The MMST Computer-Integrated Manufacturing System Framework

John McGehee, *Member, IEEE*, John Hebley, and Jack Mahaffey

***Abstract*— Computer-Integrated Manufacturing (CIM) is the harmonious connection, integration, and interoperation of automation equipment within a manufacturing facility. In a semiconductor wafer fab, this includes integration of the processing equipment with all of the supporting systems for product and process specification, production planning and scheduling, and material handling and tracking. Traditionally, CIM systems have been characterized as monolithic mainframe-based systems and/or inflexible islands of automation with limited interoperability. Today's manufacturing demands fully integrated dynamic systems which directly support the concepts of lean, flexible and agile manufacturing to high quality standards. These requirements drove the design of a new CIM system which was developed for the Microelectronics Manufacturing Science and Technology (MMST) program. This paper provides an overview of the MMST CIM system framework which is based on open distributed system and object technologies. The CIM system was demonstrated in a 1000 wafer pilot production run in 1993 which achieved world record cycle time, and is now being commercialized as part of the WORKS™ product family from Texas Instruments.**

I. Introduction

SEMICONDUCTOR wafer fabrication is a very complex and capital-intensive manufacturing process [1]. A modern wafer fab, costing several hundred million dollars to construct, contains several hundred machines and personnel running three shifts per day, seven days per week to produce tens of thousands of wafers per month. Production of each wafer requires several hundred process steps, each of which must be carried out in a hostile, near particle free enivornment to exacting tolerances.

Complexity and cost continue to increase with each new generation of semiconductor product. Approximately five years of process and equipment development are required to achieve reasonable quality and yield for next generation devices [2]. Each generation requires more process steps with tighter tolerances, and smaller geometrics on larger dies.

Compounding the increasing complexity of semiconductor process and equipment technologies is a massive shift in manufacturing strategy brought about by global competitive pressures [3]. Semiconductor companies are being forced to move from high volume production of commodity parts to low volume, flexible and leaner production of application-specific parts. Pressure to reduce cost is leading to a re-thinking and in several cases a re-engineering of virtually every aspect of semiconductor manufacturing.

Computer-Integrated Manufacturing (CIM) offers the greatest hope for dealing with this increasing complexity. Unfortunately, CIM systems of today fall far short of meeting the challenge. Typically implemented as either monolithic mainframe-based systems or distributed isolated islands of automation, traditional CIM systems suffer numerous problems:

- limited flexibility due to a lack of design for change;
- nonintegrated due to lack of a common system architecture and design model;
- limited functionality because a single supplier cannot provide full functionality or because multiple suppliers cannot provide applications that can be integrated;
- high development costs and poor quality due to a lack of reusability and to the use of poor or outdated development methodologies, processes, and tools;
- high maintenance cost due to designs and implementations which are difficult to understand, modify and extend;
- high procurement, installation, and support costs due to the use of outdated architecture and hardware delivery technologies;
- difficult to use due to the use of nonintuitive text oriented user interfaces.

Clearly a new, revolutionary approach to semiconductor manufacturing is required to cost effectively meet the integrated circuit needs of semiconductor manufacturers and their commercial and military customers. This was the conclusion drawn by TI and the DoD when the ideas for the MMST research and development program were conceived. A contract for this program, sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Wright Paterson Laboratory was awarded to TI in October of 1988.

The objectives for the MMST program are covered in [4] and summarized in Table I. Achieving these objectives required significant innovations in all aspects of wafer manufacturing, including new rapid thermal processes, in situ sensors, modular processing equipment, and development of a next generation CIM system architecture which could exploit the capabilities of these new technologies while addressing the deficiencies of current CIM technology. This paper provides an overview of the MMST CIM system framework which is based on open distributed systems and object technologies. Additional background on the CIM system can be found in [5] and [6].

Manuscript received August 1, 1993; revised November 10, 1993 and December 22, 1993. This work was supported in part by the Air Force Wright Paterson Laboratory and the DARPA Microelectronics Technology office under Contract F33615-88-C-5448.

The authors are with the Information Technology Group, Texas Instruments, Inc., P.O. Box 655012, MS 3635, Dallas, TX 75265 USA.

IEEE Log Number 9216487.

WORKS™ is a registered trademark of Texas Instruments, Inc.

TABLE I
COMPARISON OF MMST AND CONVENTIONAL PROCESSING

Current Technology	MMST Technology
Large, fixed 20,000 wafer capacity	Modular, 100–1000 wafer/month capacity
Factory cost > $500M	Factory costs $30–50M
Silicon only	Generic manufacturing (HgCdTe, GaAs)
Class 1 cleanroom	Class 1000 cleanroom
30% of processes use liquids	Liquid free processing
25–50 wafer batches	Single wafer processing
Batch sampling	Real-Time process control
1–3 month cycle time	5–15 day cycle time
Mixed equipment configurations	Modular vacuum processing equipment

II. MMST CIM SYSTEM REQUIREMENTS

The requirements for the MMST CIM system were derived through an analysis process which was based upon the objectives for the MMST program as a whole. Essentially, the CIM system was required to integrate and automate the wafer manufacturing process both horizontally (i.e., order entry through finished wafer output) and vertically (i.e., production planning through embedded machine control). Specifically, the following functions were to be provided:

1) planning of factory operation from order entry through wafer production;
2) scheduling of factory resources to meet the production plan;
3) modeling and simulation of factory operation;
4) generation and maintenance of process and product specifications and recipes;
5) tracking of work-in-progress (WIP);
6) monitoring of factory performance;
7) machine monitoring, control and diagnosis;
8) process monitoring, control and diagnosis.

In addition to these general functional requirements, a number of additional MMST requirements were derived from the principal objectives of reduced cost and cycle time with higher flexibility and quality. These requirements are shown in Table II and together they formed the basis of the subsequent CIM system architecture analysis, design, and implementation phases. The following sections provide an overview of the architecture, including the fundamental principles, application partitioning, system services, and development of a generic framework for semiconductor manufacturing.

III. CIM SYSTEM FOUNDATION

As discussed in the prior sections, the vision for the MMST factory of the future was driven by the principle objectives of flexibility, reduced cycle time, improved quality, and low cost. At the outset of the program, it was recognized that the CIM system if properly designed could have significant impact on each of these objectives. Flexibility could be enhanced by designing a CIM system which supports single wafer as well as to batch processing. Cycle time could be reduced by a CIM system with a bottleneck resource-regulated scheduling strategy which reduces WIP. Quality could be improved with CIM system support for automatic material identification

TABLE II
MMST CIM SYSTEMS REQUIREMENTS AS DERIVED FROM FACTORY OBJECTIVES

MMST Factory Objective	Derived CIM System Requirements
Cycle Time	- pilot wafer elimination - WIP reduction/tracking - dynamic look-ahead scheduling of bottleneck resources - capacity based planning - factory simulation/modeling - workload balancing - resource-regulated material release - high performance application software
Cost	- distributedUnix workstation-based implementation - incremental/modular expansion - support for automation - paperless operation - increased application development productivity - decreased application maintenance - JIT planning/scheduling - cost modeling/tracking
Flexibility	- single wafer processing - incremental/modular expansion - scaleable application software - machine independent process recipes - integrated applications - dynamic system extension/evolution - distributed client-server architecture - automatic configuration/application binding
Quality	- process control/diagnosis - process recipe management/upload/download - automated material identification - process data collection/analysis - factory/machine performance monitoring - preventive maintenance tracking/scheduling - SQC - high quality application software - effects-to-settings translation

and recipe download to eliminate operational errors, and embedded process control which can keep each process in tune. System purchase, operation, and maintenance costs could be significantly reduced with appropriate choices for CIM system technology. Finally, and perhaps most importantly, all of these objectives could be facilitated with the highest quality of software engineering.

Traditionally, CIM system software has been characterized as expensive to develop, inflexible, and difficult to maintain or extend, all of which are characteristic of what is often referred to as the "software crisis" [7]. Fortunately, a new paradigm has emerged which specifically addresses each of these problems and provides a significant lift in overcoming them. The paradigm is based on object technology, and it represents a revolutionary new approach to all phases of software development from analysis and design, to implementation and operation. A detailed description of the object-oriented approach can be found in [8]. In a nutshell, it is based on four fundamental principles.

1) The object abstraction consisting of both data and procedures (called methods) which enables software entities to model the real world, making them more understandable and stable over time.

2) Encapsulation of data and behavior into objects which enables them to be changed without affecting the clients of those objects.
3) Inheritance (of data and behavior) and composition which facilitate the creation of reusable software components.
4) Dynamically bound polymorphic behavior which enables the construction of applications, tools, and frameworks which can accommodate a wide variety of object types without type specific code.

Together these principles provide a powerful new foundation for software engineering in general and system architecture in particular. The MMST design team adopted the object-oriented approach whole-heartedly as the foundation upon which to base the CIM system architecture. The CIM system was developed using object-oriented analysis, design, and rapid prototyping methodologies, and is implemented in an object-oriented environment and programming language (Smalltalk). Each CIM application is comprised of a set of objects with well defined services and graphical object-oriented direct manipulation user interfaces. Further, the fundamental principles and attendant advantages of the paradigm at the object level have been exploited at the subsystem level as well, through use of an object-oriented framework design concept which is discussed in a section VI.

IV. CIM System Architectural Principles

In addition to its object-oriented foundation, the CIM system architecture was based on a number of other fundamental principles. These principles were a direct reflection of the vision of the system architects and were used to set the overall architectural direction and to tradeoff alternatives in architectural design. The most important of these principles are discussed in the following paragraphs:

Distributed Object-Oriented Peer-to-Peer Architecture: The architecture is fundamentally object-oriented for reasons mentioned earlier. Applications are distributed to the point of most need, and their services are available to distributed clients. Distributed applications enhance performance and fault tolerance, and facilitate incremental flexible expansion.

Fully Integrated Dynamically-Bound Applications: Each application is comprised of objects with well known services. External applications needing those services deal directly with the objects that provide them, enhancing performance and fault tolerance. For example, an application needing to know the status of a particular machine sends a message to the corresponding machine object.

Additionally, applications are dynamically bound to the greatest extent possible. A client object doesn't bind with a server object until the point in run time when the service is needed. For example, when a wafer is ready for the next process step, the scheduler considers all machines capable of performing that step at that instant, including any that may have just been brought on line. Dynamic binding makes the system more resilient and adaptive to instantaneous changes in factory status or configuration.

Intelligent Proactive Pull-Oriented Object Model: Objects are given as much intelligence as is practical and take on proactive role in improving the state of their domain. For example, machines will actively seek work when they are idle. They also know their preventive maintenance (PM) requirements and will request PM services when due. This pull-oriented object model also serves to enhance the operation of the factory. Material is not released into the factory until there are resources available to process it, reducing work-in-progress (WIP) and decreasing cycle time.

Event-driven Object Interaction: Objects are automatically notified whenever an aspect of interest of an object of interest changes. This eliminates the need for polling for object status, reducing CPU overhead and increasing system responsiveness. Furthermore, the objects to be notified are dynamically bound and are transparent to the object which has changed.

Universal Object Communication With Transparent Object Location: Any object in the system can communicate with any other object without knowledge of or regard for (short of transport delay) its location. Restrictions are applied for certain objects for safety or security reasons and to protect encapsulation. Location transparency reduces application code volume and complexity and facilitates portability.

Object Purity First; Compromise Only Where Essential Maintaining object purity throughout the system is essential if all the benefits of the paradigm are to be realized. Exceptions introduce limitations which destroy the simplicity, power, and ease of use.

Once a pure object design is completed, analysis of that design will indicate if the paradigm must be violated, usually either for capacity or performance reasons.

Graphical Direct Object Manipulation/Navigation UIF Paradigm: User interfaces are object-oriented and users interact with the system following the paradigm of direct object manipulation. That is, objects on the screen should represent familiar real-world objects with which a user interacts by addressing (pointing at) them and selecting what he wants them to do. In addition, he can navigate from an object to any other object that it references. For example, it is possible to navigate from the factory object to any object in the factory or from a machine object to any object within the machine.

Generic High-Level Abstractions: Every limb in the object class inheritance hierarchy is rooted in a series of rich abstractions that are generic across application domains. For example, while leaf nodes on the limb might be specific to Semiconductor Manufacturing, root nodes are applicable to manufacturing within any industry. This enables the CIM framework to be used in alternative manufacturing domains.

Any Application/Screen From Any Workstation: An application can be executed on any node in the system and have its user interface either locally or remotely located. For example, it is possible to execute the Planner application on one workstation with its user interface on another. This promotes flexibility by separating locality of use from locality of execution.

Any Platform/Operating System/Network: Applications are portable to the greatest extent possible to enable transparent exploitation of advances in computer hardware and system

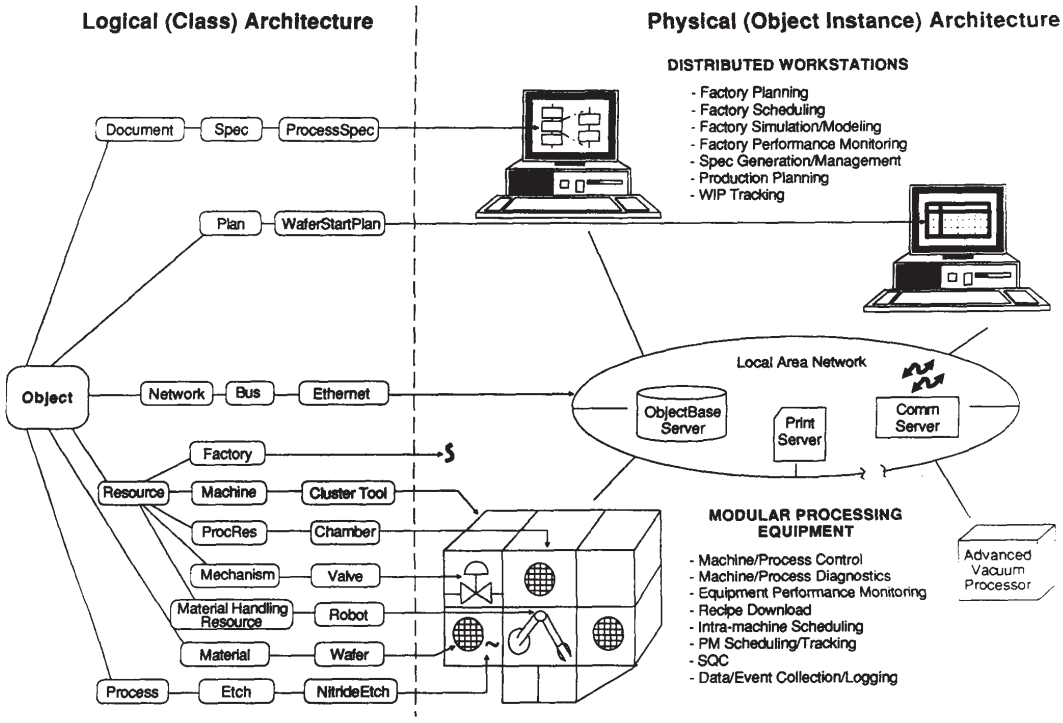

Fig. 1. Object-oriented system architecture.

software. To a large extent, this feature is facilitated by the chosen environment (Smalltalk).

Standards Where Appropriate: Adoption of appropriate industry standards simplifies the requirements, reduces development time and effort, facilitates portability, and enhances resale. Several standards were adopted at the outset (e.g., UNIX™, X Windows™, Motif™, Ethernet™, TCP/IP, Sockets, SECS, SEMI GEM) and several others are being tracked for possible future incorporation (e.g., several OSF and OMG standards).

V. CIM System Architecture Overview

The architecture of a distributed object-oriented system is multidimensional. Examples of the two most important dimensions, the logical and the physical, are shown in Fig. 1.

The logical architecture is represented by the object class inheritance hierarchy. At the top of this hierarchy lie the most abstract and generic classes (e.g., Document, Plan, Resource, Material). These abstractions are reusable across domains. For example, Document contains the data (e.g., revision number) and behavior (e.g., print) required for all types of documents within an Enterprise. At the bottom of the class hierarchy lie the most concrete and specialized classes (e.g., Process Flow Spec, Wafer Start Plan, Cluster Tool and Wafer). These abstractions, generally only applicable to semiconductor manufacturing, inherit the data and behavior of all of their superclasses and specialize them, either through adding data and behavior, or by overriding that which is inherited.

The physical architecture is represented by the object instance hierarchy as shown on the right hand side of Figure 1. Objects (instances of leaf node classes in the logical hierarchy) are grouped by many "using" or "containing" relationships and distributed to the various compute nodes throughout the factory. For example, a Machine (object) contains many other objects (e.g., Chambers, Valves, Robots, Pumps). These are instantiated on the various computers within the machine, such as the host user interface computer and one or more embedded process control computers.

Each instantiated object (e.g., Robot) has a set of responsibilities (e.g., material movement) and provides a set of services (e.g., moveMaterial) to other "qualified" objects within the system. Other objects may be involved in providing the service (e.g., the Robot must collaborate with both the source and destination objects for material handoff). Inter-object messaging is used for all object interaction, and parameters associated with these messages, both passed and returned, are also objects.

UNIX™ is a registered trademark of AT&T.

X Windows™ is a registered trademark of the Massachusetts Institute of Technology.

Motif™ is a registered trademark of the Open Software Foundation.

Ethernet™ is a registered trademark of the Xerox Corporation.

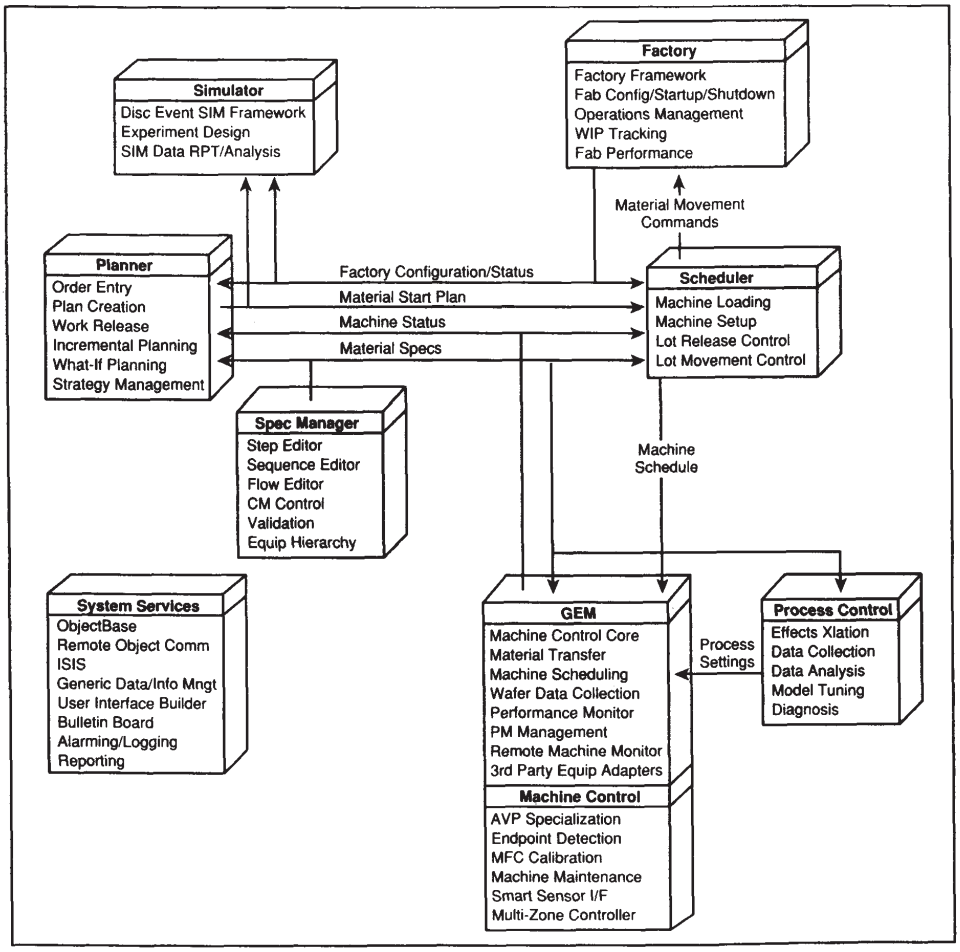

Fig. 2. MMST CIM system applications.

A. CIM System Services/Application Environment

The services provided by the CIM system have been partitioned into 8 major applications as shown in Fig. 2:

- Factory Manager,
- Factory Planner,
- Factory Scheduler,
- Factory Simulator,
- Spec Manager,
- Generic Equipment Model,
- Machine Control,
- Process Control.

Each application typically contains several subapplications, any one of which can be executed as a separate UNIX process on any node in the system. Applications interact with each other via messages between their well known objects. For example, if the Scheduler needs to know the status of a particular machine, it sends a message to the instance of the GEM Machine object of interest. A brief overview of each application is provided in the following paragraphs. Refer to subsequent papers in this issue for more detailed information.

Factory Manager: The Factory Manager is logically the highest level application in the CIM system. It contains many factory level abstractions (e.g., Factory, Area, Material, User, Material Transporter) that serve to tie all the applications together (e.g., it is possible to navigate to any object in the factory from the Factory object itself). The Factory Manager provides services for starting up and shutting down the factory and for coordinating the activities of operations management personnel while the factory is running. Fab performance monitoring (e.g., throughput, cycle time) and work-in-progress (WIP) tracking facilities are also provided.

Factory Planner: The Factory Planner is the focal point for the acceptance of factory orders and the generation of

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.