex factory, there may be multiple area levels, with areas made up of sub-areas.
- Complex processing equipment (such as a cluster tool) is made up of processing chambers and internal material handling and storage resources. The task of the processing equipment is to perform a process or metrology step or a closely related group of steps on a material lot or process group. The task of the processing resource in the processing equipment (such as a chamber in a cluster tool) is to perform a single process step on a single wafer or on a collection of wafers in a process resource group.
- Material handling and transport systems are made up of interconnected stockers, tracks, conveyers, robotic handlers and other material storage and movement equipment. The task of the material handling systems is to store and move materials in transport groups.

- Manufacturing operators and other factory personnel are important resources to facilitate and supervise material processing and handling, preventive maintenance, and in some cases, perform manufacturing steps such as moving material or performing a visual inspection.
- Other factory resources include durables (reticles, cassettes, pods, etc.) and consumables (gases, utilities, etc.).
- The information model in Figure 8 shows the CIM Framework Resource Composition Architecture. The interfaces in the box labeled Resource Abstract Interface Group define the generic resource architecture. In general, resources can be composed of other resources. In particular, at the factory and area levels, Factory can be composed of Areas and an Area can be composed of other Areas. Areas group Support Resources, such as machines and persons. This is not a strict composition so that support resources can be in more than one Area grouping. At the machine level, Machine Resources are composed of other Machine Resources, in general. In particular, Production Machines are composed of Process Resources.

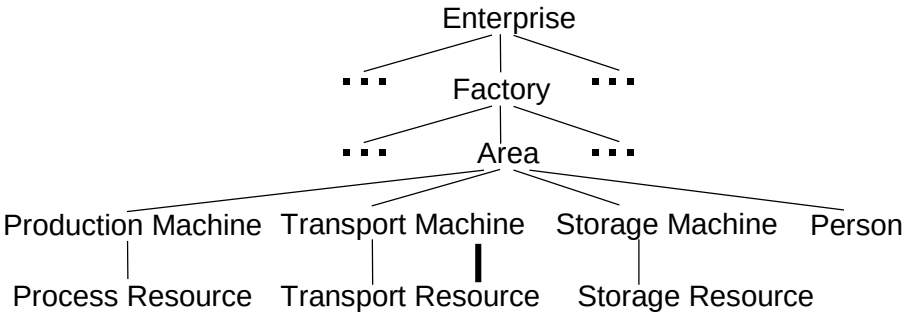

Figure 7
Factory Resource Hierarchy Model

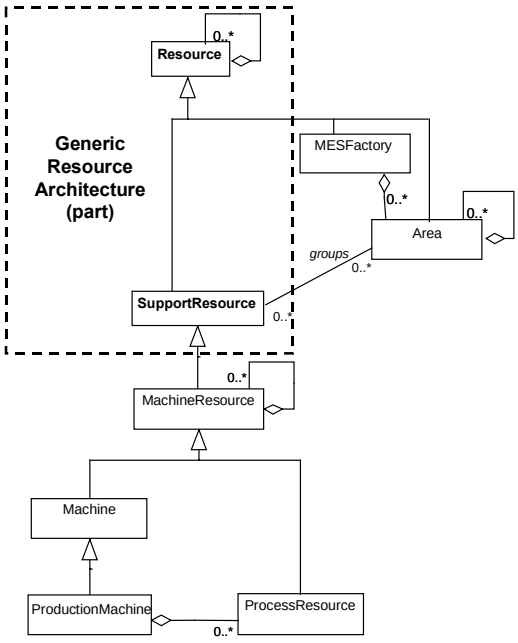

Figure 8
Resource Composition Architecture

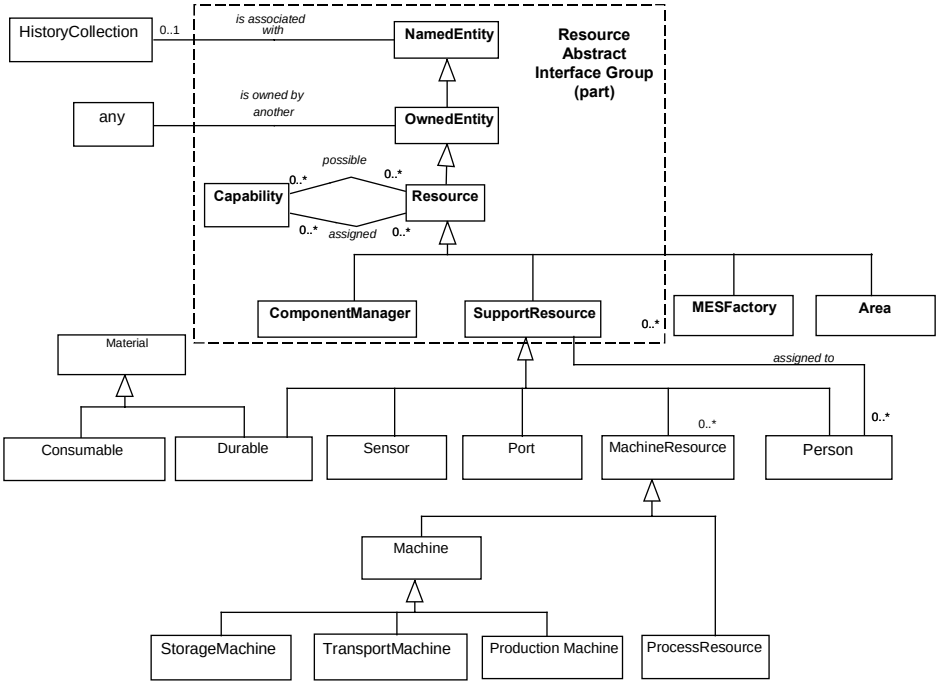

Figure 9
Resource Type Architecture

7.3.3.3.2 Figure 9 shows the CIM Framework Resource Type Architecture. Specializations of these interfaces, such as Machine Resource and Durable, provide component-specific details that extend the generic resource architecture. The functions added at each level of type specialization are discussed next.

7.3.3.3.3 *Named Entity*

7.3.3.3.3.1 The CIM Framework Named Entity interface provides for names and histories to be associated with any CIM Framework entity, including resources.

7.3.3.3.4 *Owned Entity*

7.3.3.3.4.1 The Owned Entity interface provides for resources to be organized into hierarchies.

7.3.3.3.5 *Resource*

7.3.3.3.5.1 The Resource interface provides for

- composition of subresources into complex resources,
- representing resource capabilities (such as capacity or process capability) (See Section 7.3.3),
- a simple state model (available, not available) and services to startup and shutdown the resource, and
- modeling of resource levels as in Figure 7 and Figure 8.

7.3.3.3.5.2 Each Resource has an attribute called resourceLevel to support the factory resource composition hierarchy of Figure 8. The CIM Framework defines the following resource levels: Factory (called MESFactory so as not to be confused with the OMA's object Factory type), Area, Machine and Machine Resource. The Area level is optional, or it may have multiple levels (such as an area called LithoBay with sub-areas called Zone1 and Zone2). Other levels can be defined. The resourceLevel concept, combined with the Named Entity and Owned Entity, also provide a name scoping facility.

7.3.3.3.6 *Support Resource*

7.3.3.3.6.1 The Support Resource interface provides services for reserving resources (to support scheduling, for example) and to track resource utilization and effectiveness and associate maintenance tasks with resources (see Figure 10 and Section 7.3.3.5). The support resources include machine resources, material load ports, sensors, durables and people.

7.3.3.3.7 *Machine Resource*

7.3.3.3.7.1 The Machine Resource interface provides attributes for machine description, serial number, model number, vendor, software version, etc. The interface

extends resource composition to provide machine resources as composites of machine (sub)resources and specialized process resources with associated process capabilities. The Machine Resource interface also extends the resource state model to include the states and substates of SEMI E10 and SEMI E58. Specific subtypes of Machine Resource and Machine are possible. The CIM Framework defines the subtypes for Production Machine and Process Resource. Additional subtypes to support Transport Machine and Storage Machine interfaces are also possible if there are additional attributes or behaviors required that Machine and Machine Resource interfaces do not already provide.

7.3.3.3.8 *Port, Sensor, Durable, Consumable, Person, etc.*

7.3.3.3.8.1 The other support resource subtypes provide additional, resource-specific, attributes and behavior. Note that durables are also subtypes of Material so they can be moved and tracked as material. Consumables are not currently considered as resources, but a given implementation of the CIM Framework could have consumables be support resources to track their consumption and replenish them as regular maintenance. Persons are considered support resources to schedule them and associate them with machines and other resources and to have associated "maintenance" schedules for training and certification.

7.3.3.4 *Resource Capabilities*

7.3.3.4.1 Resources have associated capabilities, as Figure 9 shows. Factory operations and scheduling use capabilities to identify resources with the capability to perform a specific task. A CIM Framework capability is represented as a text string, providing flexibility to define types of capabilities appropriate for a variety of resources. For example, an MESFactory resource may have capabilities defined in terms of its capacity and the product families it can produce, and a Process Resource may have capabilities that define its C_{pk} process capability and its throughput.

7.3.3.4.2 The Capability interface maintains a list of Resources which have that specific capability, and the associated Resource interface maintains a list of its overall possible capabilities and the subset of possible capabilities that resource is assigned to perform.

7.3.3.4.3 The Capability interface is specialized for Process Capabilities associated with Process Resources. This provides an association to the Process Operation Specifications a Process Resource can perform and the Process Durables and the person Skills needed to support the Process Capability. Other Resources can specialize the Capability interface.

7.3.3.5 *Resource Tracking and Maintenance*

7.3.3.5.1 Support Resources have an associated resource tracking and maintenance management function, as Figure 10 shows. A Resource Tracking Supervisor for a Resource monitors and records resource usage and status and creates Maintenance Jobs according to Maintenance Specifications. Factory Operations, with the aid of a scheduler or dispatcher, assigns Persons and other support resources defined in the Maintenance Specification and initiates Maintenance Job execution. The Resource Tracking Supervisor then supervises Maintenance Job execution. The Maintenance Job and Resource Tracking Supervisor specialize the Job/Job Supervisor pair as Section 7.3.4.6 describes. The Resource Tracking Supervisor also records appropriate history, such as maintenance logs and resource utilization and state, in the Named History Collection for the Resource (inherited from Named Entity). Resource type-specific specializations of the resource tracking and maintenance interfaces are possible.

7.3.4 *Job Architecture*

7.3.4.1 This section describes the job management and control architecture of the CIM Framework. It defines a hierarchical job supervision and control structure that manages factory-level jobs (create product material to fill enterprise product requests) and implements these factory-level jobs by creating and managing machine-level process jobs and transport jobs, coordinated with maintenance jobs and other support jobs. The job control architecture defines and manages relationships between manufacturing resources (process equipment, transport equipment, people), material groups (lots, product groups, process groups, transport groups) and specifications (product specifications, process flows and recipes). This is the central functional architecture which integrates the CIM Framework components into a coherent manufacturing execution system architecture. The concepts are derived from existing standards and related standards efforts [ANSI], [ALBUS], [OMA], [WfMC].

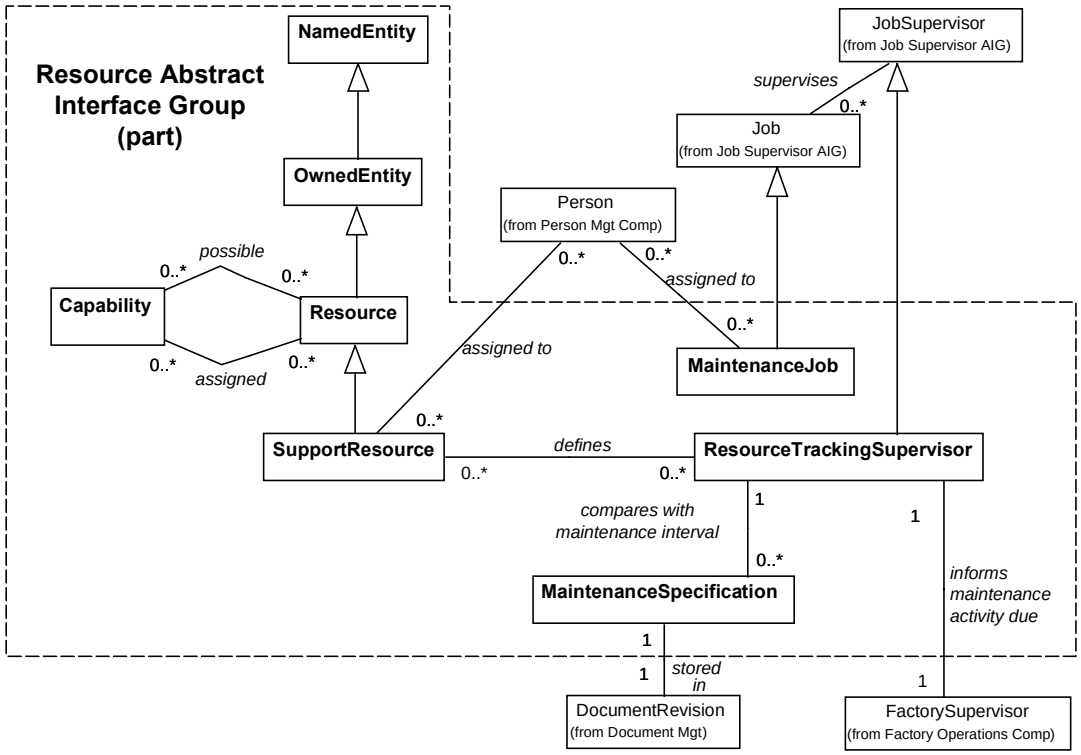

Figure 10
Resource Tracking and Maintenance Architecture

7.3.4.2 *Job Architecture Concepts*

7.3.4.2.1 A job represents a unit of work requested of and performed (or facilitated) by a factory entity that results in some change to the overall factory state. There are several important aspects of a job within the CIM Framework:

- A job typically takes a non-zero time to perform and has a non-zero chance of refusal or failure.
- A job may encapsulate a decomposition into a combination of jobs/tasks/activities.
- There is a notion of higher-level jobs and lower-level jobs. Coordination of lower-level jobs are delegated to other job supervisors to ensure that the higher-level job is completed.
- There is a job requestor.

7.3.4.2.2 The CIM Framework specifies a number of structures for job control. The complexity and variability of the factory requires some organizational structure and separation of manufacturing tasks. Breaking up a complex task into a coordinated interoperation of simpler tasks enables practical solutions to complex problems (the principle of "divide and conquer"). This results in a key organizational structure based on the factory resource hierarchy of Figure 7 and Figure 8 , with the separation of tasks summarized in Section 7.3.3.3. The job control architecture defines how tasks are assigned and coordinated across the hierarchy of factory resources.

7.3.4.2.3 Another key job structure is the relationship between manufacturing tasks and the material and resources used to carry out the task. For example, Factory Operations is responsible for efficiently allocating machine resources with the required processing capability to the material work-in-progress to drive the material through its process flow. A job is a combination of a requested task and the material and resources needed to execute that task (see Figure 13). The relationship between task, material and resource, combined with a hierarchical job structure (based on a hierarchical resource structure) results in complex relationships between tasks, material, and resources at multiple levels.

7.3.4.2.4 Given the complexity, scope and variability (chance of failure or partial success) of jobs, the CIM Framework separates job control into explicit functions.

It does not bury job control into material management functions for driving process flows, nor does it bury job control into machine management functions for driving process operations. The CIM Framework makes job control explicit, providing an architectural structure to attach decision support logic (such as scheduling utilities), business processes (workflow) and business rules that enforce operational policy. Further, the CIM Framework distributes and coordinates job control among factory jobs, production machine jobs, transport jobs, and maintenance jobs. This allows job control to manage "local" complexity while coordinating factory-wide operations toward "global" objectives.

7.3.4.3 *Hierarchical Task Structure*

7.3.4.3.1 At the lowest level of the factory hierarchy (the resource level in Figure 7), the tasks are single process or metrology operations or material movements. Through a complex, context-dependent combination of single tasks, products are manufactured and delivered to customers. This complex combination of single tasks is a task structure as shown in Figure 11. In manufacturing operations, these structures are pre-defined as task procedures, work flows, process specifications, etc. To accommodate manufacturing variability and exceptions, though, the structures must also be adjusted and modified as they are executed. For example, as factory operations selects specific machine resources to perform process steps, it may insert machine-dependent setup tasks and operation sequences, it may modify step specifications (recipes) with machine-dependents settings, or it may insert transport steps to get the material to the machine.

7.3.4.3.2 Notice in Figure 11 that tasks of a higher level are decomposed into combinations of tasks for lower level resources. Each manufacturing resource has a thread of tasks that must be coordinated with the tasks of other resources. The higher level task is completed when the combination of lower level tasks is completed. Figure 11 illustrates this with the coordination of material movement and processing operations and with the coordination of operations within the processing equipment. Resource maintenance, advanced process control calculations and other tasks must also be coordinated with material processing, inspection and movement tasks. The role of job control is to decompose, coordinate, monitor, adjust, and report on this hierarchical structure of tasks.

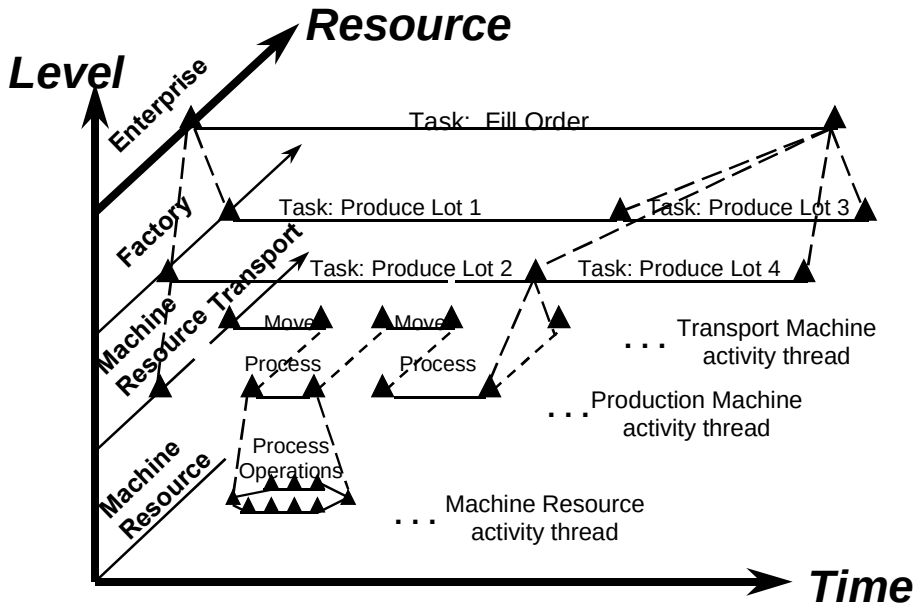

Figure 11
Hierarchical Task Structure

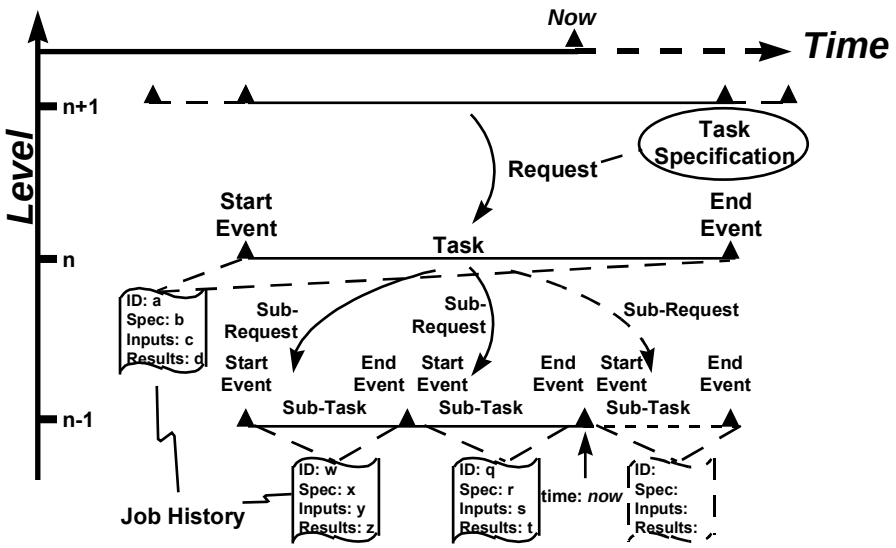

Figure 12
Elements of a Hierarchical Task

7.3.4.4 *Elements of a Task*

7.3.4.4.1 The Hierarchical Task Structure of Figure 11 defines the requirements for a multi-level task model. Figure 12 shows the elements of a hierarchical task, indicating some of the additional data for an overall job control structure. A task is requested from a superior level. Along with the request is a task specification, which is often a template or procedure for how to carry out the task, including a combination of subtasks and monitor and adjustment checkpoints along the way.

7.3.4.4.2 As each task is started, completed, modified or aborted, the job control for the task publishes appropriate events or other notifications. The task endpoints illustrated in Figure 12 as triangles represent factory states or some specific aspect of a factory state. When a task is completed, the factory state is changed in the specified way. For example, after a wafer deposition task, the product state is changed, with a layer of oxide deposited on all the wafers in a lot. When a movement task is completed, the product location state is changed to the load port for a machine to perform the next process operation. These intended factory states often have side effects on other factory states. For example, after a process operation, the equipment state changes to reflect its utilization, consumables consumption, etc. Job control achieves orderly, coordinated, efficient changes to factory state that result in products, that is, that efficiently turn raw wafers into product wafers.

7.3.4.4.3 As the job control executes the task, task results are collected in a job history that includes a job identifier, the (modified) specification used, and recorded inputs and results. Higher level job results could be a roll-up, an abstraction, or simply pointers to lower level job results.

7.3.4.5 *Job Structure*

7.3.4.5.1 A job is a relationship between all the elements necessary to perform a task with specific control functionality to carry out the task in light of manufacturing variability. A job implements a specified task on specified material using specific resources. The job structure of Figure 13 is the relationship mechanism for controlling and changing factory material and resource states and recording job results in associated job histories. That is, the job instantiates and manages the relationships in the structure of Figure 13.

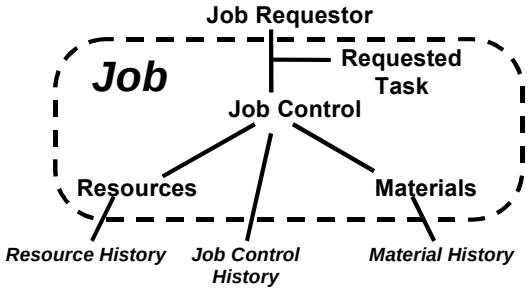

Figure 13
Job Structure

7.3.4.6 *Jobs and Job Supervisors*

7.3.4.6.1 Jobs are transient entities. They are created to perform a task and go away when the task is complete. A job supervisor is the persistent object that responds to job requests by creating jobs. Figure 14 shows an information model for the generic Job Supervision architecture.

7.3.4.6.2 There is a corresponding job supervisor which specializes the generic job supervisor; a Factory Job Supervisor manages Factory Jobs; an Area Job Supervisor manages Area Jobs; and a Production Machine Job Supervisor manages Production Machine Jobs. There is also a Maintenance Job Supervisor and an Advance Process Control job supervisor (the Control Execution Manager in the Advanced Process Control component).

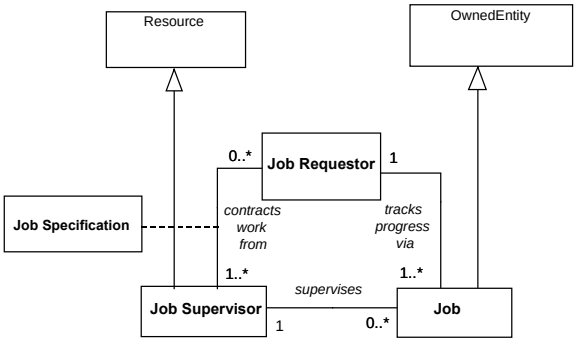

Figure 14
Job Requestors, Job Supervisors, and Jobs

7.3.4.6.3 The entities in Figure 14 are as follows:

- Job.
  - A unit of work that takes time and may fail.
  - Has state (see Figure 15).
- Job Supervisor.
  - Receives requests for work, facilitates creation of a job for the task and returns a reference to that job to the requestor.
  - Manages all jobs within the component that implements it.
- Job Requestor.
  - Requests work.
  - Receives job progress through interface methods (informJobStarted, informJobCompleted, informJobTerminated) and published events.
- Job Specification.
  - Job description (process flow, recipe, transport destination, maintenance spec, etc.).
  - Priority and deadline.

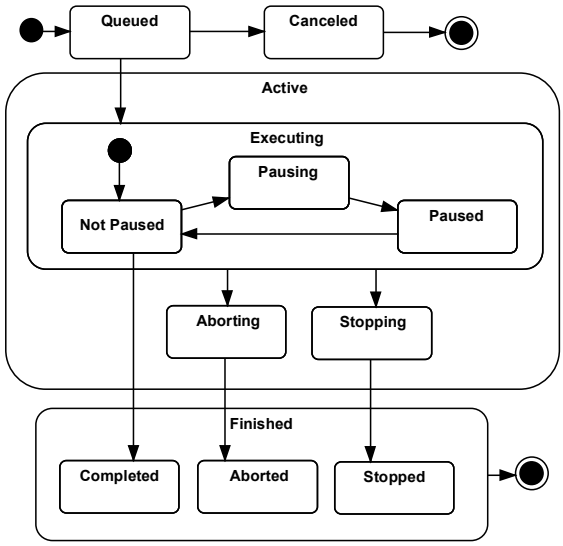

Figure 15
Job State Model (CIMFW)

7.3.4.7 *Job Supervision Design Principles*

7.3.4.7.1 Jobs and Job Supervision are encapsulated within a component. Their interaction and the division of responsibility between them is hidden. At any level, the requestor of activity will request work of a Job Supervisor and receive in return the handle to a Job which represents this work. This requestor will not have visibility to how the Job Supervisor performs the work beyond what is specified in the original activity specification and what is reported later as data. Not visible or accessible are the Job Supervisor's internal and lower-level activity requests (and resulting jobs).

7.3.4.7.2 A job requestor does not micromanage the job. Instead, the job requestor creates a specification of the work to be done and hands it to the Job Supervisor when requesting the work. Once the work request is accepted, the Job Supervisor's component controls the execution of that job within the limitations of the job specification and its business rules. The only exceptions are coarse commands such as job abort or pause. The responsibility of the Job Supervisor and Job is to perform the activity and report back to the requestor on the success or failure of the effort.

7.3.4.7.3 There is no predefined limitation on the facilities within the factory that may be used to satisfy a job request. However, a Job Supervisor may be

configured to be limited to specific factory resources that it can call on to perform work.

7.3.4.7.4 The job requestor has an interface so the Job/Job Supervisor can report overall job progress. The Job Requestor interface includes the methods requesting that it be notified that a job has started, completed, or been terminated. The requestor or other components can subscribe to job state change events and other events for a more detailed job status.

7.3.4.7.5 The Job Supervisor interface provides high level information about the jobs it is currently managing. The details of any specific job are provided by the Job itself, not the Job Supervisor. From the Job Supervisor, the Job Requestor is able to

- request activity,
- locate jobs that meet certain criteria,
- request lists of all jobs being performed, and
- control all the jobs as a group (e.g. "abort all jobs"), but not individually (not "abort job X"—this would be a responsibility of the Job interface).

NOTE 13: Much thought should be given to the use of these "all jobs" commands since they can cause a great deal of harm if not designed and used properly.

7.3.4.7.6 The Job interface is the sole source of all public information about a job. It also provides all specific control of the job (pause, abort, etc.).

7.3.4.8 *Hierarchical Job Structure*

7.3.4.8.1 The factory and task hierarchy of Figure 11 and Figure 12 results in a multi-level (hierarchical) job control structure, with a given Job/Job Supervisor requesting work through other (sub)Jobs. As Figure 16 illustrates, higher level job controllers specify and request work from any number of lower level job controllers and monitor job progress through status and external feedback. At the lowest level, job control results in directly manipulating equipment actuators based on real-time sensor feedback. (Note, this lowest level of job control is outside the MES scope of the CIM Framework—it is within the scope of the equipment control.) The job request, status, and feedback information are all candidates for storage in job history.

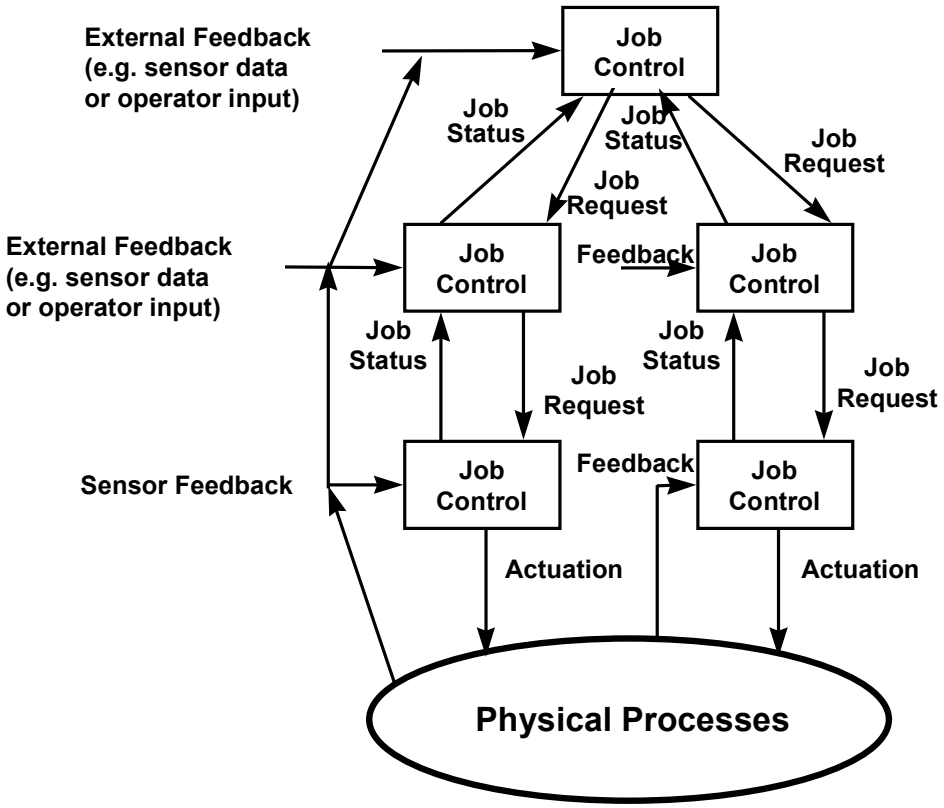

Figure 16
Hierarchical Job Control

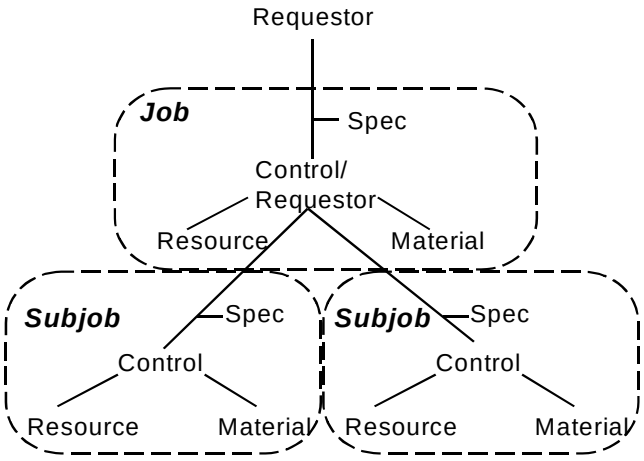

Figure 17
Hierarchical Job Control Pattern

7.3.4.8.2 Placing the job structure template of Figure 13 into the job control hierarchy of Figure 16 results in the hierarchical job control pattern of Figure 17.

7.3.4.8.3 At each level, the job control

- Receives job requests with associated job specifications,
- Decomposes the job into subjobs,
- Defines resource relations between the job and subjob (e.g., selects and schedules resources for subjobs within the scope of the job supervisor, such as machines assigned to an area or process chambers in a production machine),
- Defines material relations between the job and subjob (e.g. how lots are assigned to transport groups and process groups, including decisions on batching, splits, joins, etc.),
- Requests the subjobs and provides associated subjob specifications,
- Monitors subjob progress,
- Reports job progress to the job requestor and other interested functions,
- Records job history.

7.3.4.9 *CIM Framework Job Structure Summary*

7.3.4.9.1 Figure 18 shows the job hierarchy pattern applied to the material movement resources of Figure 7. Figure 19 shows the pattern of Figure 17 applied to the material processing resources of Figure 7 (branching to multiple lower-level resources is not shown). Figure 20 summarizes how the CIM Framework components come together into an integrated manufacturing execution system.

- The product material hierarchy and genealogy is modeled in the Product Management components of the CIM Framework Specification.
- The task specification hierarchy is modeled in the Specification Definition and Recipe Management components.
- The resource hierarchy is modeled in the Factory Operations (for Factory and Area levels), Equipment Tracking and Maintenance, Durables and Consumables Management, and Factory Labor components.
- The control hierarchy is modeled in the Factory Operations (Factory Jobs), Production Machine (Production Machine Jobs), Material Transport and Storage (Transport Jobs), and the maintenance job supervision aspects of the various Resource Tracking and Maintenance specializations (Maintenance Jobs).

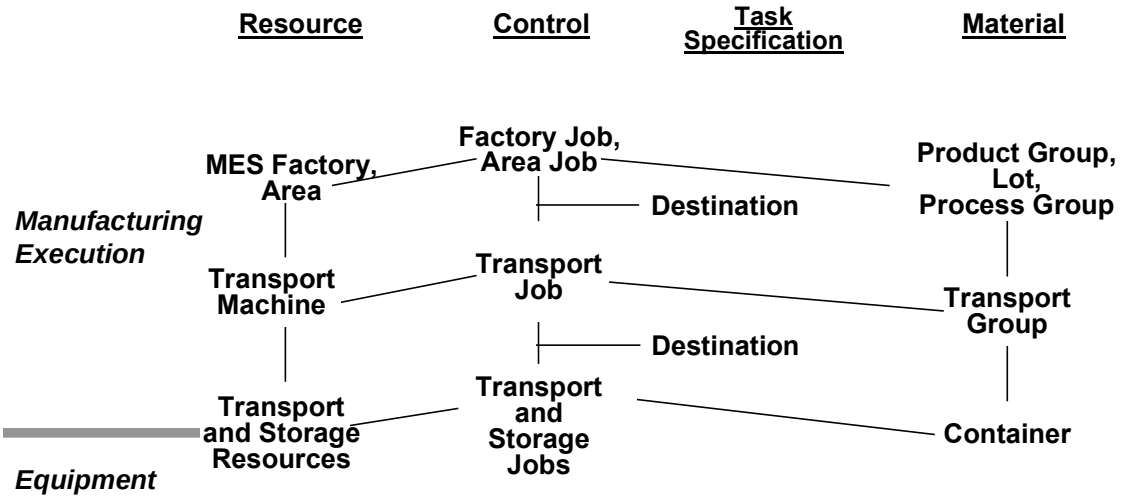

Figure 18
Hierarchical Job Control - Material Transport Aspects

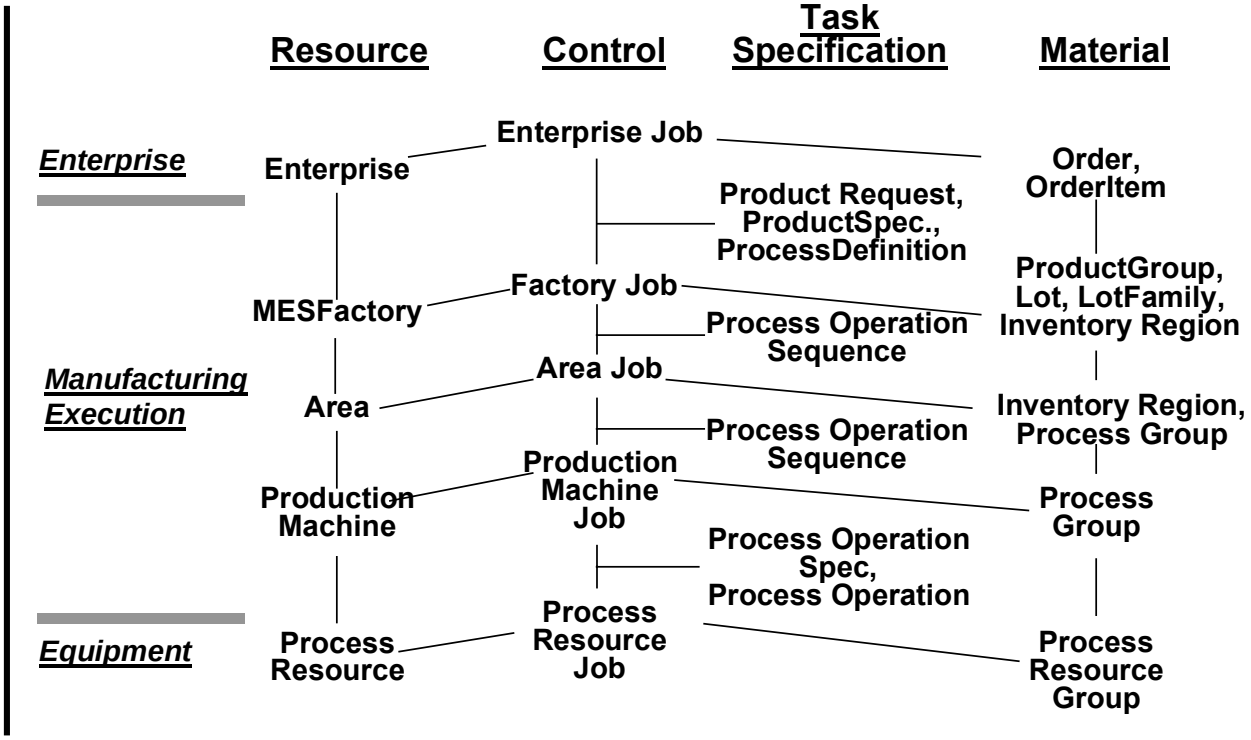

Figure 19
Hierarchical Job Control - Material Processing Aspects

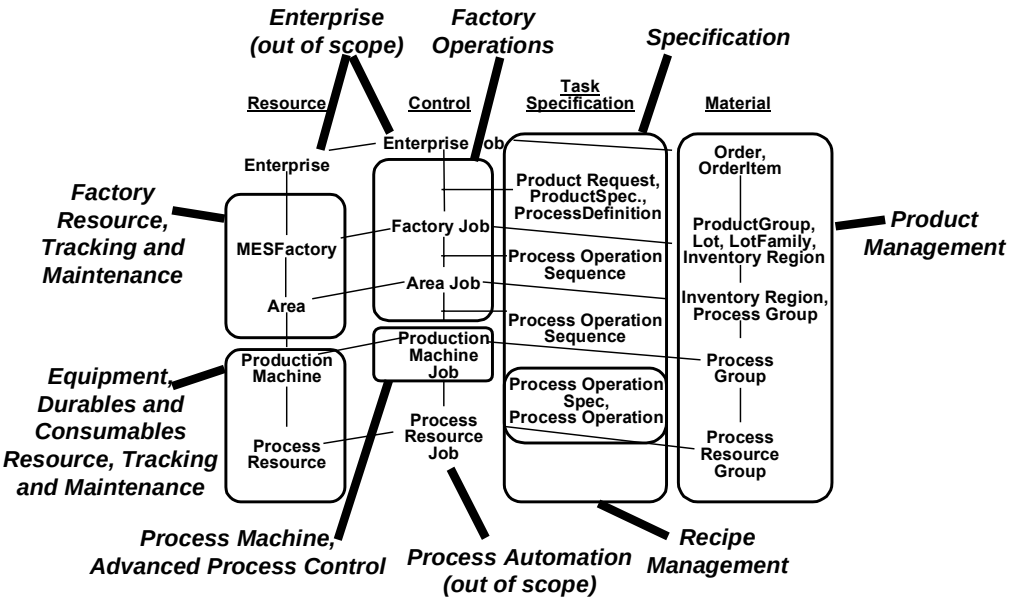

Figure 20
Integrated Hierarchical Job Control

7.3.4.9.2 As Section 7.3.3.3 describes, each of these job control views shows a job supervision function that performs the tasks for each level of the factory resource hierarchy.

7.3.4.9.2.1 Factory Job Supervision: Factory Operations satisfies enterprise product requests. It releases material and creates Factory Jobs which drive material work-in-progress through its process flow. The Factory Job Executor requests, schedules, monitors and coordinates supporting jobs: production machine jobs, transport jobs, maintenance jobs, etc.

7.3.4.9.2.2 Area Job Supervision: This optional level serves Factory Operations and is responsible for multiple machine activities (areas may map to bays, cell controllers, linked lithography, etc.). This level is not required, but is available for organizing large factories. Area job supervision could support factories that have a factory within a factory model with separate but integrated operations management for separate units of manufacturing capacity. It could also support factories transitioning from legacy systems, with parts of the factory under a CIM Framework-conformant MES and other parts under a legacy MES that is fronted or wrapped by a CIM Framework conformant Area Job Supervisor component. Another example could be a factory that provides a pseudo-cluster cell controller to group stand-alone process equipment into an integrated workcell that behaves like a cluster tool. The CIM Framework does not specify separate Area Job Supervision interfaces. These functions are met by the Production Machine Job interfaces with no change—the same interface serves both functional roles. Areas typically receive a sequence of process jobs as a request (implemented by requesting lower-level process jobs and material transport jobs), whereas simple machines usually receive a single process job (implemented through direct interaction with process equipment).

7.3.4.9.2.3 Machine Job Supervision: This level lies within the CIM Framework Production Machine component. It accepts activities that apply to a single Machine. It delegates work directly to the physical equipment, either through some equipment interface driver or directly through the GEM/SECS interface.

7.3.4.9.2.4 Transport Job Supervision: This level lies within the Material Transport and Storage component. It takes requests to move material from one location to another. While the CIM Framework provides for layers of Transport Job Supervisors in a complex interbay and intrabay material handling system, today's typical installation has a single material handling system controller (a single Transport Job Supervisor). Complex production machines with internal material handling and storage may also be Transport Job Supervisors.

7.3.4.9.2.5 Maintenance Job Supervision: Maintenance is a preventive (scheduled) or reactive (repair on failure) activity on specific resources (machines, durables, etc.) using labor resources and materials. Maintenance job progress is reported and tracked and maintenance job history is stored. The generic Resource Tracking and Maintenance component defines the

overall maintenance job supervision functions (see Section 7.3.3.5), and the various resource types implement these functions in the Equipment Tracking and Maintenance, Durables Management, Consumables Management, and Factory Labor components.

7.3.4.9.2.6 Advanced Process Control (APC): APC includes lengthy calculation activities that may not succeed and which impact the performance of other jobs. For example, an algorithm may not converge, a sensor may emit bad data, or process recipes and job specifications may be modified. The CIM Framework APC component leverages the overall job architecture, where control strategies and scripts are the job specifications, sensor analysis and control execution environments are scheduled resources, and the state of process specifications (machine settings, modified process flows) are the result of APC job execution.

7.3.4.9.2.7 Enterprise and Process Resource: The job control hierarchy of Figure 19 provides interfaces to the enterprise at the top level and the equipment at the bottom level. The enterprise level is the requestor of work for the factory. Enterprise control is outside the MES functional scope of the CIM Framework. The equipment level is also outside the scope of control of the CIM Framework. It is usually implemented by process controllers embedded in or piggyback on equipment. The equipment level is modeled for the MES to allow process data to be collected and organized for specific equipment resources such as a specific chamber in a cluster tool.

7.3.4.10 *Job History*

7.3.4.10.1 Any entity can have an associated CIM Framework History. The CIM Framework defines some specific named histories, as Figure 21 illustrates. Product-oriented data is captured in production history associated with the material hierarchy and is used, for example, for material traceability and defect analysis. Process-oriented data is captured in process history associated with the job supervision hierarchy and is used, for example, for run-to-run process control. Equipment and other resource-oriented data is captured in resource history associated with the resource hierarchy and is used, for example, for preventive maintenance and warranty tracking.

7.3.4.10.2 As jobs are executed (and possibly broken down into sub-tasks as shown in Figure 12), their results are captured as various types of histories. For example, an Area Job's history may be captured within an InventoryRegionHistory. A Resource Level Job's history may be captured within an E10PerformanceHistory or within a ResourceMaintenanceLogHistory.

7.3.4.11 *Task Specification and Workflow*

7.3.4.11.1 The specification of a task takes various forms. For Factory Jobs, the specification is the process flow defining the process steps to transform material into products. For Production Machine Jobs, the specification is the specific steps in the process flow that this production machine is to perform, along with process recipes. For Transport Jobs, the specification defines the destination location for the material transport group. For maintenance jobs, the specification defines the specific maintenance tasks for the resource. For advanced process control jobs, the specification defines a script that includes the sensor processing and algorithm steps of a control strategy.

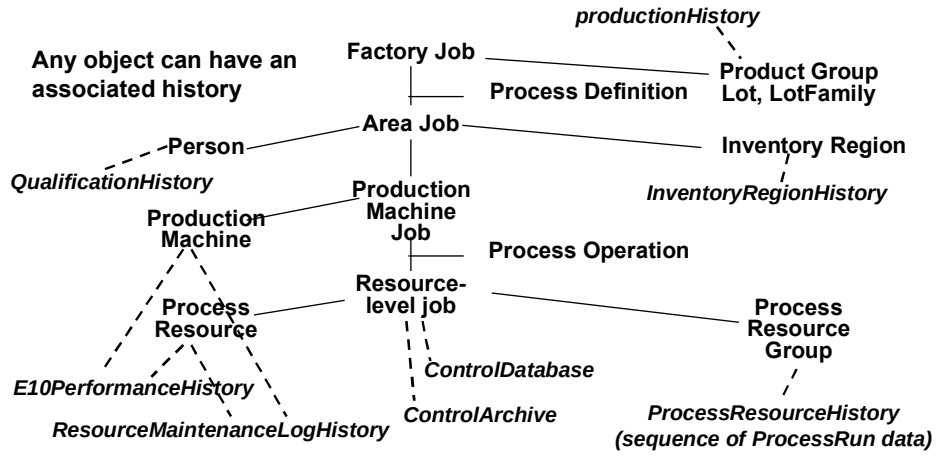

Figure 21
History Structure

7.3.4.11.2 In general, each task specification is a definition of a business process and a task workflow to carry out that business process. There is a diversity of information that could appear in a business process, including associated operational business rules and constraints, workflow traversal logic, and data structures for tasks, recipes, steps, etc. There is not industry consensus on representing these generic task specification details. For each specific job type, the CIM Framework defines some specification structure, but leaves room for suppliers and users to define additional data and behavior and to leverage workflow engines and evolving workflow standards [Workflow, WfMC].

7.4 *Specification Conventions*

7.4.1 This section provides an overview of the notations required to express the CIM Framework specifications. These notations provide the representational formalisms for all CIM Framework specifications. This section is a reference rather than a tutorial. Hence, those unfamiliar with these topics are strongly encouraged to consult the materials listed in the references for a more thorough explanation.

7.4.2 CIM systems requirements and the architectural principles upon which the CIM Framework was founded were derived through use of industry standards, practices reported in the literature, and other state-of-the-art information. The requirements of the CIM Framework are specified using a combination of graphical and textual notations. Where applicable, methods were employed and notations applied that were supported by the automation of computer-aided software engineering tools.

7.4.3 *Model Specification and Graphical Notation Usage*

7.4.3.1 This section briefly describes the graphical notations used to specify the semantics of the framework, including the following:

- Component Relationship Model.
- Component Information Model.
- Component Interaction Diagram.
- State Model.

7.4.3.2 *Component Relationship Model*

7.4.3.2.1 The Component Relationship Model was developed specifically for the framework specification as a mechanism to show relationships among framework components. It shows the logical combination of components and the relationships among the component parts. Figure 22 depicts the Component Relationship Model.

7.4.3.2.2 This model is based on the Unified Modeling Language (UML) Class Diagram [UML] with added stereotypes to represent the components. The UML Association concept is abstracted to represent the high-level relationships between components of the framework.

7.4.3.3 *Component Information Model*

7.4.3.3.1 The Component Information Model shows the specified CIM Framework interfaces along with the relationships between those interfaces. This model is based on the UML Class Diagram [UML] with the stereotype «Interface» to indicate that the classes are interfaces rather than implementation artifacts. This model shows generalization as applied to interface inheritance, aggregation of interfaces, and the use of an Association Class to represent data associated with an association between two interfaces. The associations also capture specific semantics of the relationship, including multiplicity and optionality as shown in Figure 23.

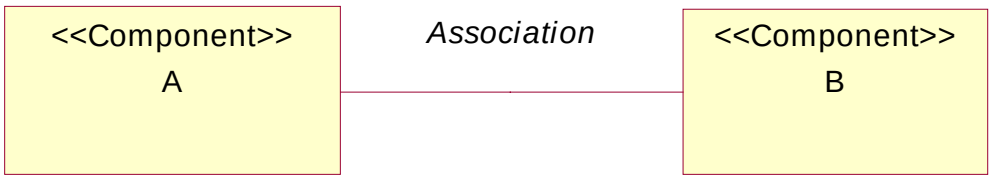

Figure 22
Component Relationship Model

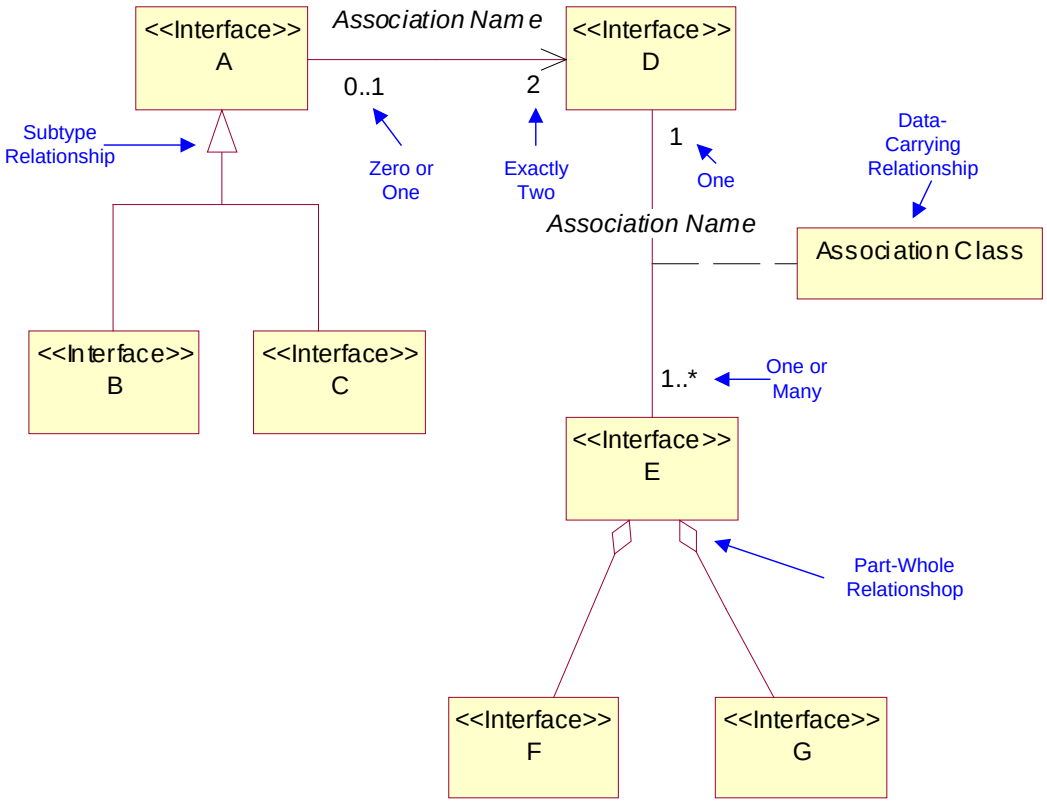

Figure 23
Information Model Example

7.4.3.4 *Component Interaction Model*

7.4.3.4.1 The Component Interaction Model expresses framework dynamics by describing the sequence of collaborations between objects supporting CIM Framework interfaces. This model is represented using the UML Sequence Diagram [UML]. The vertical lines each represent an object that conforms to a role specified by a CIM Framework interface. This is called a "lifeline." The connecting arrows represent messages that flow between these objects and the data they convey. Each message must map to a defined operation on the interface of the message recipient. The example shown in Figure 24 of a Sequence Diagram models a portion of the interaction between a bank customer and an Automated Teller Machine (ATM).

7.4.3.5 *State Model*

7.4.3.5.1 The State Model shows behavior associated with CIM Framework interfaces as changes in state that result from specific events. The required behavior for an interface is conveyed through UML State Diagrams [UML] and textual tables that offer supporting details. Under this notation which is based on Harel Statecharts, states may be divided into substates, thereby forming a hierarchy of states. Substates must be one of two types, termed AND substates (representing concurrency) and exclusive OR substates (representing a finer breakdown of a parent state).

7.4.3.5.2 Object State Tables provide supplementary (to the State Diagram) state information including descriptive state definitions, state query mechanisms, the triggers effecting state transition, and the actions resulting from state transitions.

7.4.3.5.3 *Object State Definitions and Query Table*

7.4.3.5.3.1 The Object State Definition and Query Table provide supplementary information to the State Diagram.

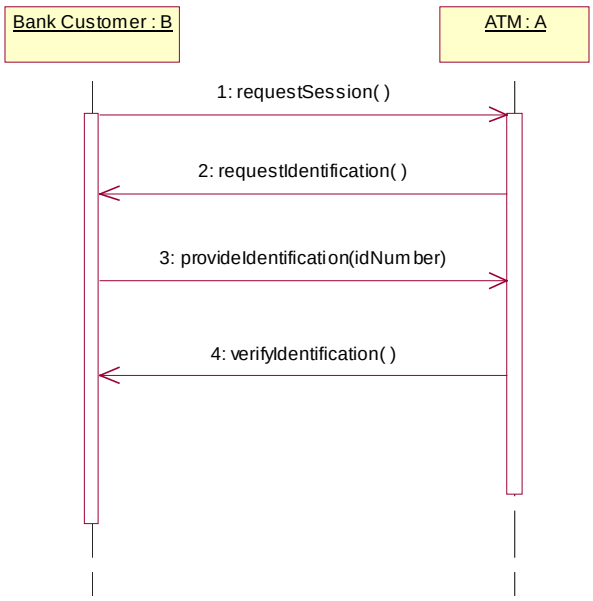

Figure 24
Component Interaction Model Example

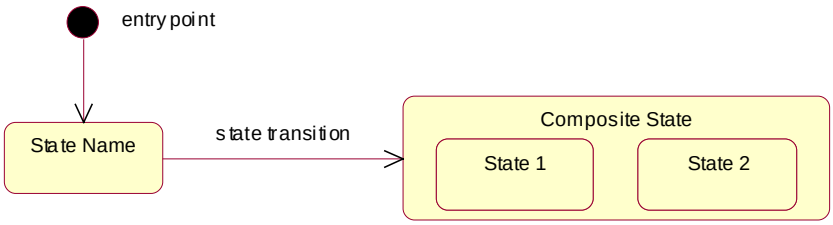

Figure 25
State Diagram

7.4.3.5.3.2 Table 1 provides a detailed description of each object state and identifies the query mechanism for determining if the object is in that state. Given the example provided in this section, an entry in the State Definitions and Query Table would appear as shown.

7.4.3.5.4 *Object State Transition Tables*

7.4.3.5.4.1 Another supplement to the state model is the Object State Transition Table (Table 2).

7.4.3.5.4.2 It lists the transition from the state diagram identified by the starting and ending states, and the event that causes the transition between these states.

7.4.3.5.4.3 Within the CIM Framework specification, only those triggers and state changes relevant to external interfacing are shown to help define how an external entity (in this case a driver) interoperates with an object (in this example an automobile).

7.4.3.5.4.4 While this table defines triggers for state transitions, there is no guarantee that the transition will take place in response to the trigger. In the above example, if the car is out of gas the engine will not go to the running state; if the light is burned out, it will not transition to the on state; depressing the accelerator pedal when the auto is in the off state will have no effect, etc. In the CIM Framework, object state can be queried to ensure successful transitioning in response to a triggering message. Events are also used in some cases to inform the client of a transition or change of state.