A Manufacturing Planning and Execution Software Integration Architecture

S. C. Feng
Y. Zhang[1]
Manufacturing Systems Integration Division
Manufacturing Engineering Laboratory
National Institute of Standards and Technology
Metrology Building A-127
Gaithersburg, Maryland, 20899
U.S.A.

Abstract

Process planning and manufacturing execution are two major functions in a product development cycle. Industries require these functions to be able to plug and play with other related manufacturing applications, such as design, resource planning, production scheduling, and equipment control. There are information barriers to interoperability among commercial systems in these areas. Based on the activity models for process planning and manufacturing execution, a set of foundation interfaces has been developed to cope with the problem. The interfaces are expected to evolve as the new technology and industrial practices emerge.

Keywords

Interoperability, Manufacturing Execution, Manufacturing Software, Process Planning, Integration Architectures, Systems Integration

[1] Guest Researcher, also affiliated with Tsinghua University

G. Jacucci et al. (eds.), *Globalization of Manufacturing in the Digital Communications Era of the 21st Century*

1 INTRODUCTION

Manufacturing execution system (MES) is a relatively new technology, compared with computer aided design (CAD), computer aided process planning (CAPP), and computer aided manufacturing (CAM) systems. Many industries currently use the technology for managing factory-floor information and activities to increase productivity and improve quality. Examples of factory-floor information and activities are resource allocation, batch/lot dispatching for production, quality management, operation planning, detailed scheduling, labor management, product tracking, and keeping records of product genealogy. MES cannot stand alone in factories. It has to be closely integrated with related systems, such as CAPP, enterprise resources planning (ERP), data collection, scheduling, and equipment control systems. Since these systems run on a variety of computer hardware and software platforms, it is beneficial to industrial users if these manufacturing software systems can exchange data and messages (function calls) with each other in a diverse computing environment.

A feasible way of meeting this industrial need is to develop a mechanism to allow software on a variety of platforms to exchange data and messages with each other. One way to develop such a mechanism is to design a system of standard software sockets so that different software systems can interoperate or «plug and play.» It is possible to develop such a system of interfaces using currently available technology in software «gluing,» also referred to as middleware. A set of foundation interfaces has been developed for manufacturing planning and execution to cope with the integration and interoperability problems. The interfaces are described in this paper.

In addition to this section which introduces the first version of the foundation interfaces for manufacturing process planning and execution software integration, Section 2 describes current methodology and technology for manufacturing software integration. Section 3 contains the interfaces and the context in which they are developed and used. Section 4 concludes the interfaces development. Section 5 has a list of references that are cited in this report.

2 MANUFACTURING EXECUTION SYSTEM

Comparing with CAD, CAM, CAPP, and MRP, MES is a relatively new term defined for a category of industrial software used for factory floor information management (MESA International, 1997a). The primary purpose of MES is to provide an information system for optimizing production activities in a manufacturing facility with the focus on quick response to changing conditions. Subgoals of MES are follows:

1) to improve communication capability inside a facility, e.g., part programs can be electronically downloaded from CAM systems to machine tools, production activities can be reschedule to reflect unexpected machine down time or production priority changes,
2) to improve communication capability between production and other activities in a manufacturing enterprise, such as product design, process planning, resource planning, supplied chain management, service and sales, and equipment control,
3) to monitor production in real time by analyzing data collected from the manufacturing facility,
4) to provide up-to-the-minute communication between the facility and facility management, and
5) to better manage production-related data, including resource data, performance data, process data, job scheduling data, equipment/device control programs, etc.

The goal of this document is to better understand what MES is and its relationship to other manufacturing activities. Specifically, the document contains the following items: a definition of MES, the context in which an MES performs, its functions, and fundamental interface objects of MES.

With the goal of MES, a definition can be given as follows:
MES is a system that consists of a set of integrated software and hardware components that provide functions for optimizing production activities from job order launch to finished products. Using current and accurate data, an MES initiates, guides, responds to, and reports on production activities as they occur. MES provides production activity information to other engineering and business activities in the enterprise and its supply chain via bi-directional communications.

This definition is derived from a definition given in the MESA International White Paper 6 (MESA International, 1997b). In that paper, only major functions in MES are described. The description of what is an MES is missing from the paper.

3 MANUFACTURING PLANNING AND EXECUTION TECHNOLOGY INTEGRATION STATUS

Currently, commercially available manufacturing planning systems and execution systems operate on different hardware and software platforms, and these platforms are almost all incompatible with each other. To address this incompatibility, the Common Object Broker Architecture and specification (CORBA) (OMG, 1996) specifies a mechanism for software objects interoperating with each other on various commercially available computing platforms. In CORBA, a portion of an application is either a client, which requests services, or a server, which provides certain services. Between a client and a server, a software broker matches requests made by a client to services provided by a server. Both client and server can be written in a variety of programming languages and located on different platforms in a networked computing environment. To enable CORBA based communications, the boundaries between client and server must be specified in CORBA's Interface Definition Language (IDL). Such a specification can then be used by a compiler, in a CORBA toolkit, to generate stub and skeleton code with which to glue applications together. These stubs and skeletons provide communication functions and an Application Programming Interface (API) accessible from a high level language such as C or C++.

Manufacturing execution activity is a critical link in the chain of product development activities: product design, manufacturing planning, manufacturing execution, and shop-floor control. The design and planning data, such as design models, process plans, bill of materials, detailed production schedules, and quality plans need to be transferred to MES. Execution data, such machine control programs, job orders, and resource locations, need to be transferred to shop-floor controllers. Some collected data on the shop floor need to be fed back to MES. It is desirable to enable the interoperability of MES with its related applications. Based on the needs and the CORBA technology, it is possible to develop plug and play mechanisms for MES to be interoperated with its related manufacturing software systems.

There is a related architecture, the CIM Framework, developed by the Sematech consortium (Sematech, 1997). The CIM Framework has adopted the basic concept of managing manufacturing information using CORBA. The Framework is designed for sharing planning and operational information on semiconductor wafer production among different control and planning software modules. The Framework provides a set of basic entities and their relationships for *semiconductor* manufacturing shop-floor information management. Different from the CIM Framework, the paper describes a set of interfaces that enable general software interoperation in the mechanical product manufacturing domain.

4 FOUNDATION INTERFACES

Based on the CORBA methodology, a set of interfaces has been developed for the manufacturing planning and shop-floor execution information management and control for mechanical parts production. The goal of the interface design is to make the interfaces open, neutral, and extendable. The developed interfaces serve as a foundation on which more specific object interfaces can be derived and made interoperable according to CORBA. Users could extend the model by specializing, overriding, and extending some of the interfaces. The interfaces in this report are in the first version. Modification and enhancement of them will take place in the future as the technology and practices evolve.

4.1 Manufacturing Execution Activity Model

An activity model of product development has been developed in reference to a product realization model in (Barkmeyer, 1996). The top level activity is Develop Products (A0). This activity is decomposed into a series of six subactivies. They are in a diagram in Figure 1.

A1 – Engineer Product and Process

Design products and generate a manufacturing process plans and alternative plans. The product design engineering includes functional requirements, conceptual design, embodiment design, detailed design, design analysis, and the specification of bill of materials. The manufacturing process engineering includes process selection, operation planning, workpiece routing, and equipment/device control program generation. This activity provides design and processing information for the downstream resource planning and manufacturing execution.

A2 – Plan Enterprise Resources

Develop business plan and schedule to acquire necessary resources and produce products for the market. The enterprise resources include materials, finished parts, equipment, and labor skills. The enterprise resource planning function includes financial and order management, production and material planning, master production scheduling, capacity requirement planning, and job definition. It also includes business process planning and resource requirement specification.

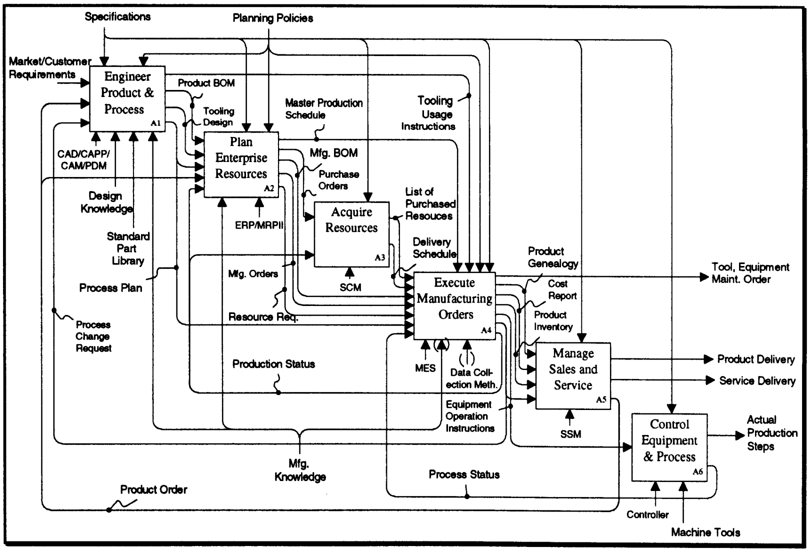

Figure 1 Product Development

A3 – Acquire Resources

Based on resource requirement plan, purchase resources from suppliers to meet the production schedule. This activity is supported by Supply Chain Management (SCM), which includes distribution, logistics, transportation management, and advanced planning.

A4 – Execute Manufacturing Orders

Based on the production plan and master schedule, carry out manufacturing orders in a production facility to produce finished goods. This activity includes initiate, manage, and report on production activities.

A5 – Manage Sales and Services

Manage sales of products and services to customers, including product delivery and receipts, product configuration, customer orders, quotes to customers, product returns, and post-sale service.

A6 – Control Equipment and Process

Using pre-programmed instructions, control and monitor equipment motions and processes in real time. This activities usually involves in distributed numerical control, programmable logic control, and factory-floor data collection.

The subactivities A4 covers the manufacturing execution functions and is further decomposed to set the context for the development and use of the foundation interfaces. The subactivities are in a diagram in Figure 2.

A41 – Develop Operation Sequence & Detailed Schedule

Based on the production plan and the master production schedule, define, sequence, and schedule operations locally on the levels of work cell, workstation, and machine in order to optimize productivity, such as minimize setup time, maximize throughput, minimize idle time, minimize queue time, and adjust shift pattern when new priority is in effect.

A42 – Dispatch Production Units

Determine which production unit in the queue is best processed next. The objective is to minimize the lead time and lateness.

A43 – Track Production Units and Resources

Provide the information on where any production unit is at all times and its disposition. Also provide the product genealogical information, such as who worked on it, current production information, component materials by supplier, lot number, serial number, any rework, measured data, or other exceptions related to the product. In the same time, provide the status information on specified resoürces, such as tools, devices, machines, and stock materials.

A44 – Manage Factory-floor Data/Document

Provide hardware/software interface links to obtain mission-critical data pertinent to production activities. Collect the data from the factory and analyze them for multiple purposes, such as product throughput, quality, delivery, and equipment maintenance. Manage documents, such as cost reports, maintenance orders, inventory reports, process change requests, manuals, specifications, company policies, etc. Control the data collection, access, and distribution. Provide version control of documents, such as part programs, operation instructions, manufacturing orders, detailed schedules, part drawings, engineering change notices, production unit records, records of communication from shift to shift, manuals, standards, company policies, safety regulations, etc.

The decomposition of the activity Engineer Process in A1, which is equivalent to process planning, is in the NISTIR 5808 (Feng, 1996a). Activities A2, A3, A5, and A6 are there to show their relations to A4. The Engineer Product in A1, A2, A3, A5, and A6 are out of the scope in the interfaces development. In addition to NISTIR 5808, a detail dimensional inspection process planning model for discrete part inspection in the product development cycle is in (Feng, 1996b). Furthermore, the activity model is documented using the IDEF0 (U.S. Air Force, 1981) methodology and techniques.

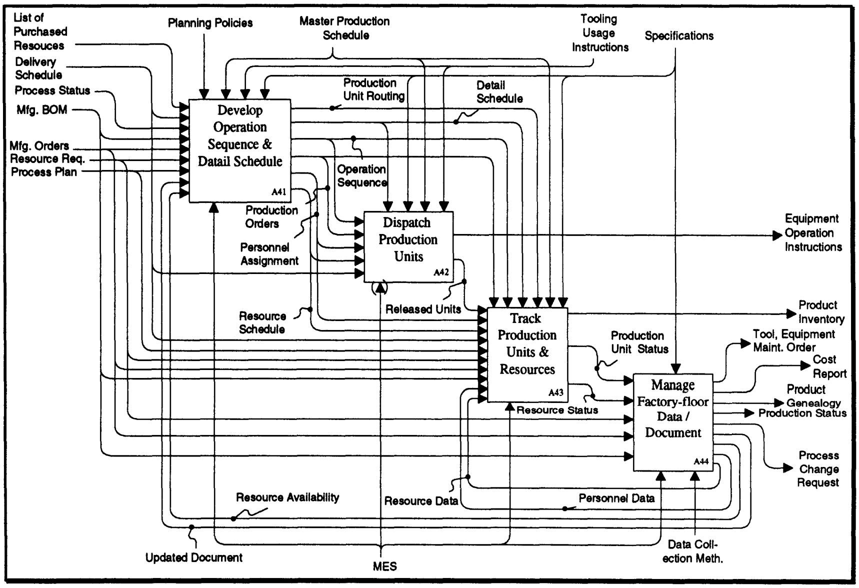

Figure 2 Manufacturing Execution

4.2 The Foundation Interfaces

The interfaces are specified using CORBA/IDL. Some of them are derived from the activities process planning and manufacturing execution and their subactivities, but not one-to-one mapping. There are two modules: the manufacturing process planning module and the manufacturing execution module. Each module contains data structure, interfaces, and type definitions. Samples of the IDL program for manufacturing execution and process planning are in Figures 3 and 4, respectively.

```
module Manufacturing_Execution
{
        ...
        interface JobOrderCreation
        {       void develop_job_order(inout Job_order a_job_order);
                void approve_the_order(inout Job_order a_job_order);
        };
        interface WorkSetupPlanning
        {       void create_primary_setup_insturctuions (out
                 Work_setup  setup);
                void create_alternate_setup_instructions(in Work_setup setup);
        };
        interface EquipmentTracking
        {       attribute Equipment a_piece_of_equipment;
                void track_equipment();
                void report_status(out string status);
        };
        interface MachineTracking
        {       attribute Machine a_machine;
                void track_machine (out string machine_status);
                void report_machine_usage(out string machine_usage_report);
        };
        interface ToolTracking
        {       attribute Tool a_tool;
                attribute string location;
                void track_tool(inout string tool_status);
                void report_status(out string status, out string tool_location);
        };
        interface FixtureTracking
        {       attribute Fixture a_fixture;
                void describe_function(inout Description funtions);
                void report_stuats (inout Description location,
                                inout Description status);
        };
        interface WorkpieceTracking
        {       void allocate(inout Workpiece a_workpiece);
                void workpiece_progress(inout Workpiece_status status);
                void workpiece_location_trace(out string the_location);
                void release(inout Workpiece a_workpiece);
        };
... }; // End of Module Manufactuirng _Execution
```

Figure 3 Sample Manufacturing Execution Interfaces

```
module Process_Planning
{       ...
        interface MaterialSelection
        {       attribute Cad_data cad_file;
                attribute Material_information selected_material;
                void select_material(in Cad_data cad_model, out
                 Material_information mi);
        };
        interface ManufacturingFeatureDerivation
        {       attribute Cad_data part_model;
                attribute Cad_data manufacturing_features;
                void derive_manufacturing_feature(in Cad_data pm,
                 out Cad_data mfs);
        };
        interface ManufacturingResourceSelection
        {       attribute Cad_data manufacturing_features;
                attribute Cad_data part_model;
                attribute Material_information material;
                attribute Manufacturing_Execution::Equipment
                manufacturing_equipment;
                void select_manufacturing_resources(in Cad_data mf,
                in Cad_data pm,
                inout Material_information material_stock,
                out Manufacturing_Execution::Equipment eqp);
        };
        interface OperationSelection
        {       attribute Manufacturing_Execution::Equipment
                 manufacturing_equipment;
                attribute Cad_data part_model;
                attribute Cad_data manufacturing_features;
                attribute Material_information material_stock;
                attribute Manufacturing_Execution::OperationSequence
                manufacturing_operations;
                void generate_operation(inout
                 Manufacturing_Execution::Equipment mr,
                inout Cad_data pm, inout Cad_data mf,  inout
                 Material_information stock,
                inout Manufacturing_Execution::OperationSequence
                 mops);
        };
        interface SetupSpecification
        {       attribute Manufacturing_Execution::OperationSequence
                 manufacturing_operations;
                attribute Cad_data part_model;
                attribute Cad_data manufacturing_features;          (continuing)
```

(continued)

```
                attribute Setup part_setup;
                attribute Setup fixuture_setup
                void specify_setups(in
                Manufacturing_Execution::OperationSequence
                 manufacturing_operation, in Cad_data, part_model, in
                  Cad_data mfs, out Setup part_setup,
                  out Setup fixture_setup);
        };
        interface OperationSequencing
        {       attribute Manufacturing_Execution::OperationSequence
                manufacturing_operation_sequence
                attribute Manufacturing_Execution::OperationSequence
                alternative_manufacturing_operation_sequences;
                void sequence_specification(in
                Manufacturing_Execution::OperationSequence
                manufacturing_operation_sequence, out
                 Manufacturing_Execution::OperationSequence
                 alternative_operation_sequences);
        };
        interface ProcessParametersDerivation
        {       attribute Material_information material_stock;
                attribute Manufacturing_Execution::OperationSequence
                manufacturing_operation_sequence;
                void process_parameter_definition(in
                 Material_information stock,
                 inout Manufacturing_Execution::OperationSequence
                 manufacturing_operation_sequence);
        };
...}; // end of module Process_Planning
```

Figure 4 Sample Process Planning Interfaces

5 CONCLUDING REMARKS

A set of foundation interfaces using CORBA/IDL for process planning and manufacturing execution software modules has been developed. The interfaces will enable software modules in various hardware/software platforms and in different locations to be able to exchange data and messages with each other. The interfaces will be updated according to the new technology and industrial practices in the future.

6 REFERENCES

Barkmeyer, E. (Editor) (1996) SIMA Reference Architecture Part 1: Activity Models , NISTIR 5939, the National Institute of Standards and Technology, Gaithersburg, MD 20899.

Feng, S. (1996a) A Machining Process Planning Activity Model for Systems Integration, NISTIR 5808, the National Institute of Standards and Technology, Gaithersburg, Maryland.

Feng, S. (1996b) A Dimensional Inspection Planning Activity Model, Journal of Engineering Design and Automation - special issue on Tolerance and Metrology for Precision Manufacturing, Vol. 2, No. 4, pp. 253-267.

MESA International (1997a) The Benefit of MES: A Report from the Field, White Paper 1, Manufacturing Execution Systems Association, Pittsburgh, PA 15215.

MESA International (1997b) MES Explained: A High Level Vision, White Paper 6, Manufacturing Execution Systems Association, Pittsburgh, PA.

OMG(Object Management Group) (1996) The Common Object Request Broker: Architecture and Specification manual, version published by the Object Management Group, Inc.

Sematech (1997) Computer Integrated Manufacturing (CIM) Framework Specification, version 1.5, Austin, Texas.

U.S. Air Force Wright Aeronautical Laboratories (1981) Integrated Computer-Aided Manufacturing (ICAM) Architecture Part II, Volume IV - Functional Modeling Manual (IDEF0), Material Laboratory