

Specifications of the OpenMES Framework

Version 1.0 (Draft alpha 2)

MSTC/JOP 1202 (2000 September 1)

Production System Modeling Technical Committee
Japan FA Open Systems Promotion Group
MANUFACTURING SCIENCE & TECHNOLOGY CENTER

Specifications of the OpenMES Framework

Version 1.0 (Draft alpha2) 2000, September 1

Version 1.0 (Draft alpha1) 1999, August 26

Edited by Production System Modeling Technical Committee
 Japan FA Open Systems Promotion Group

Published by Manufacturing Science & Technology Center, 1999

© JOP/MSTC 1999, 2000 – All rights reserved

Contents

1. SUMMARY.....	1
2. INTRODUCTION.....	2
2.1. INTENDED AUDIENCE	2
2.2. WHY FRAMEWORK?.....	3
2.3. WHY CORBA?.....	4
2.4. SCOPE OF FRAMEWORK.....	4
2.5. BENEFITS OF THE FRAMEWORK	5
2.5.1. <i>End user</i>	5
2.5.2. <i>System integrator</i>	5
2.5.3. <i>Component provider</i>	6
2.5.4. <i>Equipment manufacturer</i>	6
2.6. INTERFACE DEFINITION LANGUAGE.....	6
2.7. HOW TO USE THIS DOCUMENT.....	7
2.8. RELATED ACTIVITIES.....	7
2.8.1. <i>OMG</i>	7
2.8.2. <i>ISO</i>	8
2.8.3. <i>MSTC/JOP</i>	8
3. OPENMES OBJECTS	9
3.1. FACTORY MANAGEMENT FUNCTION GROUP	11
3.1.1. <i>Factory structure management component</i>	12
3.1.2. <i>Equipment schedule management component</i>	14
3.2. PRODUCTION ORDER MANAGEMENT FUNCTION GROUP	19
3.2.1. <i>Production order management component</i>	20
3.2.2. <i>Production lot generation management component</i>	23
3.3. PRODUCT SPECIFICATION MANAGEMENT FUNCTION GROUP	24
3.3.1. <i>Product specification management component</i>	24
3.4. PROCESS MANAGEMENT FUNCTION GROUP.....	29
3.4.1. <i>Production lot management component</i>	29

3.4.2.	<i>In-process job management component</i>	33
3.4.3.	<i>Order release management component</i>	36
3.4.4.	<i>Transfer order management component</i>	39
3.5.	MATERIAL MANAGEMENT FUNCTION GROUP	40
3.5.1.	<i>Material management component</i>	40
3.6.	TRANSFER MANAGEMENT FUNCTION GROUP	43
3.6.1.	<i>Transfer management component</i>	43
3.7.	PROCESS SPECIFICATION MANAGEMENT FUNCTION GROUP.....	47
3.7.1.	<i>Process route management component</i>	47
3.8.	EQUIPMENT MANAGEMENT FUNCTION GROUP.....	52
3.8.1.	<i>Equipment management component</i>	52
3.9.	SCHEDULE MANAGEMENT FUNCTION GROUP	67
3.9.1.	<i>Scheduler interface component</i>	67
3.10.	COMMON FUNCTION GROUP	68
3.10.1.	<i>Event Notification Management Component</i>	68
3.10.2.	<i>Remote entity component</i>	69
3.10.3.	<i>State management component</i>	70

Figure List

Figure 1 Class diagram related to the fundamental manufacturing execution.....	10
Figure 2 Collaboration diagram for the fundamental manufacture execution process	11
Figure 3 Class diagram for Factory Structure Management Component.....	12
Figure 4 Sequence diagram for the modification of equipment information	13
Figure 5 Sequence diagram for the registration of equipment information	14
Figure 6 Class diagram of Equipment Schedule Management Component.....	15
Figure 7 Sequence diagram for the creation of factory operation plan.....	17
Figure 8 Sequence diagram for the creation of equipment operation plan	18
Figure 9 Sequence Diagram of the modification of factory operation plan.....	19
Figure 10 Class diagram of Production Order Management Component.....	20
Figure 11 State diagram of MesProductionOrder	22
Figure 12 Class diagram of Product Specification Management Function Group	24
Figure 13 Sequence diagram for the increment/decrement of product specification in-use count	26
Figure 14 Sequence diagram for the registration of product specification.....	27
Figure 15 Sequence diagram for the modification of product specification.....	28
Figure 16 Class diagram of Production Lot Management Component.....	29
Figure 17 State diagram of MesLotJob	32
Figure 18 Class diagram of Process Job Management Component	33
Figure 19 State diagram of MesProcessJob.....	33
Figure 20 Class diagram of Order Release Management Component	35
Figure 21 Sequence diagram for the release of lot jobs.....	37
Figure 22 Sequence diagram for the notification of order release	38
Figure 23 Class diagram of Transfer Order Management Component	39
Figure 24 Class diagram of Material Management Function Group	40
Figure 25 Class diagram of Transfer Management Function Group	43
Figure 26 Sequence diagram for the lot transfer from storage to equipment.....	45
Figure 27 Sequence diagram for the lot transfer from equipment to storage.....	46
Figure 28 Class diagram of Process Specification Management Function Group	47
Figure 29 Sequence diagram for the registration of process route.....	49

Figure 30 Sequence diagram for the registration of process resource	50
Figure 31 Sequence diagram for the registration of process capacity	51
Figure 32 Class diagram related to MesProcessEquipment in Equipment Management Component	53
Figure 33 Class diagram related to MesTransferEquipment in Equipment Management Component	54
Figure 34 State diagram of MesEquipment	55
Figure 35 State diagram of MesProcessEquipment	59
Figure 36 Sequence diagram for the automatic execution of process equipment	63
Figure 37 Sequence diagram for the semi-automatic execution of process equipment.....	64
Figure 38 Sequence diagram for the suspend/resume of process equipment in automatic execution..	65
Figure 39 Sequence diagram for the suspend/resume of process equipment in semi-automatic execution	66
Figure 40 Class diagram of Schedule Management Function Group.....	67
Figure 41 Class diagram of Event Notification Management Component	68
Figure 42 Class diagram of Remote Entity Component	69
Figure 43 Class diagram of State Component	70
Figure 44 State diagram of a state representation.....	71

1. Summary

This document describes the specifications of OpenMES framework, which is a Manufacturing Execution System (MES) application framework came out from a project conducted under the Manufacturing Science and Technology Center in Japan. The OpenMES framework is a model of manufacturing execution dedicated for discrete production process. This document is prepared for those who are going to implement an MES application framework, and develop software components and applications for MES. Currently, there is no agreed-upon manufacturing model for the discrete process. Therefore, the development of MES has been conducted individually without relying on a common model of manufacturing, which facilitate sharing of production knowledge in practice. However, if reusable software components are prepared on the basis of a common manufacturing model, the development of MES can be realized by combining those software components, and adapted to individual, proprietary manufacturing environments with moderate customization of the components.

Benefits of the OpenMES framework are summarized below.

- Company-wide optimization can be achieved through coordination with Supply Chain Management (SCM) and Enterprise Resource Planning (ERP). Since the field information is reflected in SCM/ERP, optimization can be pursued taking account of activities for procurement, production, distribution, and sales.
- Production data can be collected for the improvement of manufacturing activities. In particular, the data needed for facilitating Plan-Do-Check-Action cycles can be provided and exploited.
- The use of Web browsers allows end users to access production order information and production result information no matter where they are. Accurate decision- making is then facilitated by the ubiquitous data access.
- A multi-vendor environment can easily be established for production equipment.
- A high-quality MES can be established within a shorter period of time relying existing, qualified software components.
- Qualified software components facilitate sharing competitive manufacturing knowledge.

The OpenMES framework covers the following units of manufacturing execution systems:

- Factory management
- Production order management
- Product specification management
- Process specification management
- Process management

- Equipment management
- Transfer management
- Material management
- Schedule management

The OpenMES framework intends to facilitate not only the development of application software and its Graphical User Interfaces (GUIs), but also the customization of data management capability and CORBA interface for interoperability.

2. Introduction

The Information Technology Promotion Agency of Japan has commissioned the Manufacturing Science and Technology Center to develop an application framework for MES as a critical portion of the Electronic Commerce Common Infrastructure Establishment Project. This document describes the specifications of the OpenMES framework came out from the government project in Japan.

In general, the purpose of defining system specifications may include:

- Standardization of the specifications,
- Facilitation of individual system development based on software defined by a project, or
- Deployment of a leading-edge technology came out of the project into practice

The above objectives may not necessarily be achieved all at once by means of a single specification document. A document for standardization does not cover the details needed for the implementation. The role of this document is to give a conceptual understanding of the OpenMES framework for those who are going to apply it to individual application development projects. The document for standardization will be subjected to future discussions in the Production System Model Special Committee of the FA Open Promotion Council at the Manufacturing Science and Technology Center.

2.1. Intended Audience

This document is prepared for the following three types of readers.

- Application programmers
- Component providers
- Framework providers

Application programmers are those who develop MES application software customizing an

implementation of the OpenMES framework. Component providers are those who actually implement an MES application framework by referring to the specifications. Framework providers are those who implement or extend frameworks by referring to the specifications.

2.2. Why Framework?

MES is a system to control and manage production process in a factory. MES receives a production plan and production orders from ERP, issues work orders to production equipment, collects actual results of production, and exercises progress management. There are problems with MES configuration and procurement.

- A production system consists of several subsystems. However, they cannot easily be integrated into a single organic whole because no standardized interface exists.
- A production system is usually maintained in a multi-vendor environment, which consists of a wide variety of manufacturing equipment. However, there is no standardized interface to be exploited for the management of manufacturing equipment. Therefore, it is important to define and provide a set of APIs that allow the equipment management in a multi-vendor environment.
- Various individual requirements exist due to a variety of production needs in each manufacturing site. It is therefore difficult to make use of prepackaged software as it is, without customization to adapt individual requirements in each manufacturing site.

Under these circumstances, production system development and maintenance costs a lot. In addition, existing production system is not fully integrated in practice. It is therefore not easy to achieve production optimization for the entire system.

However, we believe that a common model for MES exists. When we prepare software components in accordance with such a model, we believe we can create MES configurations suitable for individual production sites by combining necessary software components and customizing them. To make such an approach successful, we use an object-oriented technology. Object orientation makes it possible to make a model of a production system and disassemble it into software components. Meanwhile, the linkage between these software components can be defined to prepare a template for the production system to be configured. This template is called an MES application framework. It makes it easy to accomplish customization by, for instance, providing additional functional units or modifying the behavior of existing functional units.

In addition to functionality, the MES application framework must also meet the following requirements.

- The MES application framework must be scalable enough to cover small- and large-scale production lines.
- The MES application framework must be open and not dependent on hardware or platform (OS, etc.).

To provide adequate scalability and open connectivity, CORBA distributed objects are used and distributed to a number of servers. In addition, Java is employed as an implementation language.

When the above MES application framework is introduced, a field information system is established. As a result, a system that allows users to access data easily from any place can be configured at a lower cost and within a shorter period of development time. When this information-intensive system is enhanced, an optimum production system can be established in coordination with ERP, SCM, and equipment control system.

2.3. Why CORBA?

MES consists of a number of equipment units, server machine, and client terminals. It is intrinsically requested to perform distributed processing. Thanks to distributed processing, it is scalable enough to support small-scale production systems to large-scale production systems containing a number of factories. Further, it is requested to permit the use of software components or plug-ins from perspectives of software extensibility and connectivity with manufacturing equipment. Recently, the following platforms are highlighted as they can offer the above environment.

- DCOM
- CORBA

The hardware of MES consists of a variety of computers, including PCs, workstations, business computers, and large-scale computers. Platform-independent CORBA is suitable for configuring a seamless system in the above environment.

2.4. Scope of Framework

There exist three types of production processes to be dealt with MES.

- Continuous process
- Batch process
- Discrete process

Among these types, OpenMES is primarily developed for the discrete process. In a discrete process, the work sorted by a lot or other unit is processed while being transported between manufacturing equipment sets, which are grouped for individual processing operations. It correlates to a machining or assembling factory. The discrete process can be further sub-divided into a job shop and a flow shop. A flow shop has a fixed processing route such as a conveyor line, while a job shop has a flexible processing route. The OpenMES framework provides a model of manufacturing execution for the both kinds of discrete production.

The framework contains the following functions, which are further elaborated in the remainder of this document.

- Factory management

- Production order management
- Production specification management
- Process specification management
- Process management
- Equipment management
- Transfer management
- Material management
- Schedule management

2.5. Benefits of the Framework

2.5.1. End user

End users receive the following benefits from the employment of an MES framework.

- The use of a Web browser enables you to access production order information and production result information no matter where you are. Decision-making is then expedited and rendered accurate. For example, sales personnel can reply to inquiries about a delivery date by acquiring the information about stock and work in process. Further, field-managing personnel can reduce the degree of load concentration and decrease the amount of stock by obtaining the information about the production schedule for the next term.
- Company-wide optimization can be accomplished through coordination with SCM/ERP. Since the field information is reflected in SCM/ERP, optimization can be achieved covering sales, production, procurement, and physical distribution activities.
- Data necessary for Plan-Do-Check-Action can be obtained for production line improvement. The term "Plan" represents scheduling data; "Do", work orders; "Check", progress management; and "Action", result data. Data required for various phases can be generated and processed. Further, the necessity for preparing daily operation reports and the like is eliminated.
- Thanks to orders from the MES, two or more factories can function as a single factory. Various factory combinations can be used to match the product type.
- The OpenMES framework allows users to configure a system by using existing software components.

2.5.2. System integrator

System integrators receive the following benefits from the employment of an MES framework.

- A multi-vendor environment for production equipment can easily be established.
- A high-quality MES can be realized within a short period of time by using qualified software

components.

- User interfaces realized on top of Web browsers can be used to propose and offer advanced solutions.

2.5.3. Component provider

Component providers receive the following benefit from the employment of an MES framework. Business can be conducted by offering components into which the provider's original know-how is incorporated.

2.5.4. Equipment manufacturer

Equipment manufacturers receive the following benefit from the employment of an MES framework.

- Equipment no longer needs to be added upon each production system change.

2.6. Interface Definition Language

CORBA-based specifications of application frameworks are usually written in CORBA Interface Definition Language (IDL). However, the OpenMES project adopted Java programming language as an IDL for CORBA objects from the following two reasons.

- Development can be conducted even if application programmers are not familiar with CORBA IDL. The programmers then need to understand only the implementation language of OpenMES, namely, Java.
- Taking account of a performance issue stems from the remote object access/invoke, only the objects that require the remote interoperability are given as CORBA objects. Objects that cannot be accessed remotely, namely, non-CORBA objects are passed by value among CORBA objects. The pass-by-value is a function that is not included in the specification of CORBA 2.0, but provided with some of the CORBA ORG implementation such as VisiBroker.

Constructs in the OpenMES framework are organized hierarchically into three levels: function group, component, and object. The function-group level corresponds to a module of a cohesive sub-unit of MES applications, and implemented as a Java package. The component level is a collection of related objects in a function group. The object level is for a fundamental unit of abstraction, and implemented as a Java class. This document primarily deals with these three levels. In particular, Unified Modeling Language (UML) is used as a graphical notation of the structural and behavioral aspects of the OpenMES framework. Further details of the object internals such as method signature and instance variables are explained in the Javadoc of the OpenMES framework to be given as a separate document.

2.7. How to Use This Document

Customization with the OpenMES framework consists of the following five aspects.

- **GUI development**
With Visual Age Java, JBuilder, Visual Cafe or other development environment or JDK, develop or modify a graphical user interface so as to call a framework API.
- **Application development**
On either the client side or the server side, implement an application logic that calls a framework API. If HTML exchanges with clients are made using a servlet or JSP in situations where the logic is implemented on the server side, the associated design and implementation processes must also be completed.
- **Data customization**
Since classes are offered on the presumption that data customization is effected variously for all MES, add member variables and methods as needed.
- **CORBA interface customization**
When adding a member variable or method to an equipment interface, scheduler interface, or other CORBA interface, define an interface derived from a CORBA interface and implement the defined interface. It is then necessary to develop an application that calls a new method. When adding a variable to a CORBA class, define a class derived from an implemented class and implement the defined class. When you change the logic without applying CORBA interface changes, define a class derived from an implemented class and implement a method.
- **Framework function addition**
In consideration of reuse, add a function group and components of MES application framework by calling a class definition prescribed by the specifications. If distributed processing is required or desirable, newly define a CORBA interface and implement it.

2.8. Related Activities

2.8.1. OMG

A Request for Information (RFI) was issued in December 1997 from the MEC/MC working group of OMG's Manufacturing Domain Task Force. The RFI invited input from individuals and organizations with insight or information in any of areas related to MES, and six responses were submitted. The MES/MC working group has reactivated its planning for MES interfaces development. A new draft MES Roadmap document has been developed which reflects discussions at the Mesa OMG meeting in January 2000 as well as past work done by the MES/MC and ERP working groups.

2.8.2. ISO

The New Work Item is now being discussed by TC 184/SC5/WG1 (Enterprise Integration). The Japan Domestic Countermeasure Committee is studying the OpenMES framework.

2.8.3. MSTC/JOP

The Production System Model Special Committee is discussing specifications.

3. OpenMES Objects

The OpenMES framework consists of the following ten function groups.

- Factory management
- Production order management
- Product specification
- Process specification
- Process management
- Equipment management
- Transfer management
- Material management
- Schedule management
- Common function group

This section describes the classes and interfaces given in the above function groups.

Figure 1 is an overview of the fundamental manufacturing execution processes in the OpenMES framework. This is a class diagram, and shows the relationship among the production order class (`MesProductionOrder`), production lot class (`MesProductionLot`), lot job class (`MesLotJob`), process job class (`MesProcessJob`), work order class (`MesWorkOrder`), and process equipment class (`MesProcessEquipment`).

A `MesProductionOrder` object, which represents production orders, consists of `MesProductionLot` objects correspond to production lots. A `MesProductionOrderManager` object manages `MesProductionOrder` objects. A `MesLotJob` object represents an operation related to a `MesProductionLot` object and consists of `MesProcessJob` objects, which correspond to individual manufacturing processes such as processing, inspection, and cleaning. `MesWorkOrder` is associated with a `MesLotJob` object and set to process equipment `MesProcessEquipment` object.

Figure 2 is a diagram that illustrates collaboration among object instances in the course of the manufacturing execution. An application (AP production order management in the figure) calls the `addProductionOrder` method of `MesProductionOrderManager` to generate a production order (a `MesProductionOrder` object). The `createProductionLots` method is then called to generate a `MesProductionLot` object, and when the `releaseProductionOrders` method is called, `MesLotJobManager` generates `MesLotJob` objects. When an application (AP production lot management in the figure) calls the `createSchedule` method, a schedule is created by the `MesScheduler` object. When the `outputSchedule` method is called, the created schedule is registered to the `MesLotJobManager` object. The `releaseLotJobs` method releases a lot job into the production line. In accordance with the created

schedule, the addWorkOrder method assigns a MesWorkOrder object to a MesProcessEquipment object. When the work is finished, the finishWorkOrder method is called. Finally, the setWorkResult method associates the related MesWorkOrder, MesLotJob, MesProcessJob, and MesProductionOrder objects with the work result.

Class Diagram / Production

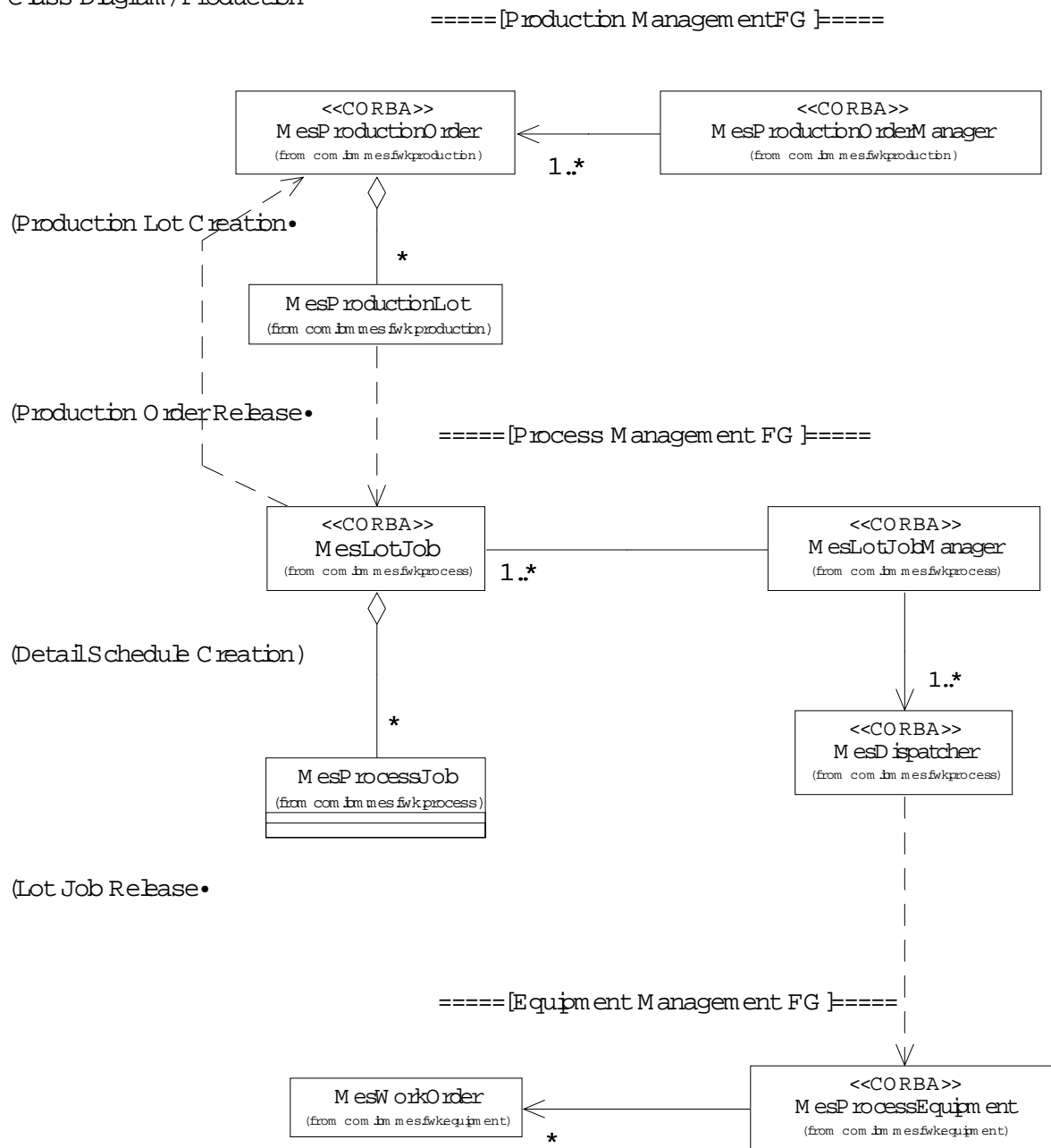


Figure 1 Class diagram related to the fundamental manufacturing execution

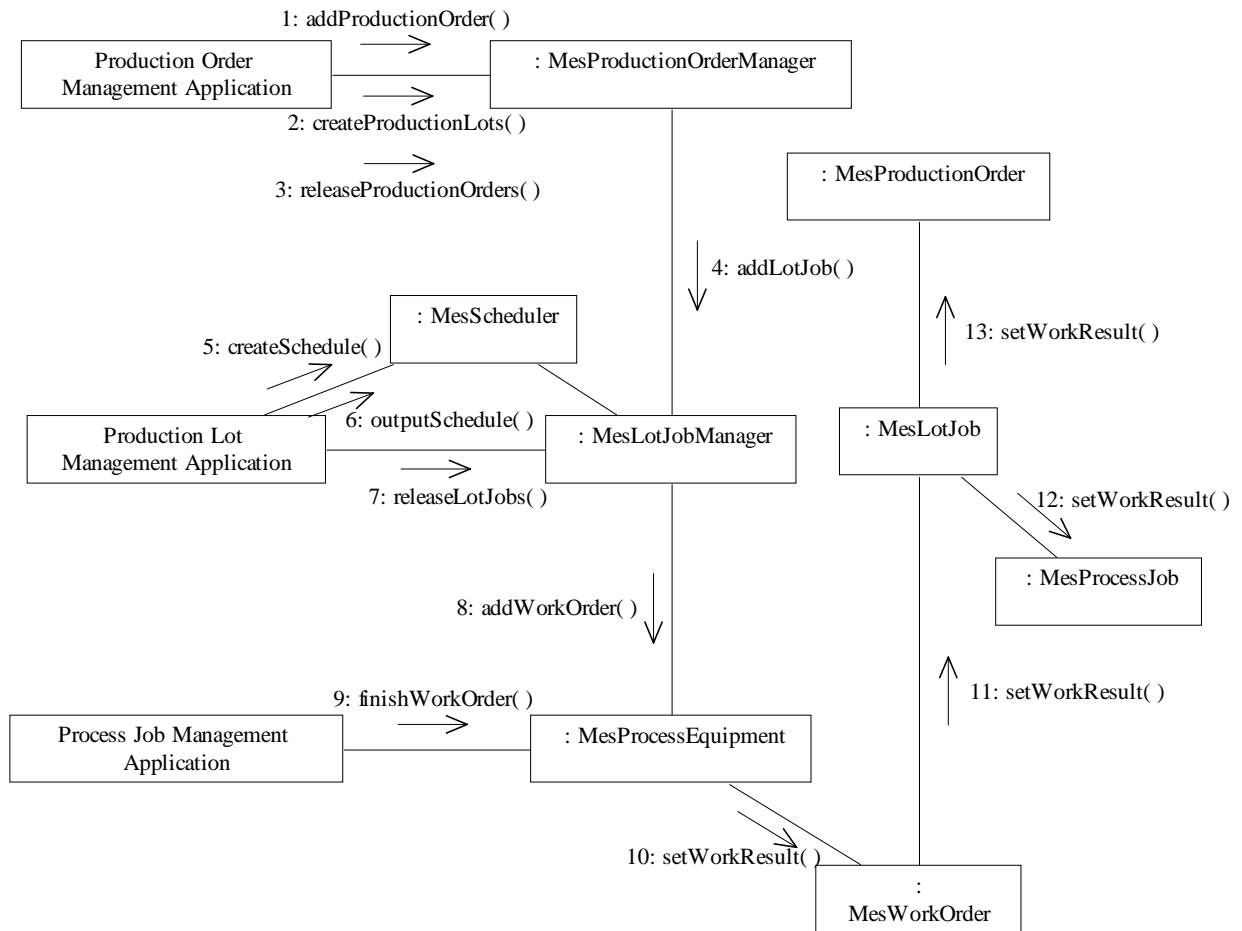


Figure 2 Collaboration diagram for the fundamental manufacture execution process

In order to realize actual manufacturing operations in a distributed environment, each class needs to provide remotely accessible CORBA interfaces. When a CORBA interface name is Foo, an implemented class is named FooImpl and its utility class for object start/management is named FooServer. FooImpl provides a set of methods that are defined for Foo. Therefore, FooImpl is not described in this document, although it may appear in some class and sequence diagrams for clarity.

3.1. Factory Management Function Group

This function group manages the factory structure and operating schedule. It consists of the following components.

- Factory structure management component
- Equipment schedule management component

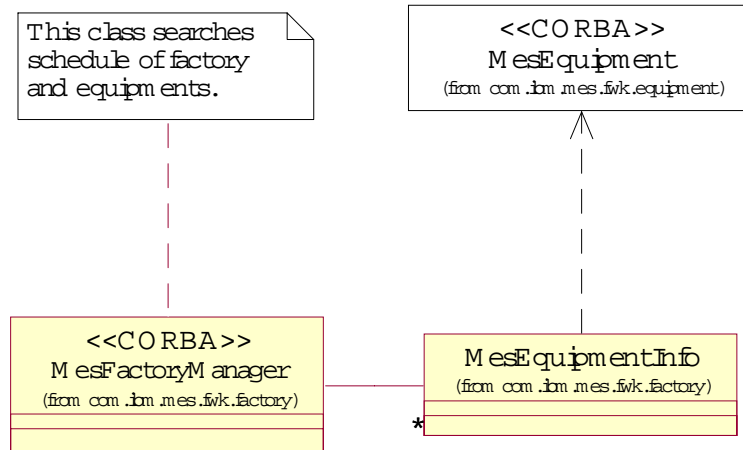


Figure 3 Class diagram for Factory Structure Management Component

3.1.1. Factory structure management component

This component manages information related to equipment installed in a factory.

(1) MesFactoryManager

This class defines manipulations (setup, search, modification, and deletion) to be performed on the manufacturing equipment information, factory operation plans, equipment operation plans, and maintenance programs. It also defines the method of acquiring available equipment during an equipment inactivity period of time or other specific period of time.

(2) MesEquipmentInfo

This class manages the information about manufacturing equipment, that is, the type, model number, and installation site values. It is presumed that customization will be effected on an individual MES basis.

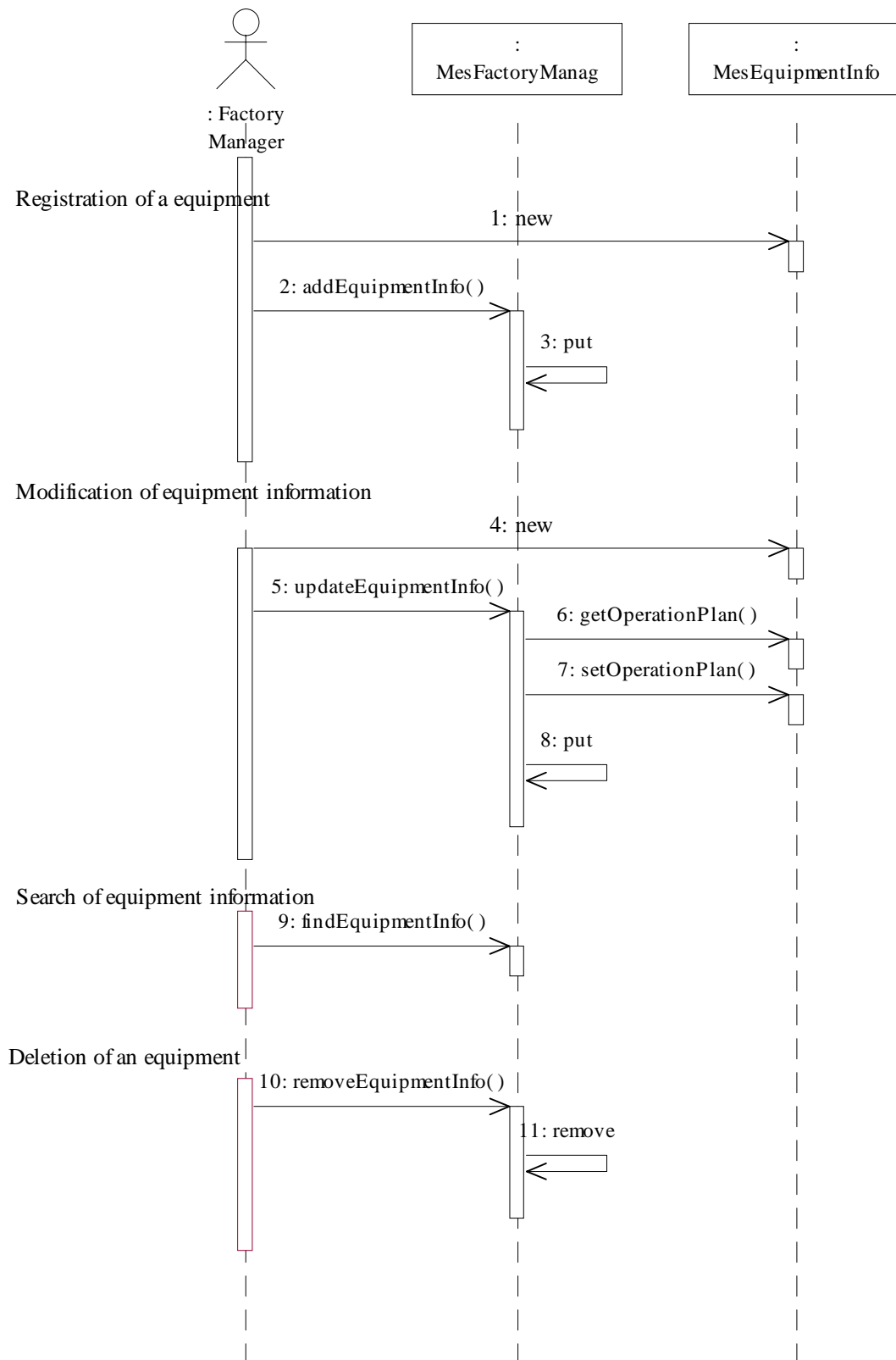


Figure 4 Sequence diagram for the modification of equipment information

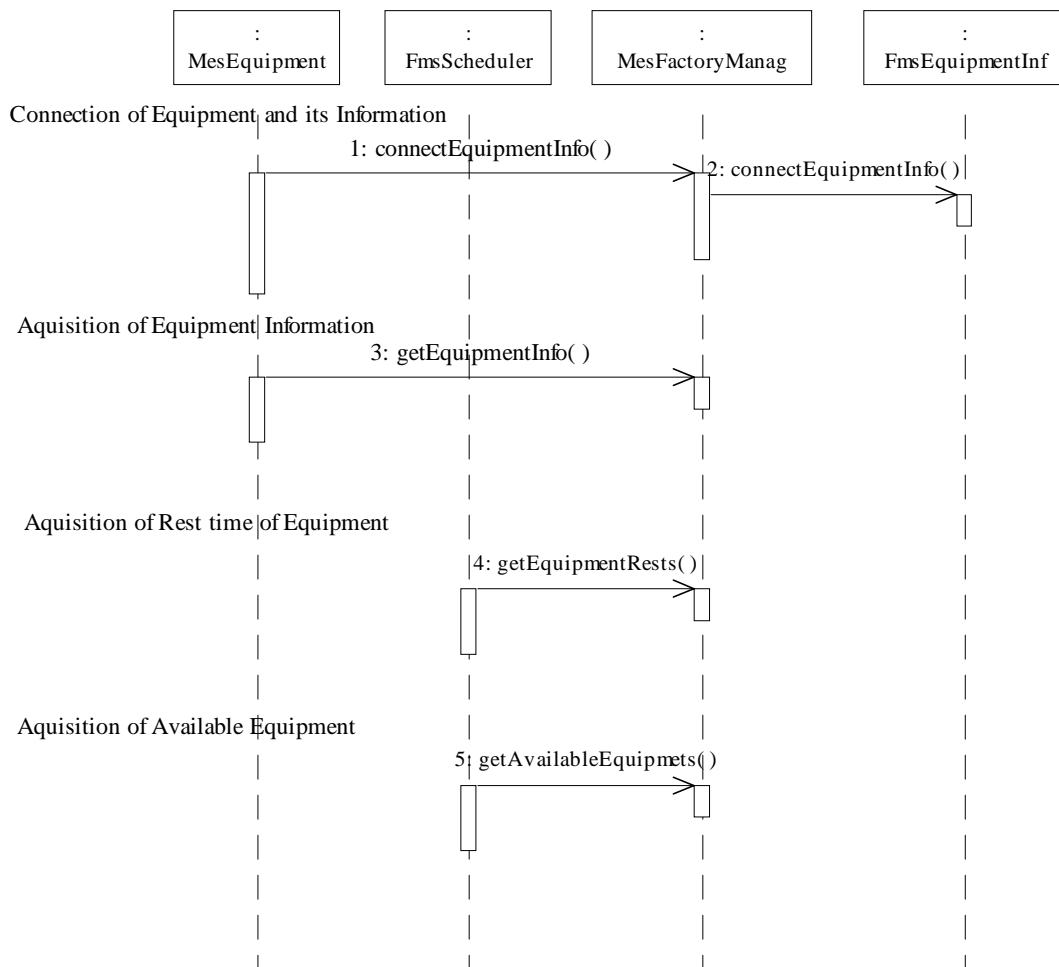


Figure 5 Sequence diagram for the registration of equipment information

3.1.2. Equipment schedule management component

This component consists of classes that offer functions for managing the schedules of a factory and individual equipment. The equipment maintenance periods are also described using this component. It also manages the result values concerning schedules.

A schedule is defined as an aggregate of inactivity periods. A factory schedule represents factory inactivity periods. An equipment schedule represents equipment inactivity periods. A maintenance schedule represents maintenance periods. An operating period recognized by the scheduler corresponds to the complementary set of the OR of factory and equipment inactivity periods.

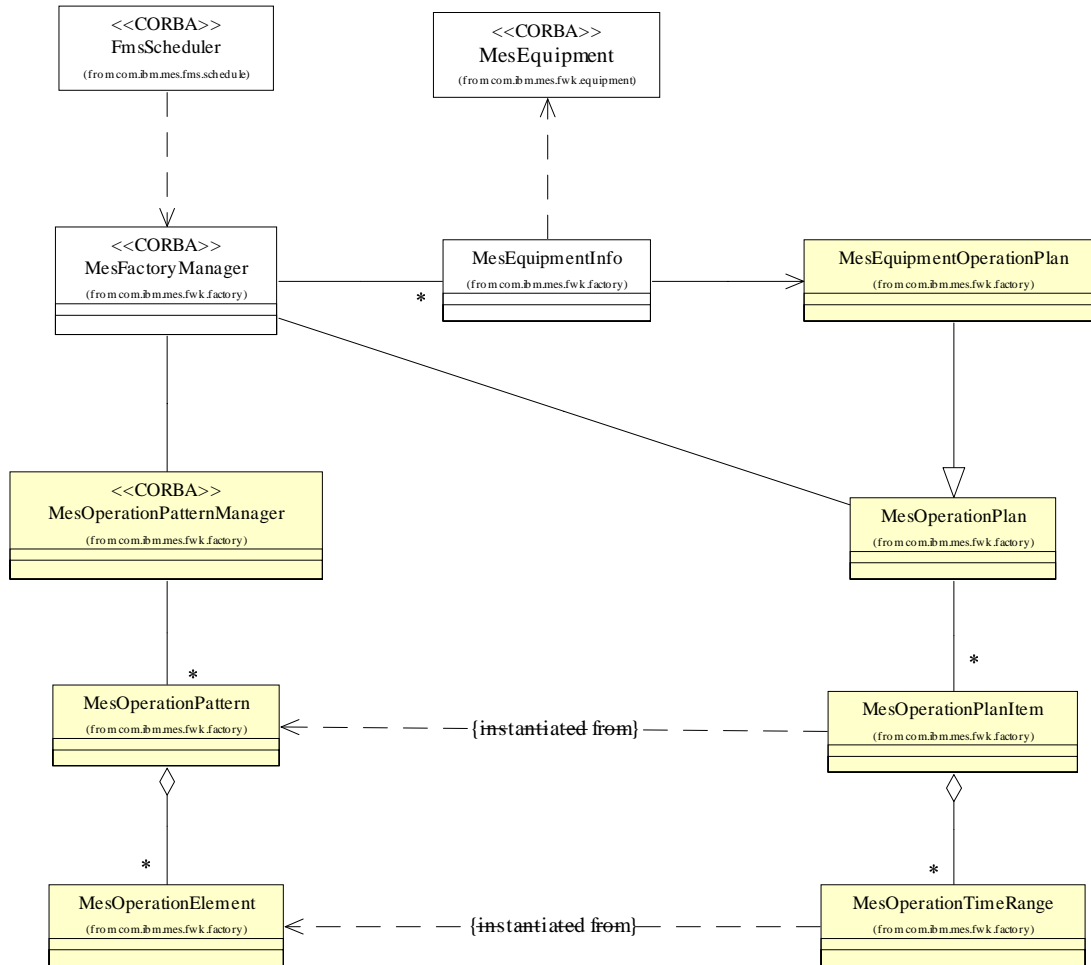


Figure 6 Class diagram of Equipment Schedule Management Component

(1) MesOperationPatternManager

This class defines the methods for manipulations (setup, acquisition, search, modification, and deletion) to be performed on manufacturing equipment management and factory operation plan patterns.

(2) MesOperationPattern

This class is an aggregate of MesOperationElement objects. The schedule pattern is to be stipulated by specifying the Interval between MesOperationElement objects. The Interval setting is variable in 1-minute increments. To employ a setting of 1-week intervals, use the value 10080 (=7*24*60).

(3) MesOperationElement

This class represents a time pattern element of a schedule. It is to be defined by specifying the offset from a start time and the duration of time. These settings are both variable in 1-minute increments.

(4) MesOperationPlan

This class offers manipulations (setup, acquisition, and deletion) to be performed on schedules. It retains an aggregate of MesOperatingPlanItem objects having differing patterns.

(5) MesEquipmentOperationPlan

This class has a plan for equipment schedule and maintenance management.

(6) MesOperationPlanItem

This class is a schedule element that is generated by making reference to MesOperationPattern. It specifies the start date/time and the number of repetitions. It retains an aggregate of MesOperationTimeRanges.

(7) MesOperationTimeRange

This class represents a time range. It manages schedules and results. It is defined by specifying the start date/time and period. The period setting is variable in 1-minute increments.

(8) MesPlanTimeRange

This class defines the time range. The inactivity start time and inactivity period are to be specified for definition. This class is returned as a hash table element as a result of inactivity period calculations.

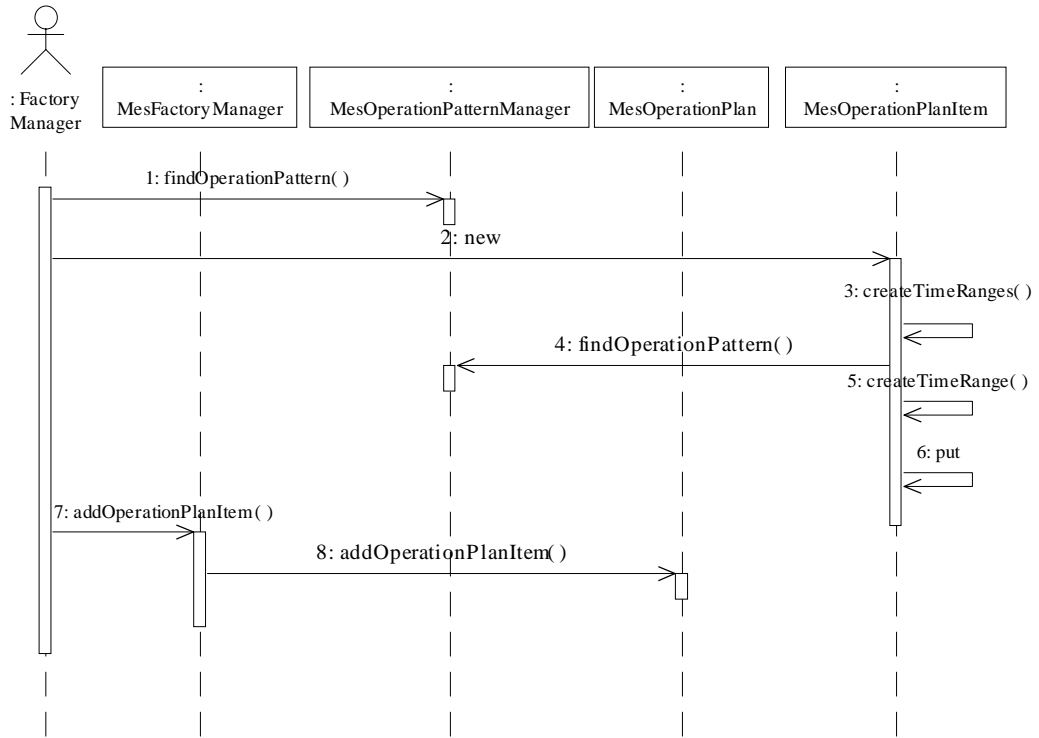


Figure 7 Sequence diagram for the creation of factory operation plan

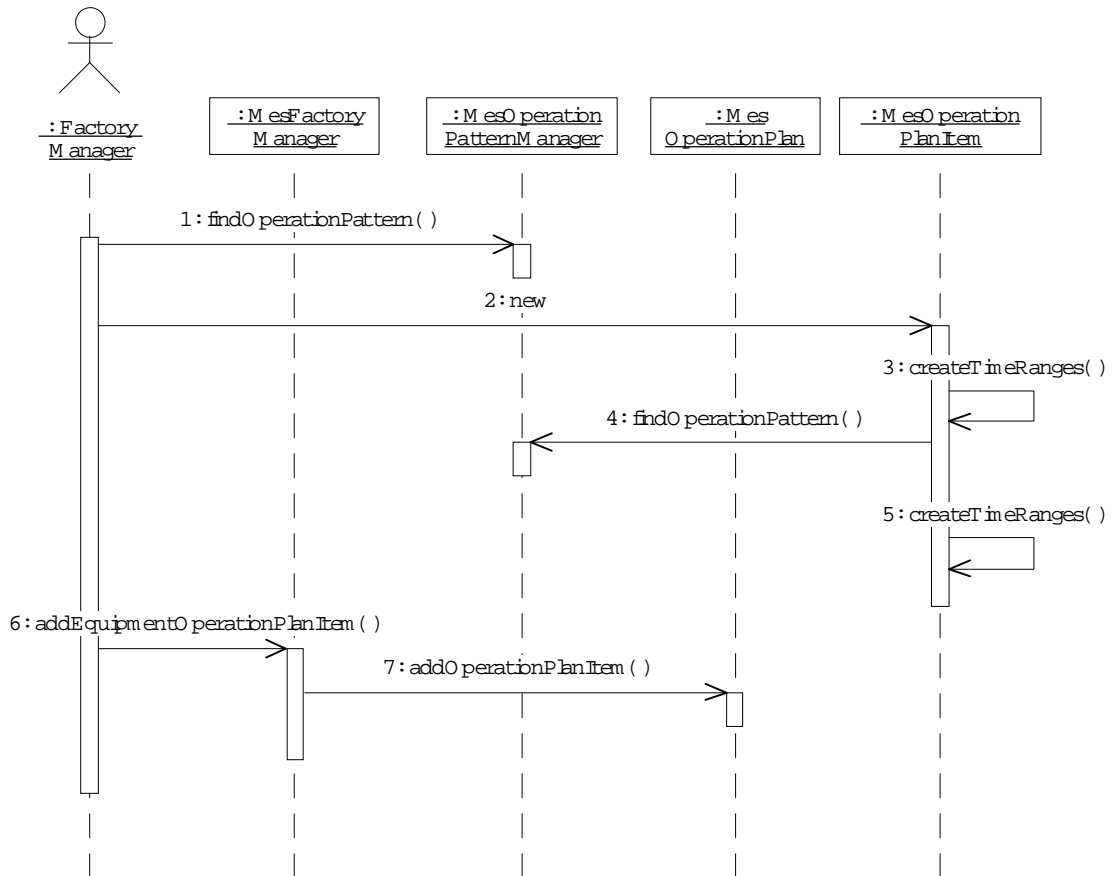


Figure 8 Sequence diagram for the creation of equipment operation plan

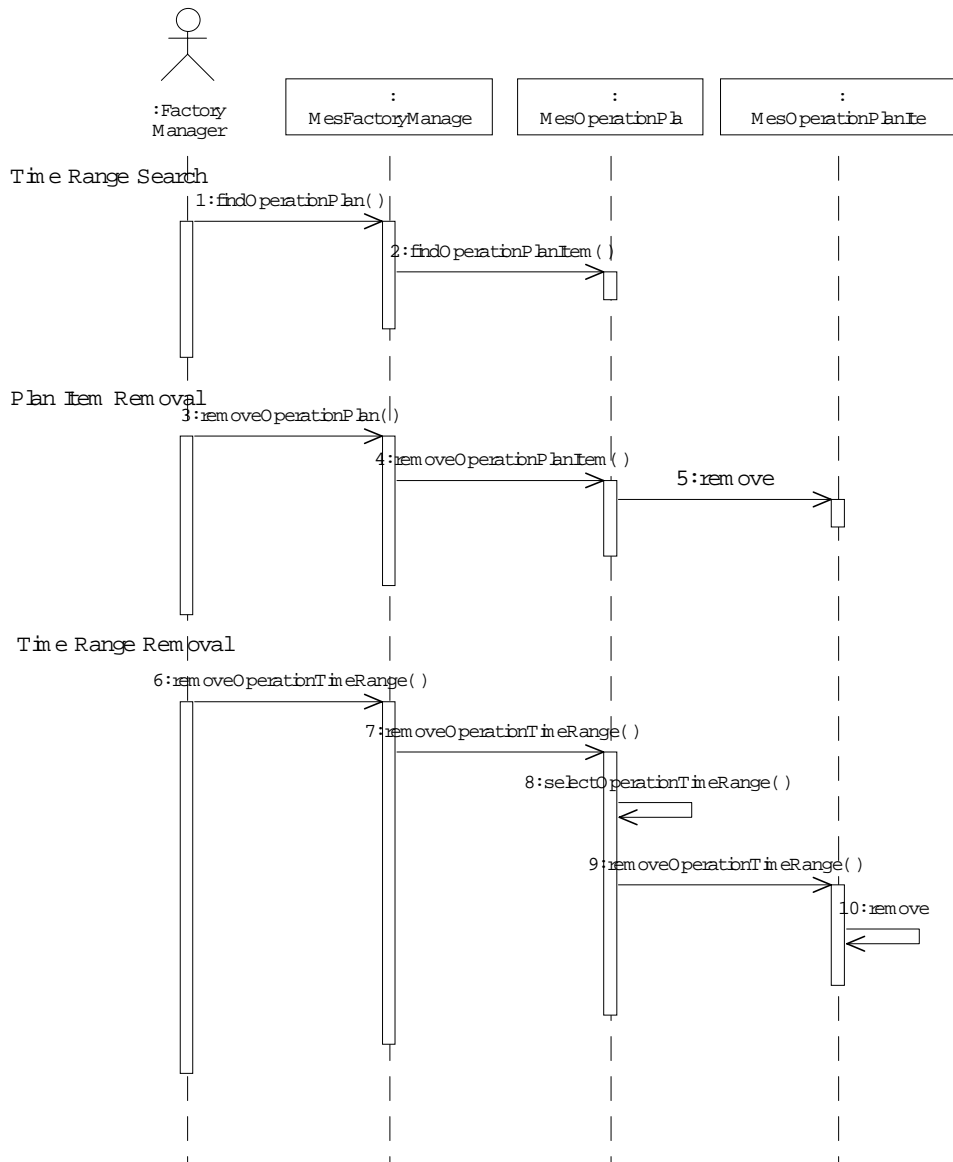


Figure 9 Sequence Diagram of the modification of factory operation plan

3.2. Production Order Management Function Group

The production order management function group offers an interface with a production planning system. Upon receipt of production orders, it can create a required number of lots, introduce the work into processes, and monitor the progress of production order executions. In other words, it manages the relationships between production orders and lots flowing in processes.

The relationship may vary with various conditions such as the production style (production on orders or speculative production), management system (MRP or serial number management), and lot segmentation.

3.2.1. Production order management component

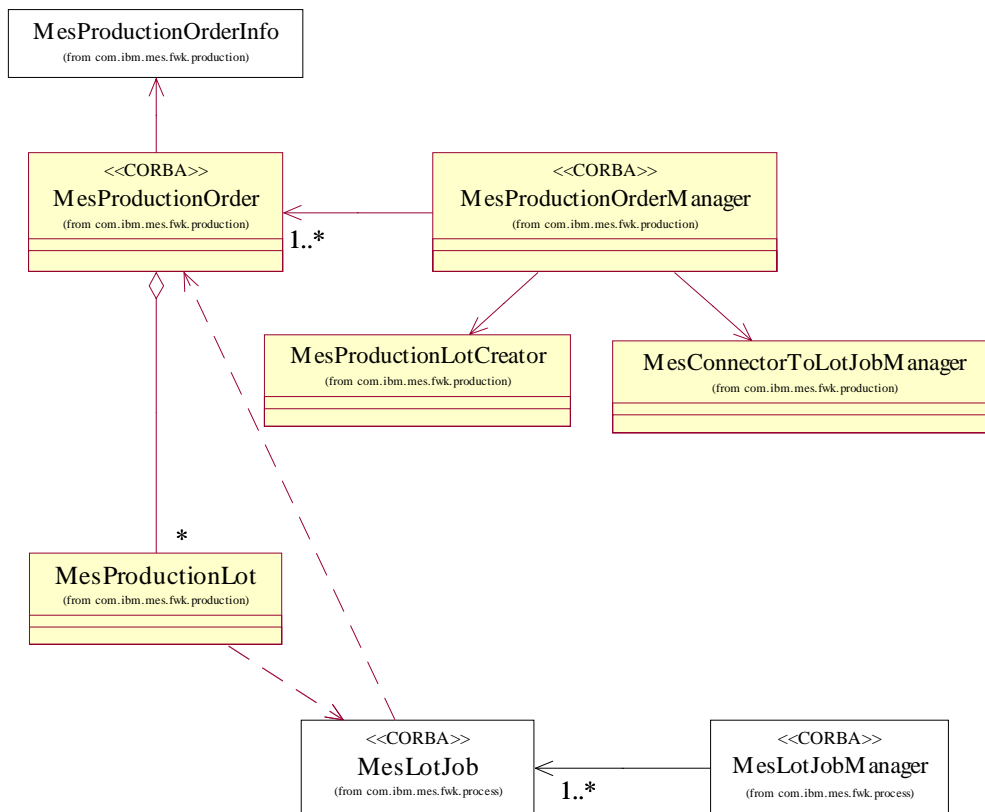


Figure 10 Class diagram of Production Order Management Component

(1) MesProductionOrderManager

This interface defines the methods for exercising MesProductionOrder management (registration, deletion, search, and modification).

(2) MesProductionOrder

This class corresponds to production orders, and provides the following functions.

- Get a production order ID
- Set or get a product ID
- Get a product specification ID
- Set or get a product specification version

- Register or get production lots
- Get the production lot count
- Select a production lot
- Set or get the production order priority
- Set or get a production order state
- Set or get production order information
- Get the current production yield
- Set or get a planned production volume
- Set or get a manufacturing yield
- Set or get the planned start date/time
- Set or get the actual start date/time
- Set or get the earliest production order issuance time
- Set or get the latest production order completion time
- Set or get the planned finish date/time
- Set or get the actual finish date/time
- Set or get the time of production order registration
- Get the aggregate of production order states
- Check whether the actual finish date/time is set for all production lots
- Check whether the actual start date/time is set for all production lots
- Delete all production lots
- Delete specific production lots
- Suspend production orders
- Release suspended production orders
- Set the delivery date for a production lot
- Set the planned start date/time for a production lot
- Set the actual start date/time for a production lot
- Set the planned finish date/time for a production lot
- Set the actual finish date/time for a production lot
- Set the actual yield for a production lot

(3) MesProductionOrderState

This class corresponds to a collection of production order states, and consists of the following states.

- Unprocessed
- Production lot generated
- Production order issued
- Detail schedule generated
- Production order issued
- Lot job started
- Production order completed
- Production order suspended
- Canceled

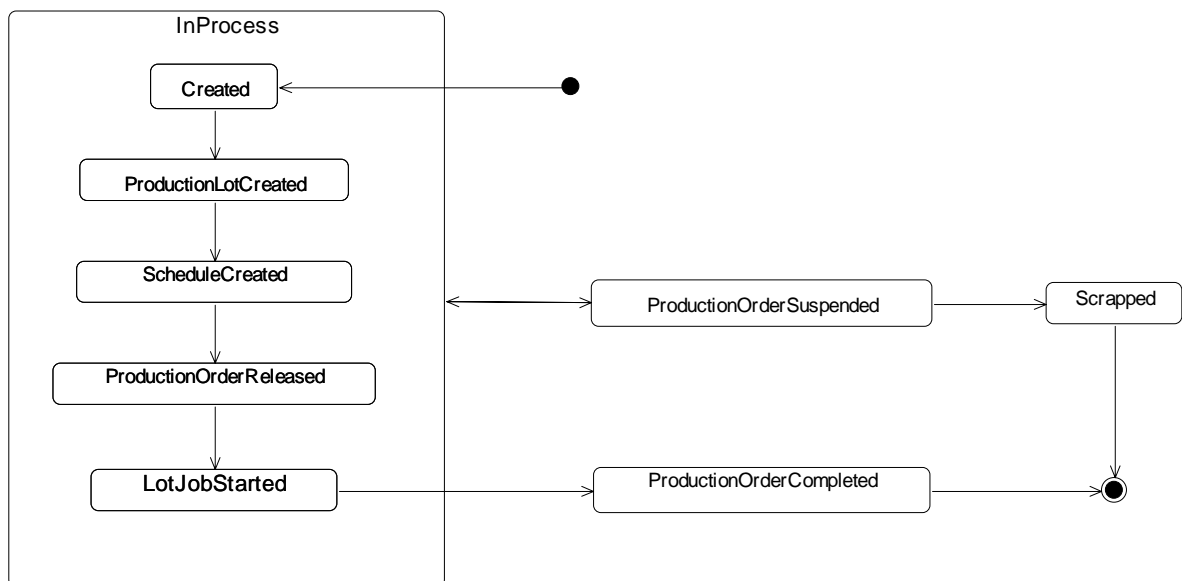


Figure 11 State diagram of MesProductionOrder

(4) MesProductionOrderInfo

This class represents production order information. This class is derived in applicable cases and used to define individual attributes and relevant operations that cannot be offered by MesProductionOrder.

3.2.2. Production lot generation management component

(1) MesProductionLot

This class represents a production lot. The defined methods have the following functions.

- Set or get a production lot ID
- Set or get a production order ID
- Set or get a delivery date
- Check whether a given production lot coincides with this production lot
- Set or get a planned volume
- Set or get a yield
- Set or get the planned start date/time
- Set or get the actual start date/time
- Set or get the planned finish date/time
- Set or get the actual finish date/time
- Set the priority of lot jobs
- Suspend lot jobs
- Resume lot jobs

(2) MesProductionLotCreator

This class generates MesProductionLot.

(3) MesConnectorToLotJobManager

This class has a function for interfacing between the production order management function group and process management function group. It sends a MesProductionLot object to a MesLotJobManager object.

3.3. Product Specification Management Function Group

3.3.1. Product specification management component

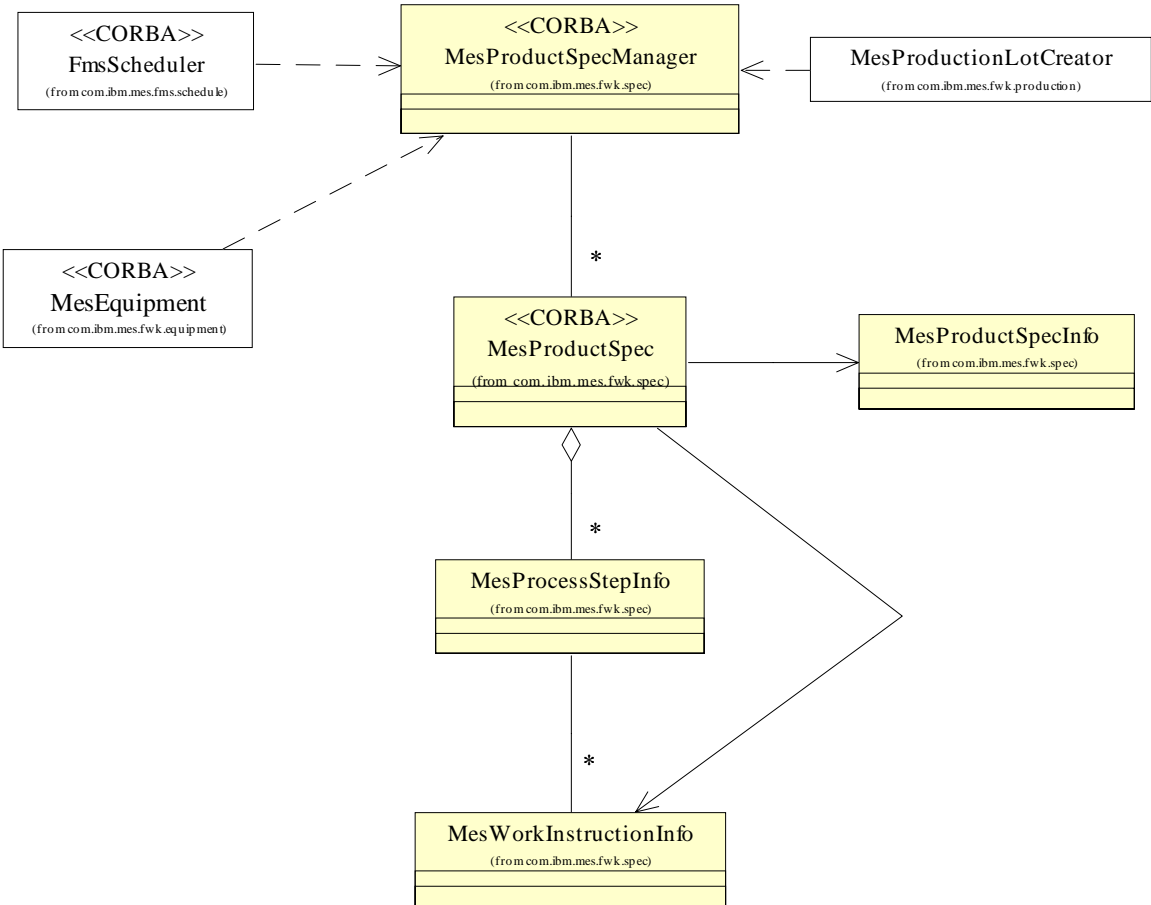


Figure 12 Class diagram of Product Specification Management Function Group

(1) MesProductSpecManager

This interface defines the methods for manipulations (registration, deletion, search, and modification) to be performed on product specifications (MesProductSpec).

(2) MesProductSpec

This interface defines the following methods, which manage product specifications (`MesProductSpecInfo`) and process step information (`MesProcessStepInfo`).

- Set or get a product ID
- Set or get a product specification name
- Register, delete, or search for process step information
- Register or delete work instruction information
- Set or get product specification information
- Increment or decrement the specification reference count by one. An associated product specification may be deleted when the reference count becomes 0.
- Set or get the product specification reference count
- Check whether this product specification is in use
- Set or get a process route
- Set or get a product version

(3) MesProcessStepInfo

This class changes the in-process work order diagram registrations and process step attributes.

- Set, get, or delete work instruction information (`MesWorkInstructionInfo`)
- Set or get product specification information (`MesProductSpecInfo`)
- Set or get a process step number
- Set or get a process step name

(4) MesWorkInstructionInfo

This class represents an in-process work order diagram. It is presumed that customization is to be effected on an individual MES basis.

- Set or get a manufacturing equipment ID
- Set or get a product equipment model number
- Set or get the related process step information
- Set or get the equipment mode

(5) MesProductSpecInfo

This class represents the additional information about product specifications. It is presumed that customization is to be effected on an individual MES basis.

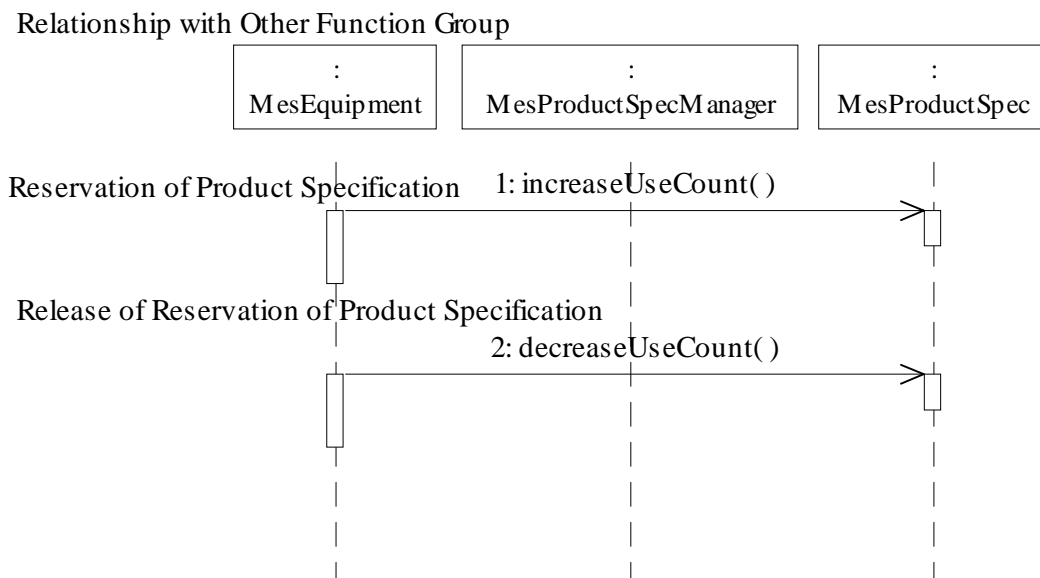


Figure 13 Sequence diagram for the increment/decrement of product specification in-use count

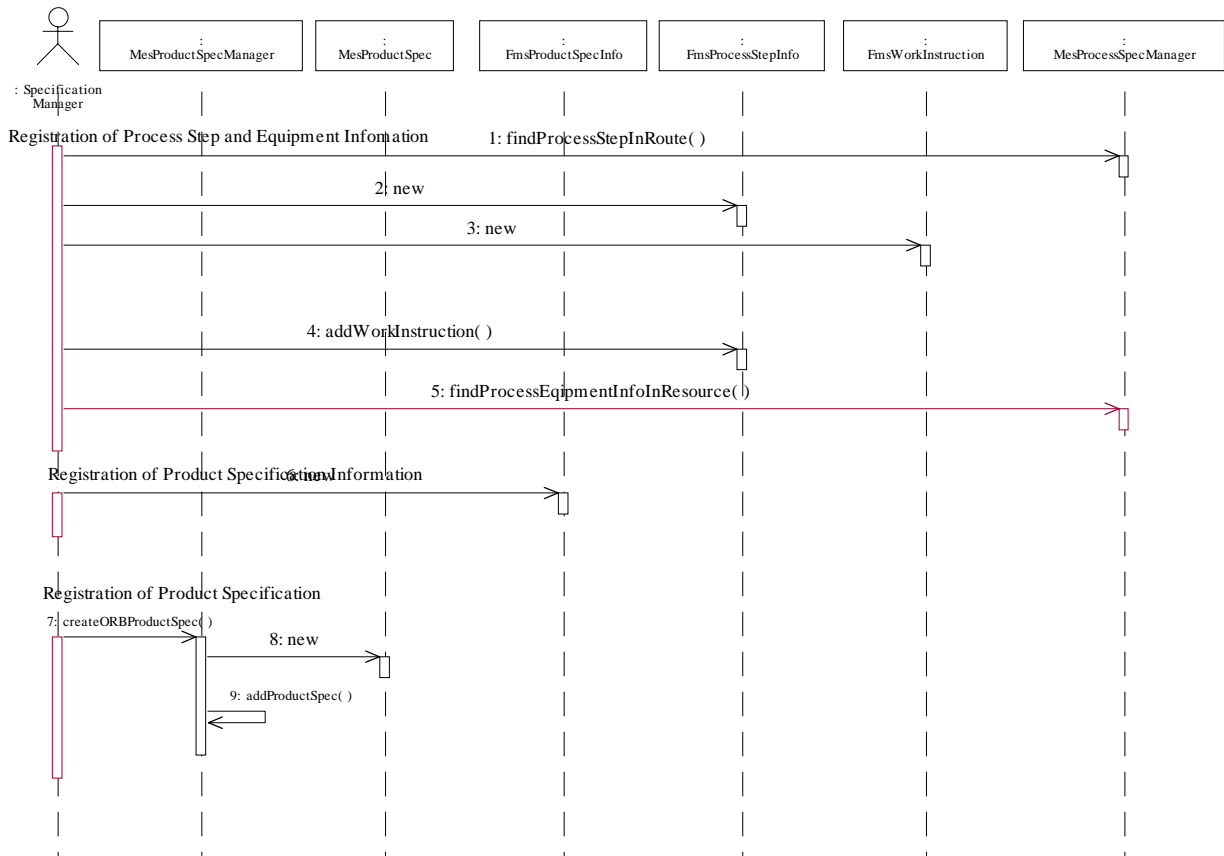


Figure 14 Sequence diagram for the registration of product specification

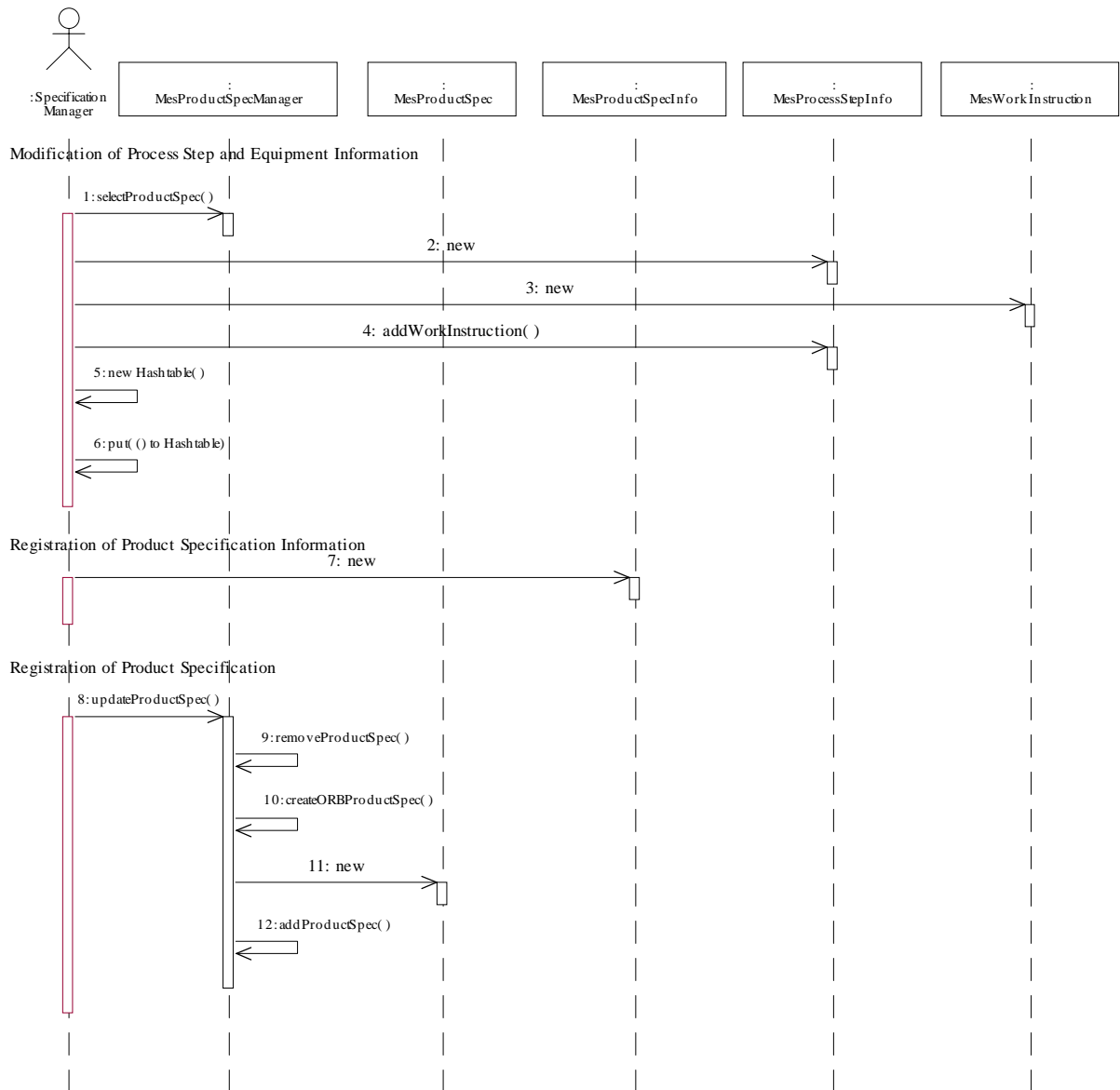


Figure 15 Sequence diagram for the modification of product specification

3.4. Process Management Function Group

The process management function group issues work orders in relation to equipment so as to perform manufacturing processes for a lot. It also follows up on a work order issuance to monitor the progress of work. Process management is mainly exercised by two classes: MesLotJob and MesDispatcher. MesLotJob manages a work plan and work result in relation to a lot, whereas Dispatcher sets work orders for various items of equipment in accordance with a work plan.

Here, the term "lot" refers to a physical lot that flows within processes or a production lot that corresponds to a physical lot. This time, modeling is done on the presumption that lots are not combined or divided from the work introduction to completion stages. It is also presumed that a collection of production orders in the same group is maintained in the production order management function group.

The process management function group consists of the following components.

- Production lot management component
- In-process work order management component
- Introduction order management component
- Transfer order management component

3.4.1. Production lot management component

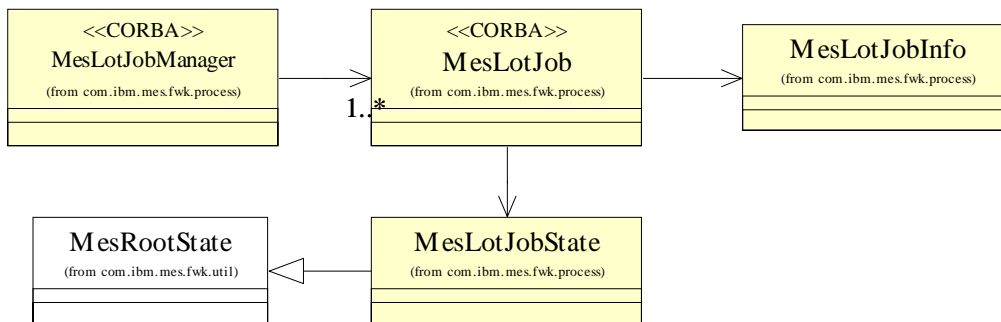


Figure 16 Class diagram of Production Lot Management Component

(1) MesLotJobServer

This class starts and manages an object for which the CORBA object MesLotJob is implemented.

(2) MesLotJobManagerServer

This class starts and manages an object for which the CORBA object MesLotJobManager is implemented.

(3) MesLotJobCreator

This class creates a lot job. For details about Creator, refer to the section on the common function group.

(4) MesLotJobHandlerServer

This class starts and manages an object for which the CORBA object MesLotJobHandler is implemented.

(5) MesLotJobHandler

This interface is provided for a collection of MesLotJob to be allocated in differing address spaces. It is not detailed in this document.

(6) MesLotJobManager

This interface exercises MesLotJob management. Declared methods offer the following functions.

- Create, delete, or search for lots
- Search for process jobs
- Release all lot jobs (to be executed after dispatching)
- Release specific process jobs
- Dispatch all process jobs
- Dispatch high-priority process jobs

(7) MesLotJob

This interface describes lot jobs. Declared methods offer the following functions.

- Register or search for process jobs
- Set or get the order of priority
- Set or get the planned start and finish time and actual start and end time
- Set or get the earliest introduction time and latest completion time
- Set or get the planned production volume
- Get the manufacturing yield
- Get the unacceptable yield
- Set or get the lot job information

- Set or get the state
- Make a request for execution start
- Report a suspension/resumption
- Report the completion of a process job
- Report the abort of a process job
- Report the cancellation of a process job
- Report the suspension of a process job
- Report the resumption of a process job

When changes are applied to the following items, they are conveyed to a work order (MesWorkOrder).

- Order of priority of all process jobs that comprise this object
- Order of priority of specific process jobs that are contained in this object
- Planned start time of a specific process job that is contained in this object

The historical information about the following items can be stored in a database at the time of execution.

- Make a request for execution start
- Report of a process job completion
- Report of a process job abort
- Report of a process job cancellation
- Report of a process job suspension
- Report of a process job resumption

(8) MesLotJobState

This class describes the state of a lot job.

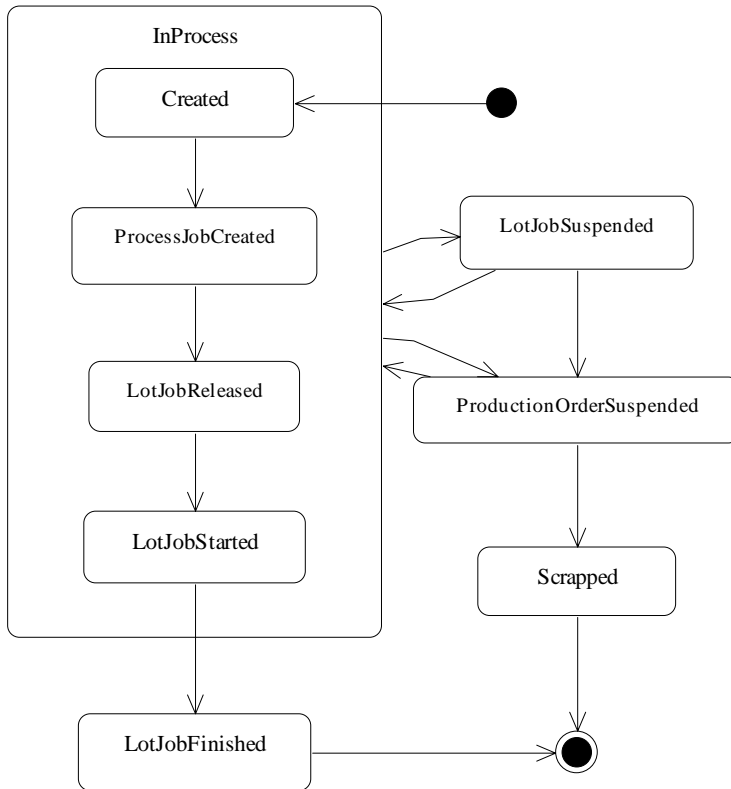


Figure 17 State diagram of MesLotJob

(9) MesLotJobInfo

This class is derived in applicable cases and used to define individual attributes and relevant operations that cannot be offered by MesLotJob. Application developers do not have to customize the CORBA object MesLotJob.

3.4.2. In-process job management component

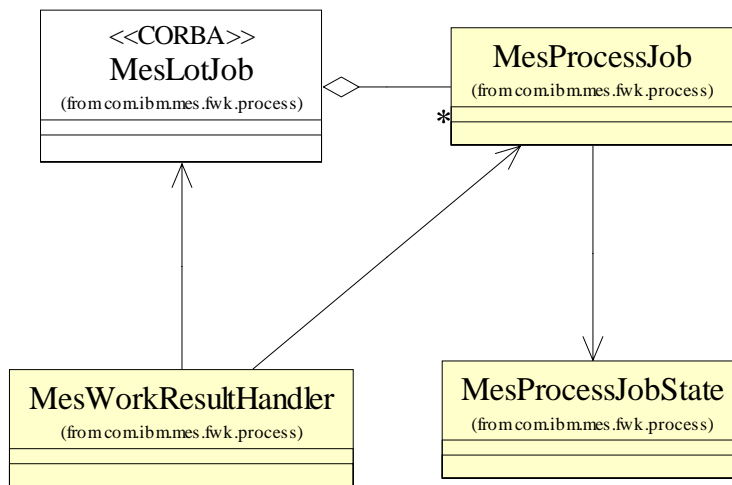


Figure 18 Class diagram of Process Job Management Component

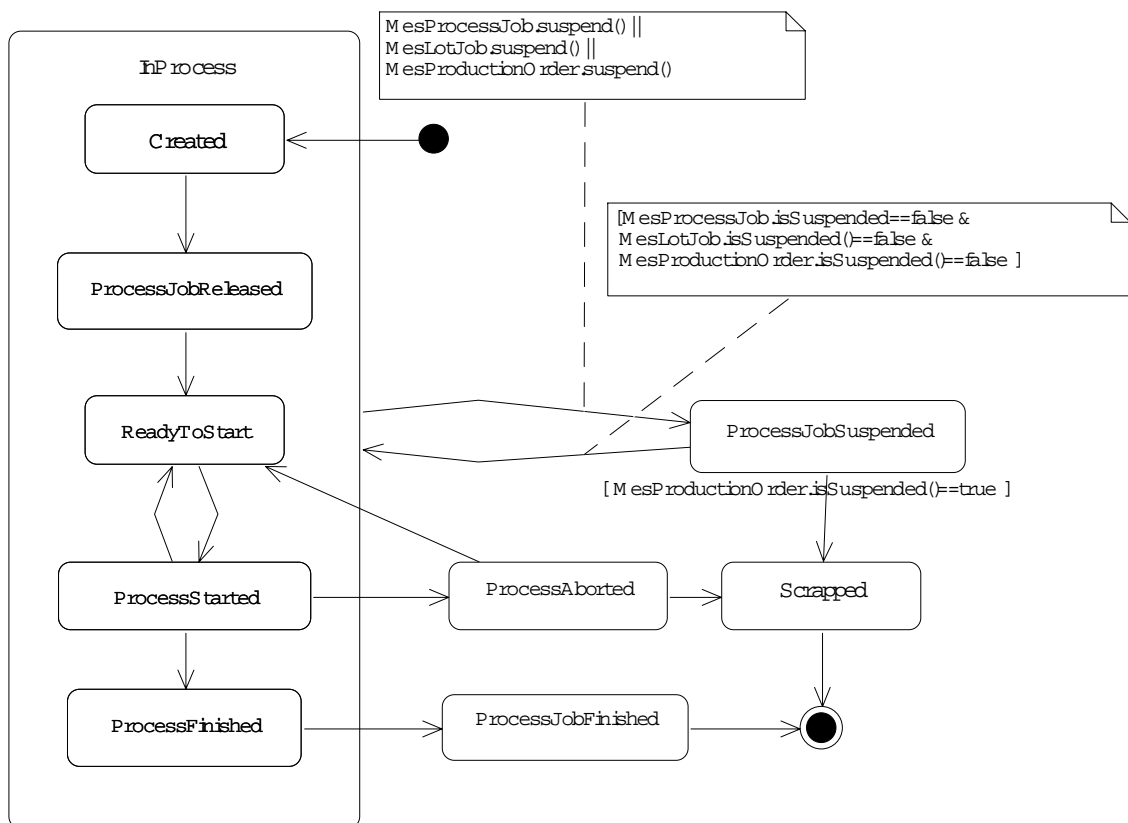


Figure 19 State diagram of MesProcessJob

(1) MesProcessJob

This class describes a process job. It has the following functions.

- Set or get the planned volume
- Set or get the actual volume
- Set or get the planned equipment
- Set or get the actual equipment
- Set or get the priority number
- Set or get the process resource
- Set or get the planned start time
- Set or get the planned finish time
- Set or get the actual start time
- Set or get the actual finish time
- Set work result data
- Set or get the unacceptable yield
- Set or get the work cancellation time
- Set or get the state
- Make a request for introduction
- Make a request for start
- Make a request for abort
- Make a request for cancellation
- Make a request for end
- Report a suspension
- Report a resumption

(2) MesProcessJobState

This class describes a process job state.

(3) MesWorkResultHandler

This class reports the following events to a corresponding production order (MesProductionOrder).

- Start of the first process job (MesProcessJob)
- Completion of the last process job (MesProcessJob)
- Work start and completion as a work result (MesWorkResult)

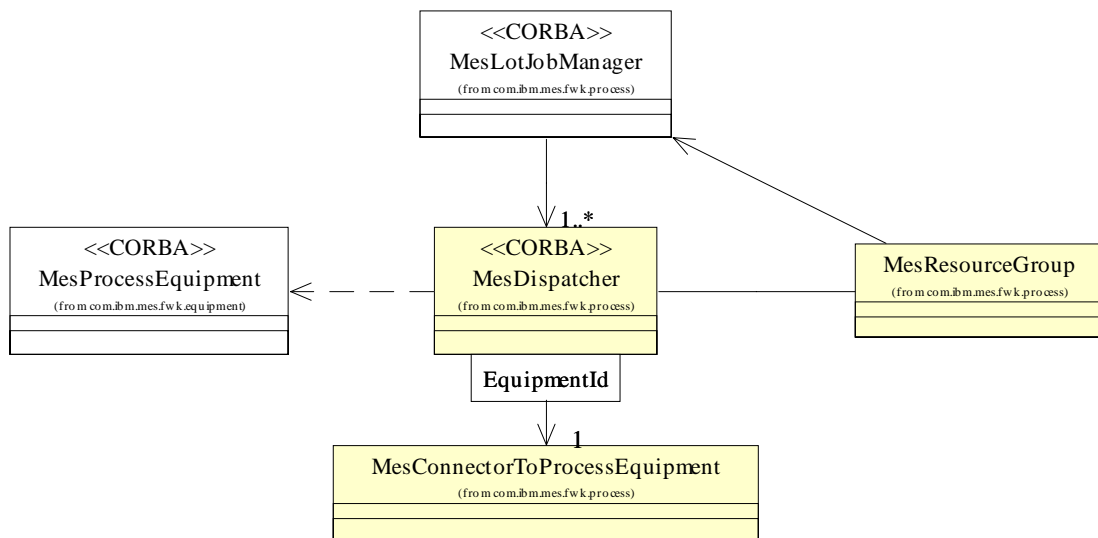


Figure 20 Class diagram of Order Release Management Component

3.4.3. Order release management component

(1) MesDispatcherServer

This class manages an object for which the CORBA object MesDispatcher is implemented.

(2) MesDispatcher

This interface defines the methods for setting a work order (MesWorkOrder) for process equipment (MesProcessEquipment).

- Register the historical information about dispatching
- Register a process job for a process equipment connector
- Dispatch all released process jobs
- Dispatch all released process jobs to designated process equipment
- Dispatch a designated number of process jobs having a high priority of dispatching to process equipment
- Dispatch a designated number of process jobs having a high priority of dispatching to specific process equipment
- Search for process jobs that are registered for designated process equipment
- Get a lot job manager ID
- Get a process job relational operator
- Make a reply to indicate whether a designated process equipment connector is registered
- Make it impossible to start a process job that is registered for a connector corresponding to alternate equipment
- Notify a process equipment connector of a process job deletion
- Delete a process job that is registered for a connector corresponding to alternate equipment
- Set a process equipment connector class name
- Set a lot job manager ID
- Set a process job relational operator
- Notify a process equipment connector of a process job parameter update

(3) MesConnectorToProcessEquipment

This class interfaces between the process management function group and equipment management function group. It is highly necessary that the connection between these two groups be customized by MES. This class is therefore provided to simplify the interdependence of the function groups.

The following two items need be customized.

- Register a work order (MesWorkOrder) for or deleting it from process equipment
- Reflect process job priority, planned start time, and suspension/release in work orders

(4) MesResourceGroup

Represents alternate equipment.

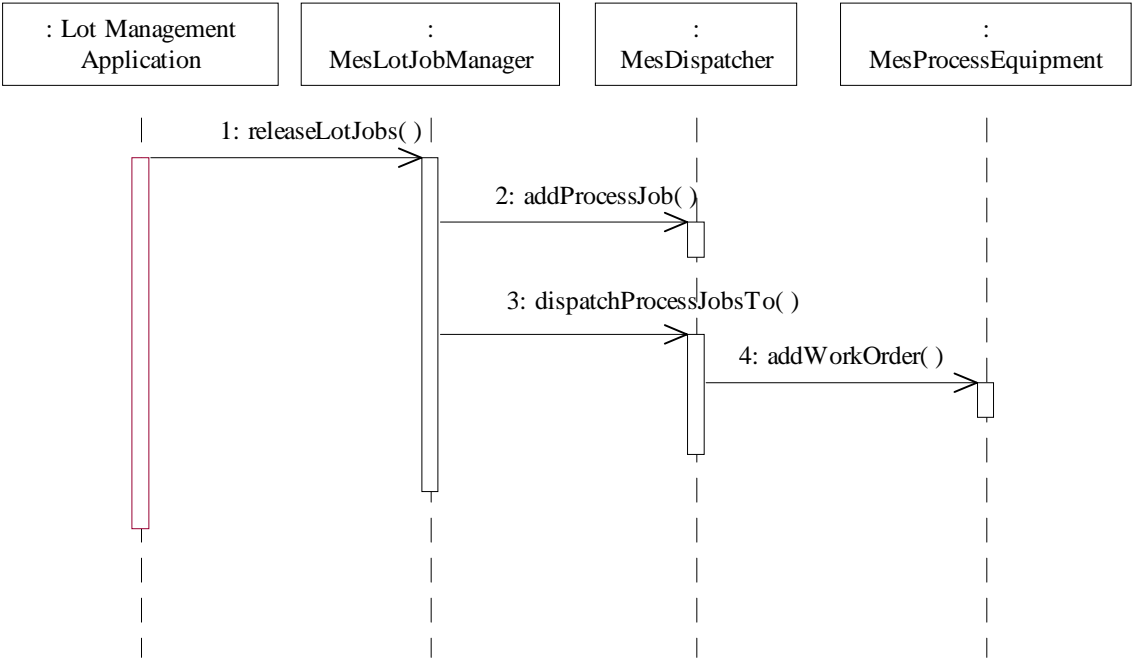


Figure 21 Sequence diagram for the release of lot jobs

Work Result Notification

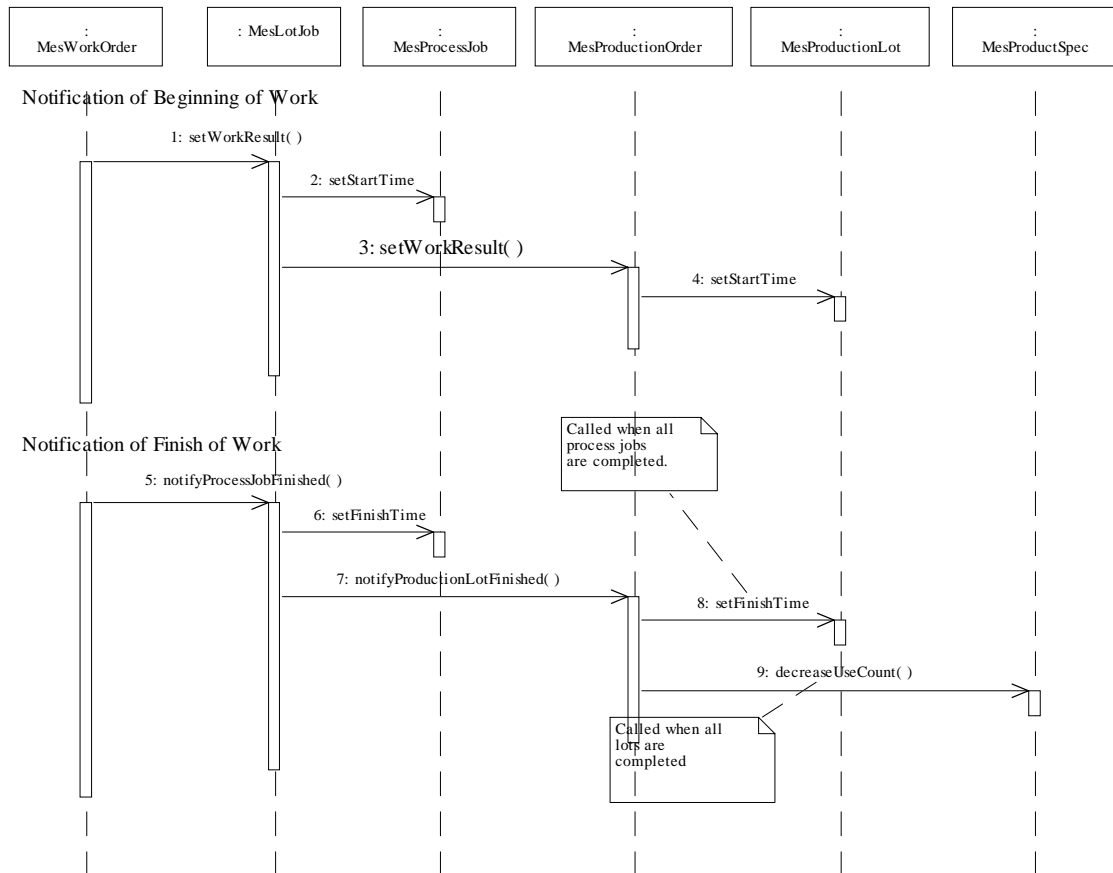


Figure 22 Sequence diagram for the notification of order release

3.4.4. Transfer order management component

This component provides functions used for the transfer order management.

(1) MesTransferRequestManager

Methods defined by this interface offer the following functions, which receive a request for forwarding a lot to equipment or removing a lot from equipment, and initiate a transfer operation when preparations for execution are completed, for instance, upon completion of target equipment operations.

- Make a request for forwarding a lot to equipment
- Make a request for loading an empty pallet for setup
- Make a request for unloading an emptied pallet
- Make a request for removing a lot from equipment
- Receive a notification of transfer completion



Figure 23 Class diagram of Transfer Order Management Component

3.5. Material Management Function Group

The material management function group manages the results of materials use. Consumable materials, called Consumable class, and durable materials, called Durable class, are defined as management targets. However, the overall inventory control of materials is not covered. For example, the master information about individual types of materials is not possessed. It is assumed that materials inventory control is exercised by ERP or other external system. The classes provided by the captioned group are designed to serve as an interface for notifying an external system of materials use.

3.5.1. Material management component

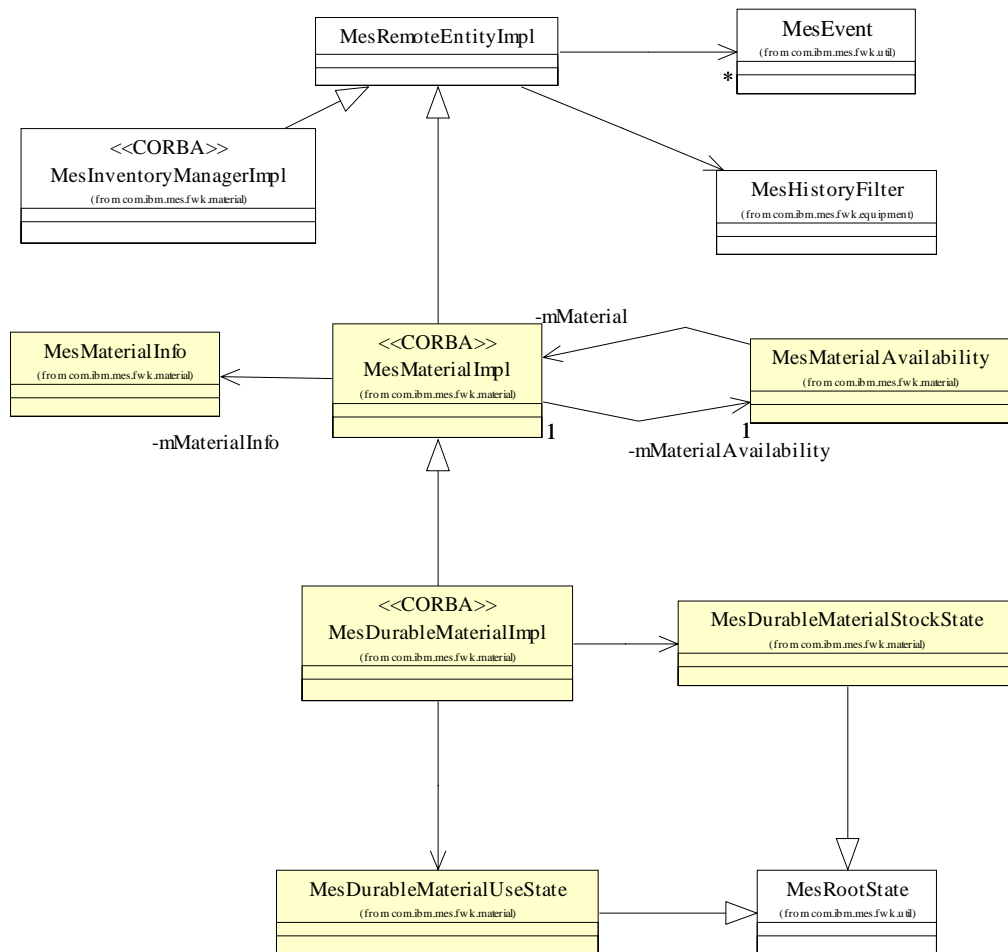


Figure 24 Class diagram of Material Management Function Group

(1) MesInventoryManagerServer

This class invokes and manages MesInventoryManager.

(2) MesInventoryManager

This interface defines the methods for managing (registering, searching for, or deleting) materials (MesMaterial).

(3) MesMaterial

This interface defines materials.

- Set or get the location
- Set or get material information (MesMaterialInfo)
- Set or get a material name
- Get a material type
- Check whether the end of useful life is reached

(4) MesMaterialAvailability

This abstract class defines a function for checking whether the end of material's useful life is reached.

(5) MesMaterialInfo

This abstract class defines materials information.

(6) MesDurableMaterial

This interface defines a durable material's functions that are inherited from a material.

- Set or get the actual use count
- Set or get the actual use time
- Get the remaining use count
- Get the remaining use time
- Set or get a state of durable material in the stock (MesDurableMaterialStockState)
- Set or get the upper-limit value for use count
- Set or get the upper-limit value for use time
- Set or get a use count of durable material (MesDurableMaterialUseState)
- Increment the actual use count by 1

- Add the actual use time

(7) MesDurableMaterialUseState

This class defines the aggregate of durable material use state.

- Ready (Ready)
- Being used (BeingUsed)
- Reserved (BeingReserved)
- Unusable (NotUsable)
- Disposed (Disposed)

(8) MesDurableMaterialStockState

This class defines the aggregate of durable material stock state.

- In stock (InStock)
- Being retrieved (BeingStockOut)
- In equipment (InEquipment)
- Being stocked (BeingStockIn)
- Disposed (Disposed)

(9) MesDurableMaterialAvailability

This class defines a function for checking whether the end of a durable material's useful life is reached. It is necessary to implement lifetime judgment logic.

3.6. Transfer Management Function Group

The transfer management function group offers lot transfer functions. However, it must be noted that the entire transfer system is often installed by a manufacturer specialized in transfer devices. Therefore, there is no knowing whether a standard model for an AGV or automatic warehouse in a transfer system can exist. In the OpenMES, therefore, some abstraction levels are provided for transfer system interfacing, and provision is made so that the entire transfer system can be handled as a black box at the highest abstraction level.

3.6.1. Transfer management component

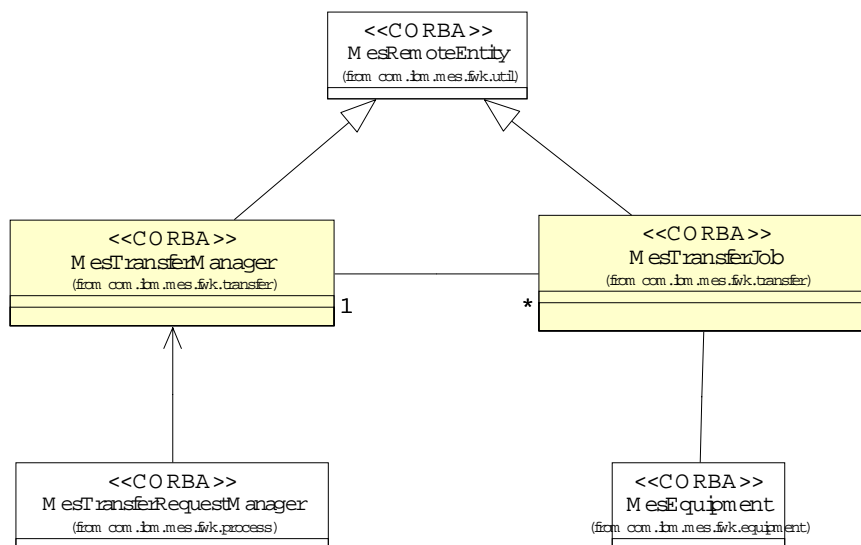


Figure 25 Class diagram of Transfer Management Function Group

(1) MesTransferManager

This interface defines transfer management functions.

- Set or get a transfer request manager (MesTransferRequestManager)
- Transfer a lot from a warehouse to equipment
- Transfer a lot from equipment to a warehouse
- Transfer a lot between equipment
- Get all the currently executed transfer jobs
- Receive a notification of transfer completion
- Get the number of currently executed transfer jobs

- Release a transfer job
- Get a transfer job (MesTransferJob)

(2) MesTransferJob

This interface defines transfer job functions.

- Get the transfer source equipment
- Get the transfer destination equipment
- Get a lot (MesLotJob)
- Set or get the order of priority
- Get the date/time of transfer job request issuance
- Get the date/time of transfer job start
- Abort a transfer job
- Cancel a transfer job
- Complete a transfer job
- Get the date/time of transfer job completion
- Get a transfer job state
- Receive the notification of a transfer job finished
- Receive the notification of a transfer job started
- Select a transfer job
- Deselect a transfer job

(3) MesTransferJobState

This class describes the state of a transfer job. A transfer job is in one of the following states.

- Ready for transfer (ReadyForTransfer)
- Waiting for transfer (WaitingForTransfer)
- Being transferred (UnderTransfer)
- Transfer aborted (TransferAborted)
- Transfer canceled (TransferCanceled)
- Transfer completed (TransferCompleted)

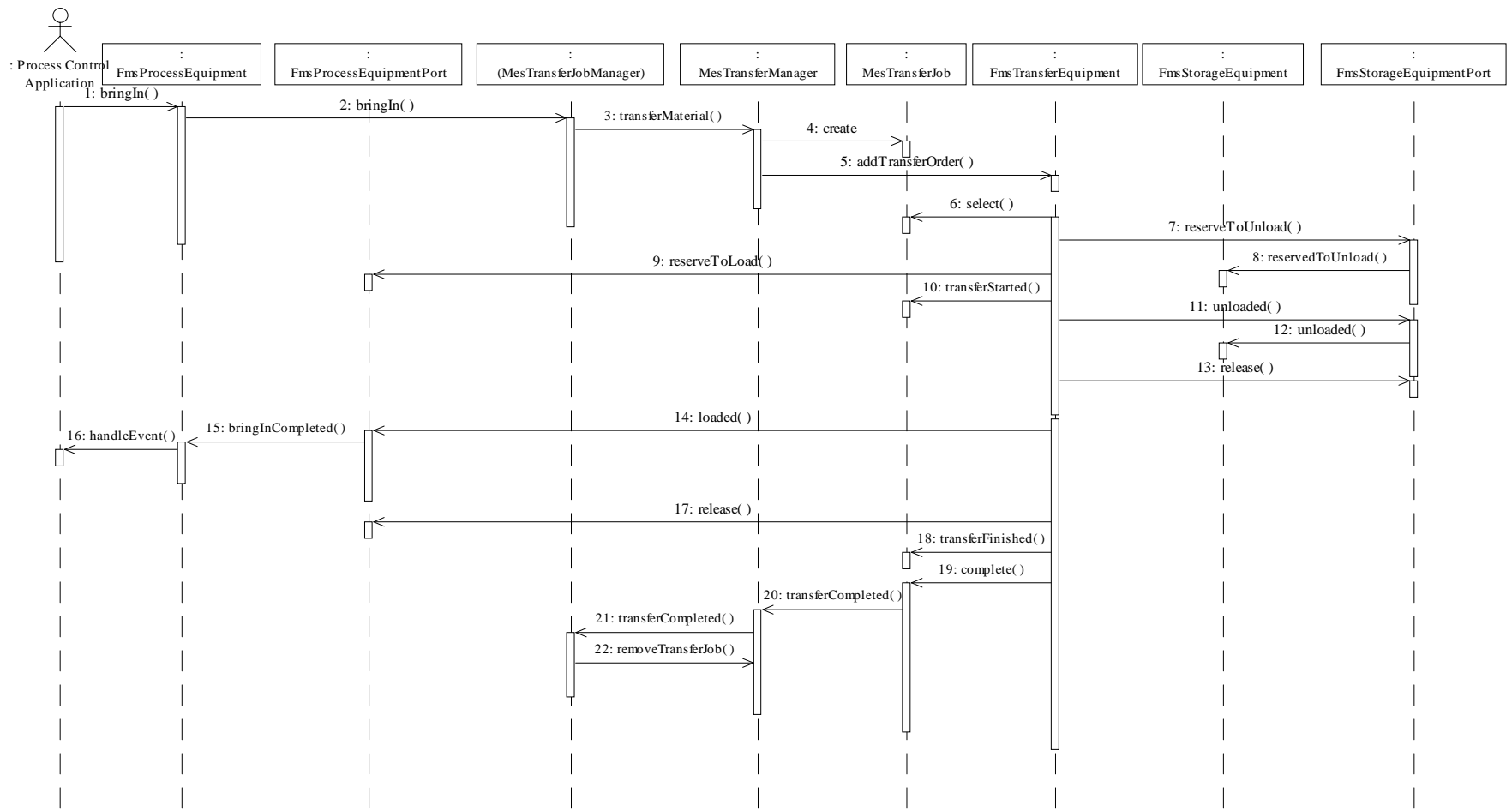


Figure 26 Sequence diagram for the lot transfer from storage to equipment

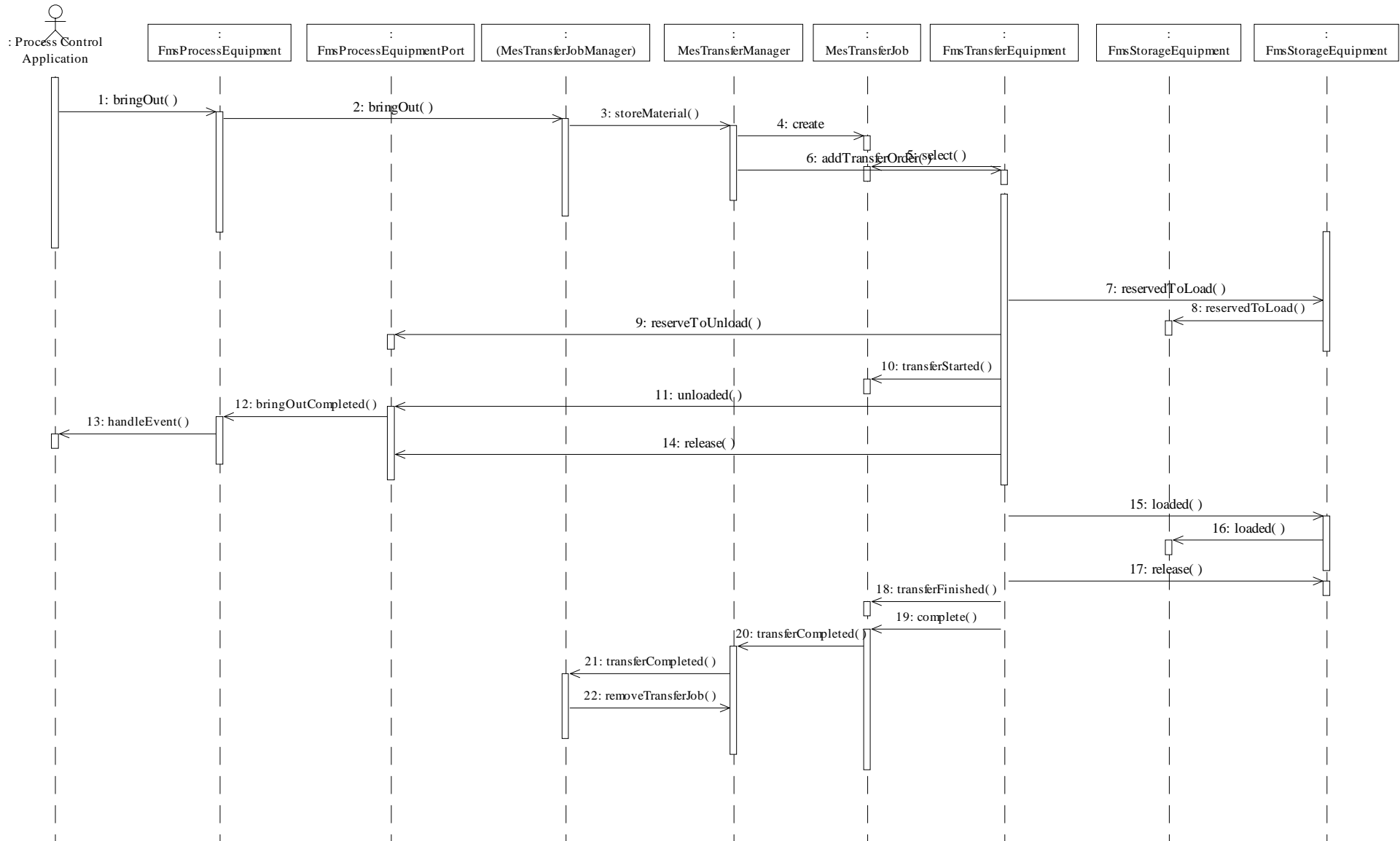


Figure 27 Sequence diagram for the lot transfer from equipment to storage

3.7. Process Specification Management Function Group

This function group provides the operations for dealing with process route and the capability of the process resource management. A process route (MesProcessRoute) consists of a sequence of process steps (MesProcessStep). In accordance with this information, a process job (MesProcessJob) is generated from a lot job (MesLotJob).

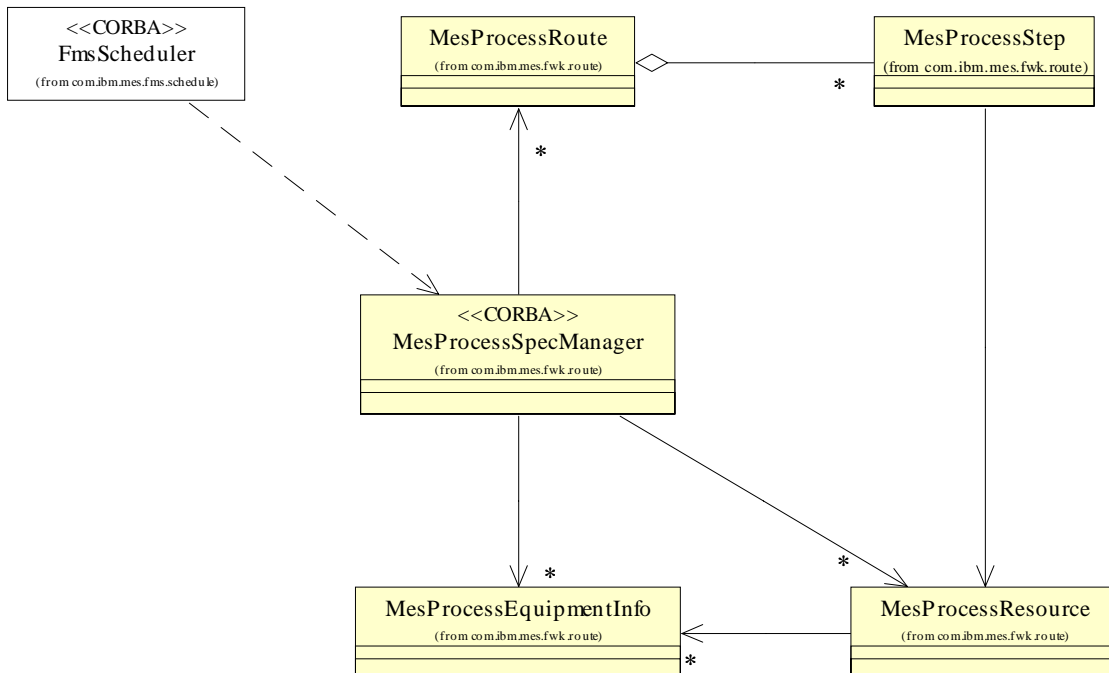


Figure 28 Class diagram of Process Specification Management Function Group

3.7.1. Process route management component

(1) MesProcessSpecManage

This interface defines the methods for operating on process route, process step, process resource, and manufacturing equipment objects.

- Register, search for, or delete process resource (MesProcessResource)
- Register, search for, or delete a process route (MesProcessRoute)
- Register a process step (MesProcessStep) for a process route
- Search a process route for a process step
- Increment or decrement the process route reference count by one

- Set, search for, change, or delete process equipment information (MesProcessEquipmentInfo)
- Associate process equipment information with a process resource
- Search for or delete process equipment information registered for a process resource
- Check whether a reference to a process route exists
- Allocate a process resource to a process step
- Deallocate a process resource from a process step
- Delete a process step from a process route

(2) MesProcessRoute

This class describes a process route.

- Register a process step for a process route
- Delete a process step from a process route
- Get a process step
- Increment or decrement the reference counter by one
- Set or get the reference counter
- Get information that indicates whether a process route is being used

(3) MesProcessStep

This class describes a process step.

- Set or get an associated a process route (MesProcessRoute)
- Set or get an associated process resource
- Register a process resource for a process step
- Delete a process resource from a process step

(4) MesProcessResource

This class offers a grouping function for manufacturing equipment that is capable of executing certain process steps.

- Set or get manufacturing equipment resource information
- Set or get a manufacturing equipment type
- Set or get an associated process step (MesProcessStep)
- Get a manufacturing equipment type
- Delete the manufacturing equipment for a process resource
- Register manufacturing equipment information for a process resource

(5) MesProcessEquipmentInfo

This class describes manufacturing equipment information.

- Set or get associated process resource (MesProcessResource)
- Set or get the model number of manufacturing equipment
- Set or get a manufacturing equipment type

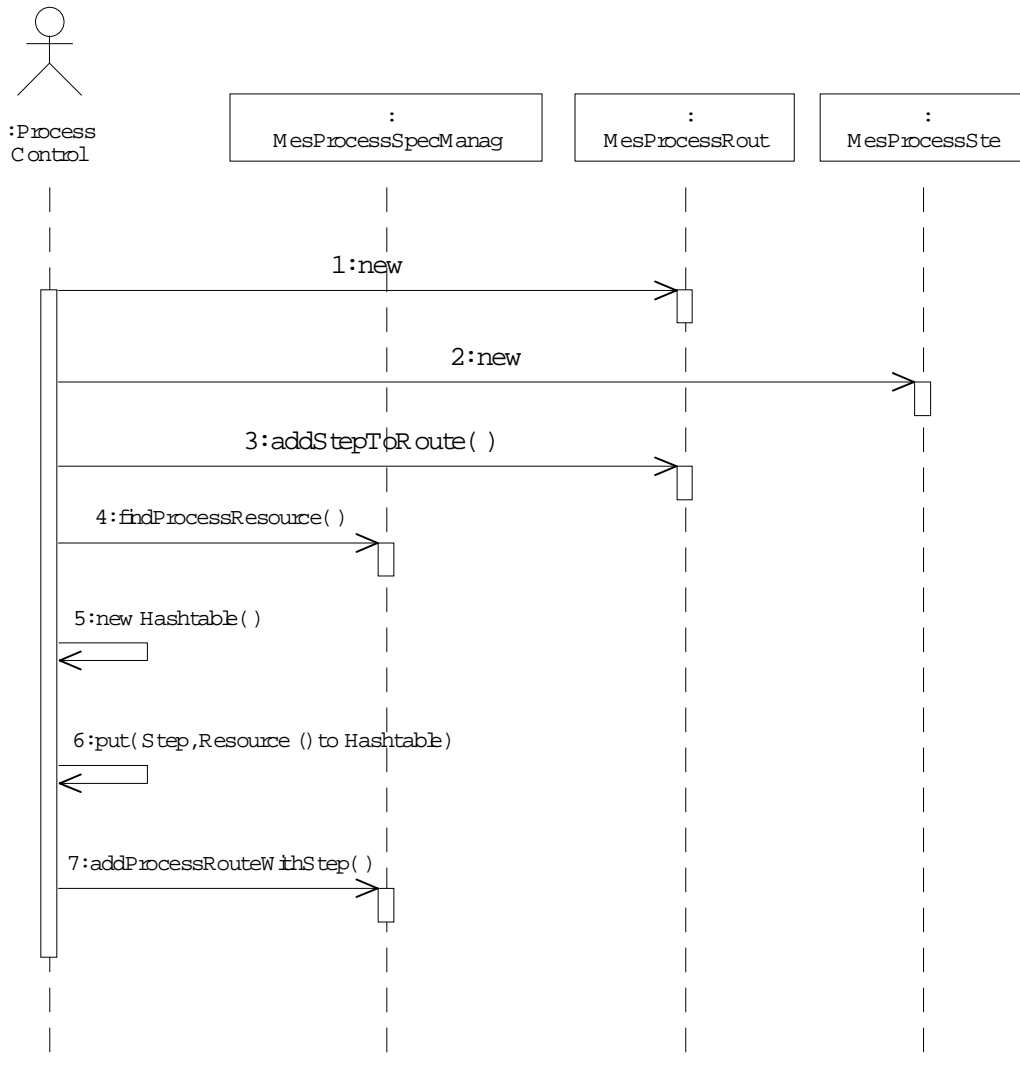


Figure 29 Sequence diagram for the registration of process route

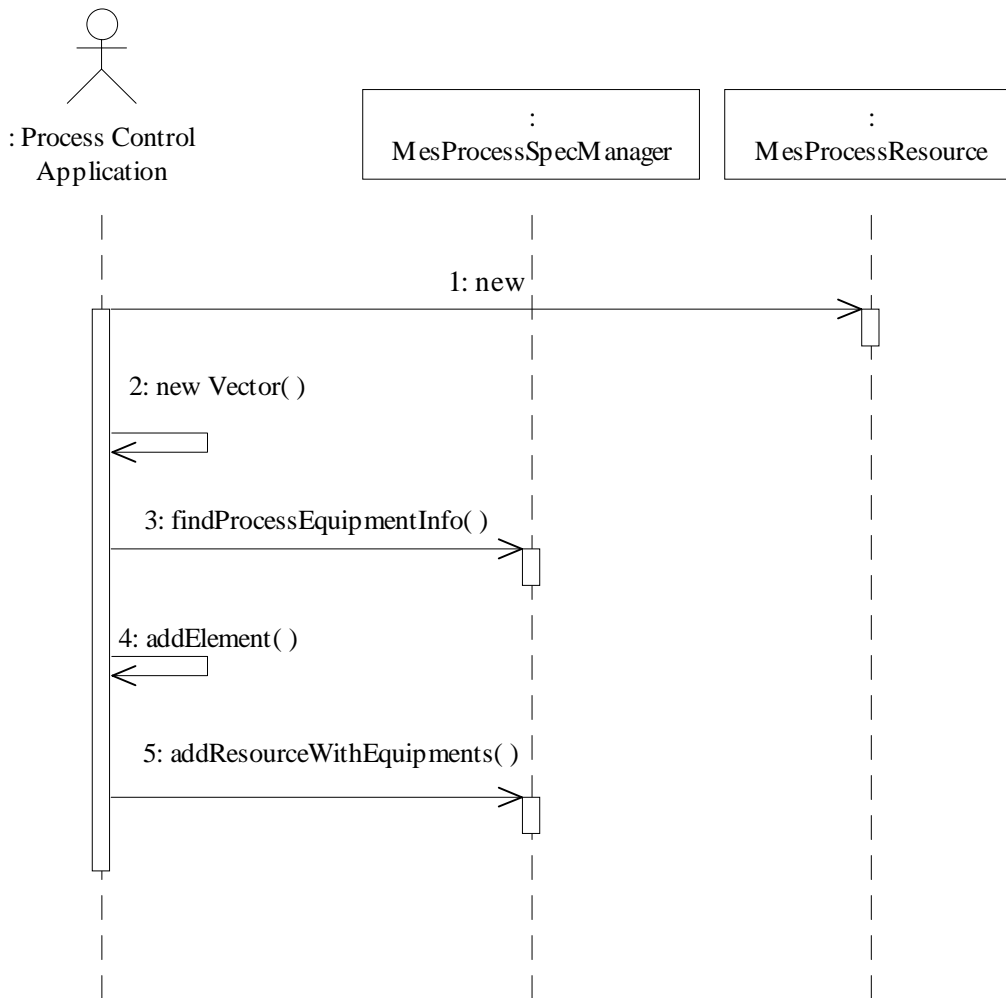


Figure 30 Sequence diagram for the registration of process resource

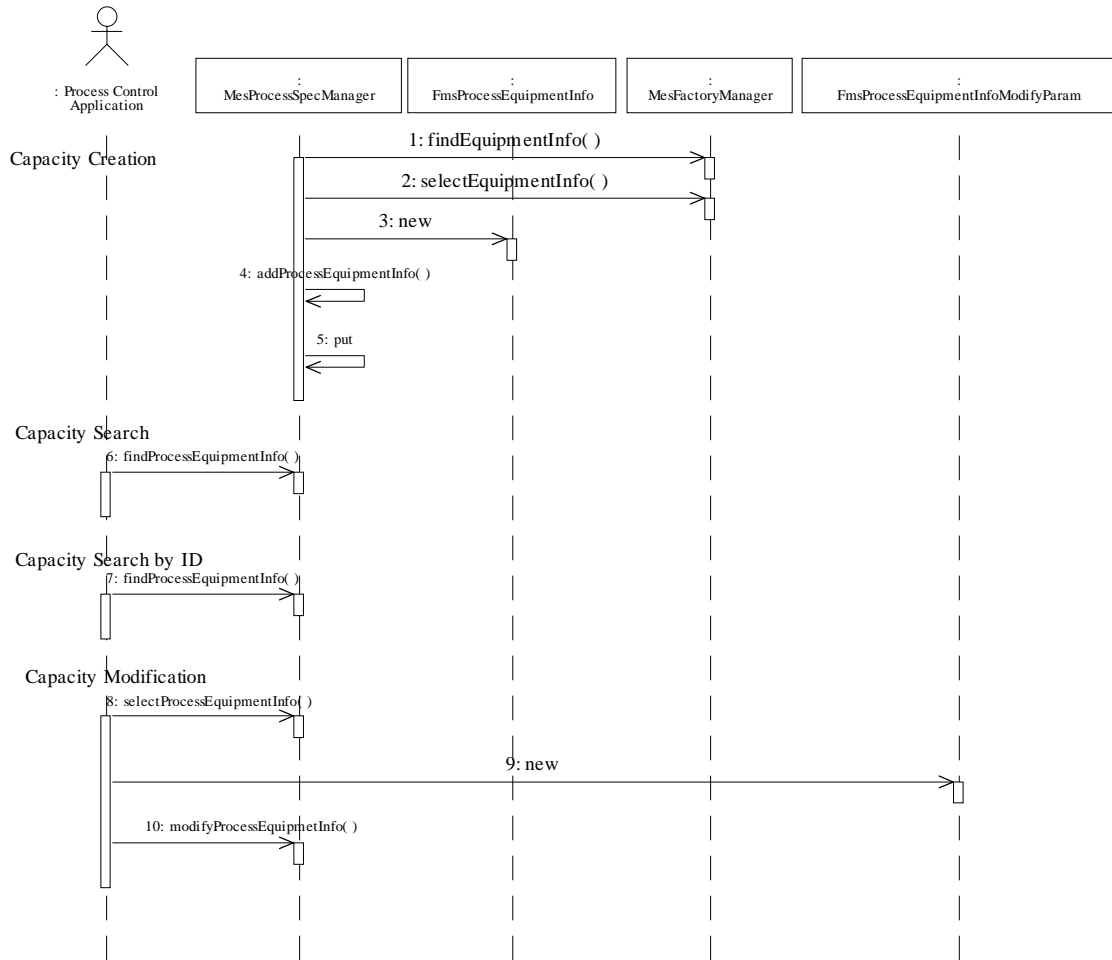


Figure 31 Sequence diagram for the registration of process capacity

3.8. Equipment Management Function Group

The equipment management function group offers functions for interfacing with work orders, work results, alarm, and other information exchanged between manufacturing equipment and MES. This function group consists of classes that correlate to various items of manufacturing equipment and a class that manages all such items of manufacturing equipment.

Equipment handled by this group is abstract equipment that does not have any physical entity. Abstract equipment is modeled on the presumption that it has the following three properties.

- Equipment as a unit of operating state management
- Equipment as a unit of process execution
- Equipment as a unit of control hardware

3.8.1. Equipment management component

The process management function group sets work orders (*MesWorkOrder*) to process equipment (*MesProcessEquipment*). The process equipment then calls a control device API via an equipment adapter (*MesEquipmentAdapter*). Upon completion of work, the control device calls a call back procedure of the process equipment (*MesProcessEquipmentCallback*), and a work result (*MesWorkResult*) is associated with a corresponding work.

Process equipment has a function for interfacing with the process management function group, and a process equipment adapter has a function for interfacing with a control device. Therefore, the knowledge of CORBA is not required for the connection to equipment.

Abstract Process Equipment

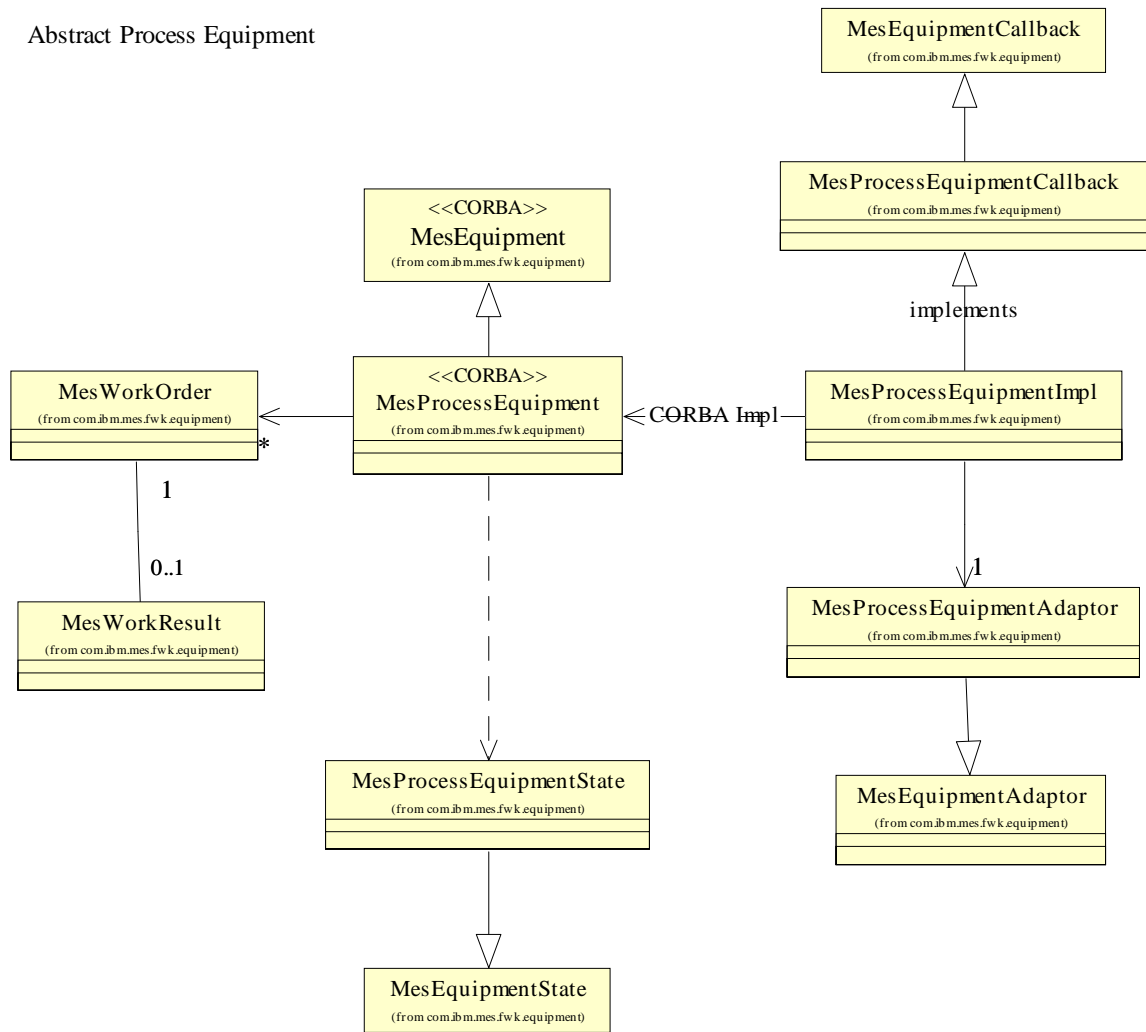


Figure 32 Class diagram related to MesProcessEquipment in Equipment Management Component

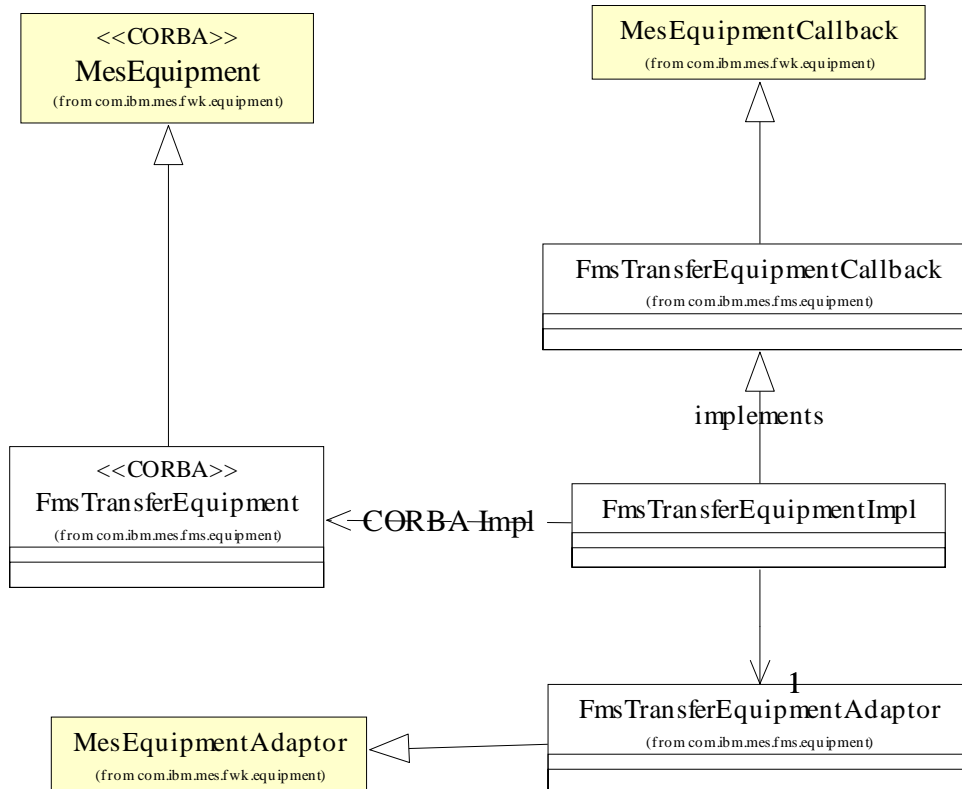


Figure 33 Class diagram related to MesTransferEquipment in Equipment Management Component

(1) MesEquipment

This interface defines the following functions for abstract equipment.

- Set the equipment installation site, model number, and type (MesEquipment)
- Get the equipment installation site, model number, and type (MesProcessEquipment)
- Start up or shut down the equipment (MesPrpcessEquipmentAdaptor)
- Report the issuance of an alarm
- Check whether an alarm is currently issued
- Get the currently issued alarm
- Clear all alarms
- Make a request for operator intervention
- Check whether a request for operator intervention has been issued
- Clear a request for operator intervention

- Check whether the equipment is online
- Set the equipment offline to make it unavailable in a manufacturing process
- Set the equipment online to make it available in a manufacturing process
- Register a transfer device port (MesPort) for the equipment
- Search for all ports that are registered for the equipment
- Delete a port from the equipment
- Get a port registered for the equipment

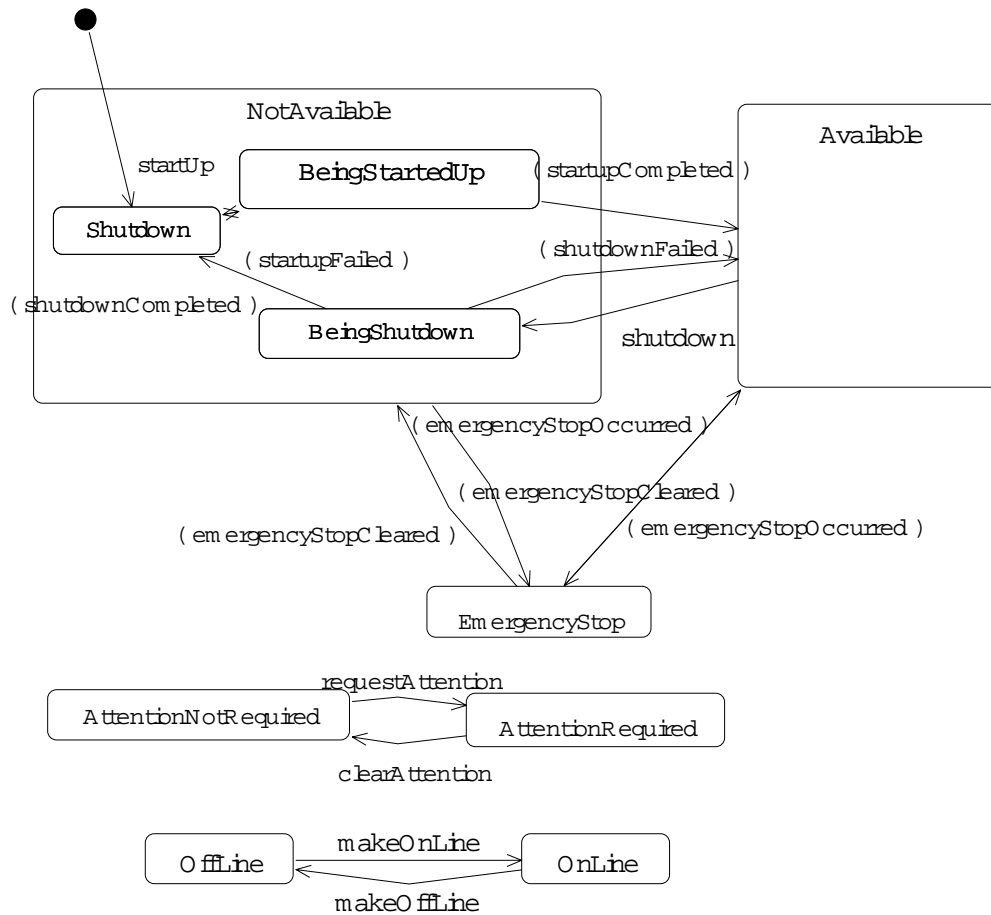


Figure 34 State diagram of MesEquipment

(2) MesEquipmentAdapter

This interface complies with a request from abstract equipment to define the functions for calling an associated control device.

- Start up the equipment
- Shut down the equipment

- Clear the currently issued alarm
- Set the equipment offline
- Set the equipment online

(3) MesEquipmentCallback

This interface defines the methods under which abstract equipment is called by its control device.

- Receive a notification of equipment startup sequence initiation
- Receive a notification of equipment startup sequence completion
- Receive a notification of equipment startup sequence cancellation
- Receive a notification of equipment shutdown sequence initiation
- Receive a notification of equipment shutdown sequence completion
- Receive a notification of equipment shutdown sequence cancellation
- Receive a notification of the clearance of an alarm
- Receive a notification of the clearance of all alarms
- Receive a notification that the equipment is online
- Receive a notification that the equipment is offline
- Receive a notification that the equipment is released from emergency stop state
- Receive a notification that the equipment is brought to an emergency stop

(4) MesEquipmentManager

This interface provides functions to register and acquire equipment (MesEquipment).

(5) MesPort

This interface defines the functions of a port (pallet changer or other joint with a transfer device).

(6) MesProcessEquipment

This interface defines the functions of abstract process equipment that is targeted for process assignment.

- Register an in-process work order
- Get all in-process work orders
- Get the in-process work order to be carried out next
- Specify the state to acquire an in-process work order
- Get work instructions related to an in-process work order

- Suspend in-process work
- Release suspended in-process work
- Get an in-process work order
- Cancel in-process work
- Abort in-process work
- Request the start of in-process work
- Complete in-process work
- Add an in-process work log
- Get the operation mode (automatic, semiautomatic, or manual) currently selected for process equipment
- Select the automatic operation mode
- Select the manual operation mode
- Select the semiautomatic operation mode
- Set or get an operation mode specific to process equipment. The specific operation mode is automatic, semiautomatic, or manual.
- Delete an aborted in-process work order
- Delete a completed in-process work order
- Delete a designated in-process work order
- Update the information about an in-process work order in accordance with the parameters related to update of a work order (`MesWorkOrderUpdateParam`)

(7) MesProcessEquipmentAdapter

This interface defines the functions that enable abstract process equipment to call its control device.

- Change the equipment operation mode to the automatic operation mode
- Change the equipment operation mode to the manual operation mode
- Change the equipment operation mode to the semiautomatic operation mode
- Start in-process work
- Receive a notification of the addition of an in-process work order

(8) MesProcessEquipmentCallback

This interface defines the callback that abstract process equipment receives from its control device.

- Request the permission for work initiation
- Receive a notification of work initiation

- Receive a notification of work completion
- Receive a notification of work suspension

(9) MesAlarm

This class defines the alarms that may be generated within equipment.

(10) MesAlarmEvent

This class defines the events that report alarm state changes (the generation or clearance of alarms).

(11) MesEquipmentAttentionEvent

This class defines the events that report attention (operator intervention) state changes.

(12) MesEquipmentLineModeEvent

This class defines the events that report line mode changes (online or offline).

(13) MesEquipmentManagerServer

This class defines an equipment startup server.

(14) MesEquipmentOpMode

This class defines the equipment operation mode (automatic, semiautomatic, or manual). The operation mode defines the method under which process equipment (MesProcessEquipment) performs process work. It is returned in response to an inquiry made about the process equipment operation mode.

(15) MesEquipmentOpModeEvent

This class defines the events that report operation mode (MesEquipmentOpMode) changes.

(16) MesEquipmentProcessingEvent

This class defines the events that report changes (start, completion, or stop) in the processing state within equipment.

(17) MesEquipmentServer

This class defines the startup server for abstract equipment.

(18) MesEquipmentState

This class defines the operation state of equipment.

- Available
- Unavailable
- Being started up

- Being shut down
- Shut down
- Brought to an emergency stop
- In an unknown state

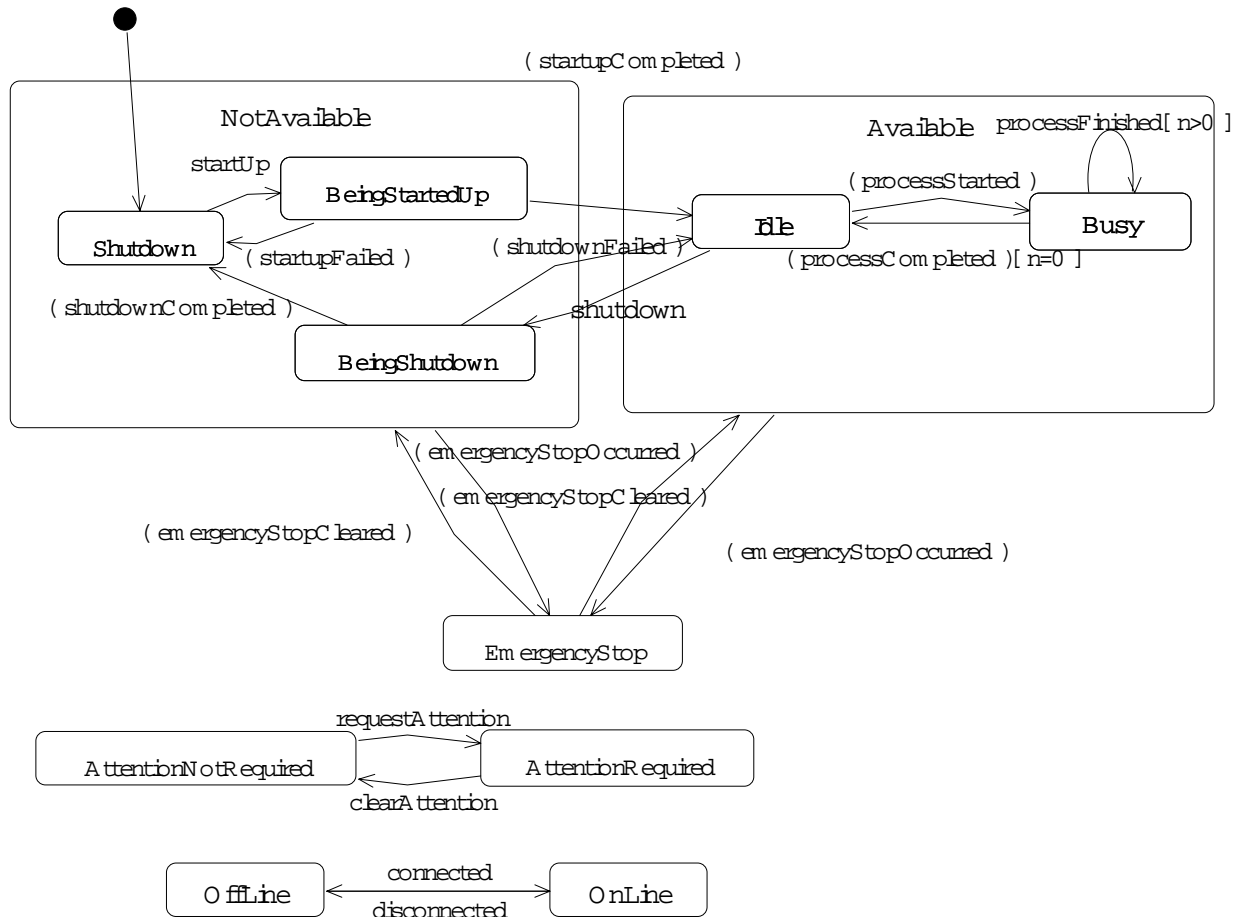


Figure 35 State diagram of MesProcessEquipment

(19) MesEquipmentStateEvent

This class defines the events that report changes in the equipment operation state (MesEquipmentState).

(20) MesProcessEquipmentServer

This class defines the startup server for abstract process equipment.

(21) MesProcessEquipmentState

This class defines the equipment operation state (on standby or executing).

(22) MesProcessEquipmentStateEvent

This class defines the events that report changes in the equipment operation state.

(23) MesWorkHistoryElement

This class defines the history elements (time stamp, lot ID, process ID, in-process work order ID, and comment) that are to be recorded together with in-process work descriptions.

(24) MesWorkInstruction

This class defines the in-process work instructions to be passed to process equipment.

(25) MesWorkOrder

This class defines in-process work orders to be received by process equipment.

- Set or get the ID of an in-process work order
- Set or get the process ID of an in-process work order
- Set or get the ID of a lot containing an in-process work order
- Set or get the planned work volume
- Set or get the planned work start date/time
- Set or get the planned work completion date/time
- Set or get the priorities of in-process work orders
- Set or get a lot job (MesLotJob)
- Set the ID of equipment to which an in-process work order is assigned
- Get the priorities of in-process work orders set
- Get the in-process work order state (MesWorkOrderState)
- Set or get the necessity for a setup change
- Inhibit in-process work from being started
- Permit in-process work to be started
- Report the start, completion, or stop of processing
- Request the permission for in-process work initiation
- Abort in-process work
- Cancel in-process work
- Complete in-process work

- Suspend an in-process work order
- Get the suspension of an in-process work order
- Release a suspended in-process work order
- Get the global suspension of in-process work orders
- Activate or deactivate the global (whole alternate equipment) suspension of in-process work orders
- Activate or deactivate the local (specific equipment) suspension of in-process work orders
- Set the priorities of in-process work orders set
- Register work results
- Register the history of in-process work

(26) MesWorkOrderAddParam

This class defines the parameters (an object requesting in-process work, in-process work instructions, and in-process work order) for assigning in-process work orders to process equipment.

(27) MesWorkOrderEvent

This class defines the events that report changes in the state of in-process work orders (MesWorkOrderState) for equipment.

(28) MesWorkOrderState

This class defines the states of in-process work orders.

- Work aborted
- Work completed
- Work being performed
- Work initiation impossible
- Work being performed and processing completed
- Work being performed and processing initiation impossible
- Work being performed and processing initiation possible
- Work being performed and processing started
- Work being performed and processing stopped
- Work initiation possible
- Work ready for initiation with the previous process completed but globally or locally suspended

(29) MesWorkOrderTransferState

This class defines the states of a lot transfer designated by an in-process work order.

- Loading in progress in compliance with a loading order
- Unload in progress
- Load completed
- Unload completed
- Load not started

(30) MesWorkOrderUpdateParam

This class defines the parameters for updating the information about in-process work orders registered with process equipment.

- Planned work completion date/time
- Planned work initiation date/time
- Order of priority
- Global suspension flag
- Priorities of work orders set
- In-process work instructions
- In-process work order ID

(31) MesWorkResult

This class defines the in-process work results to be returned by process equipment. In-process work orders can be requested to update in-process work results.

- Set or get the ID of the equipment that has performed in-process work
- Set or get the number of defective articles produced by in-process work
- Set or get the date/time of in-process work completion
- Set or get the number of articles completely processed by in-process work
- Set or get the ID of the person in charge of in-process work
- Set or get the date/time of in-process work initiation

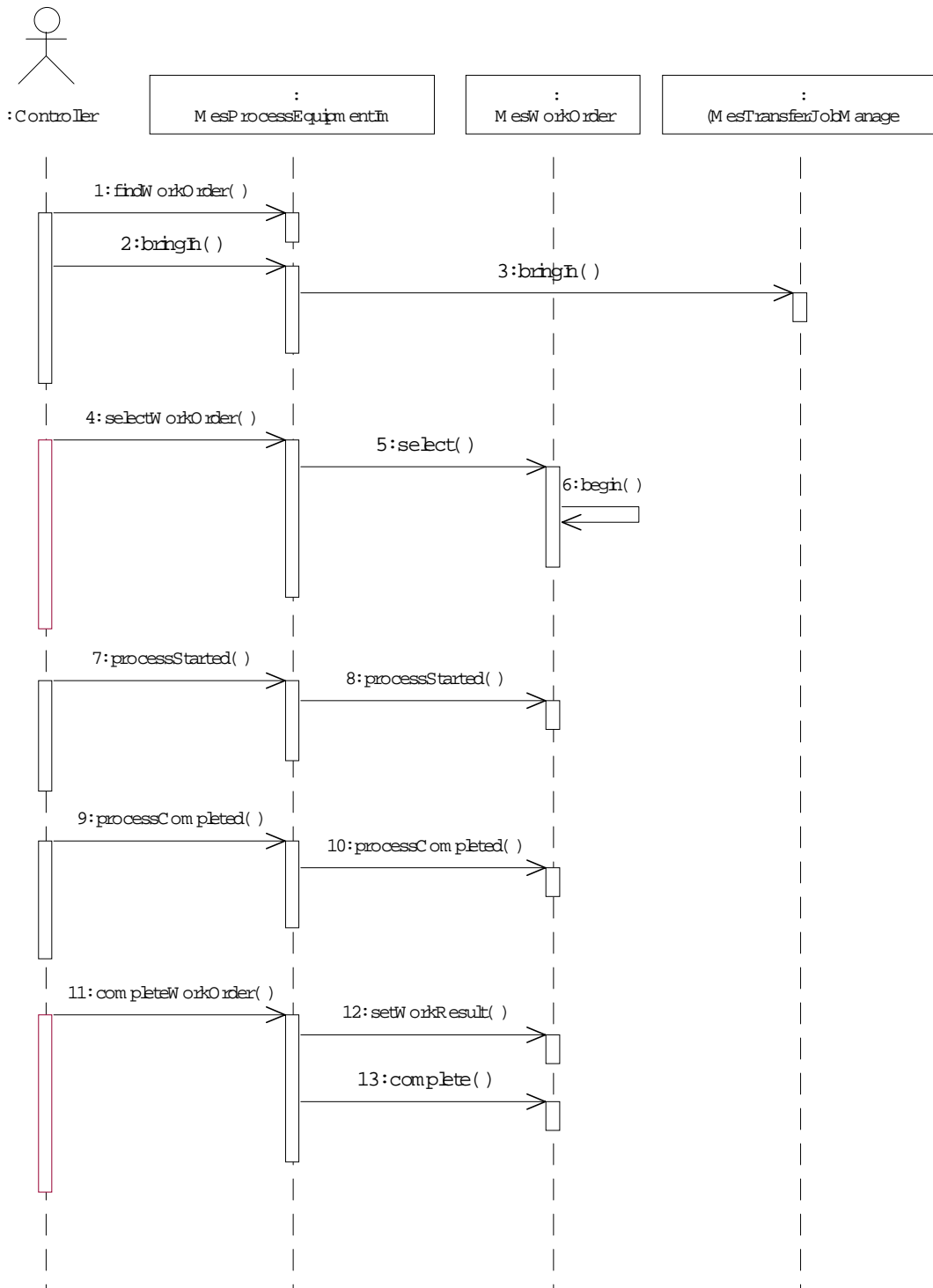


Figure 36 Sequence diagram for the automatic execution of process equipment

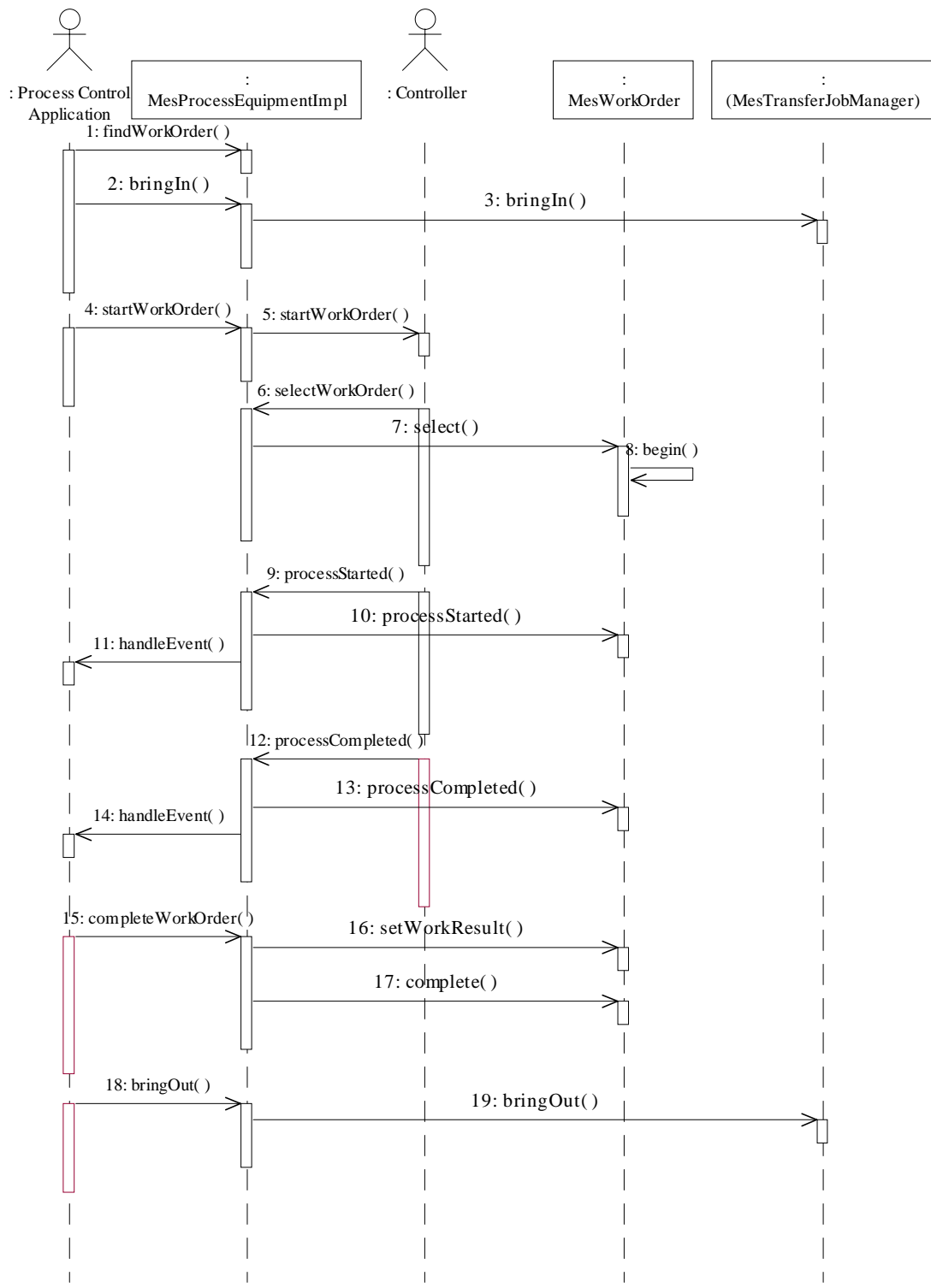


Figure 37 Sequence diagram for the semi-automatic execution of process equipment

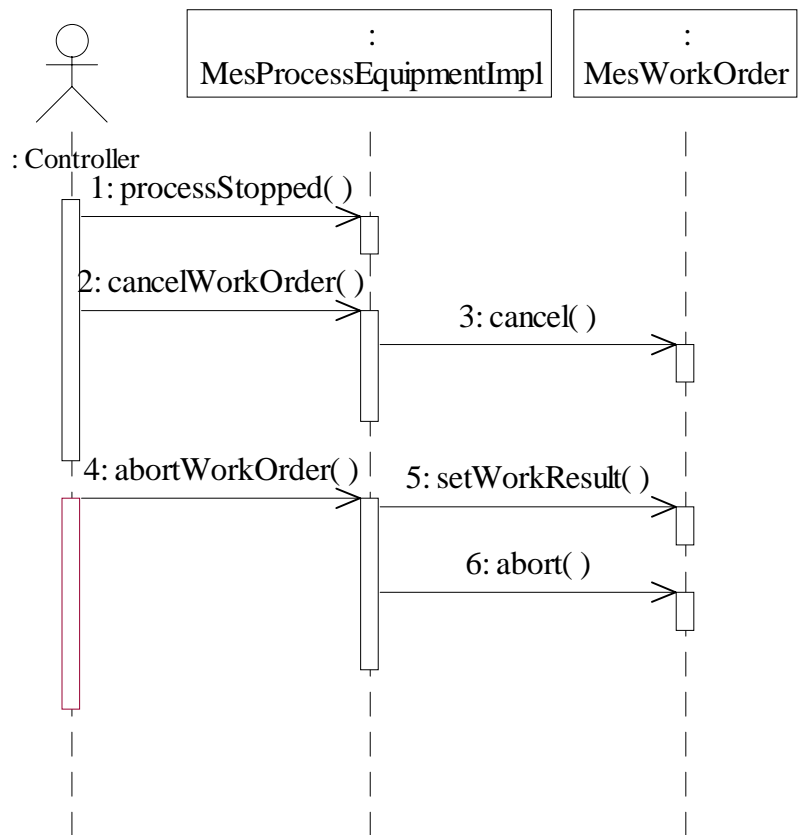


Figure 38 Sequence diagram for the suspend/resume of process equipment in automatic execution

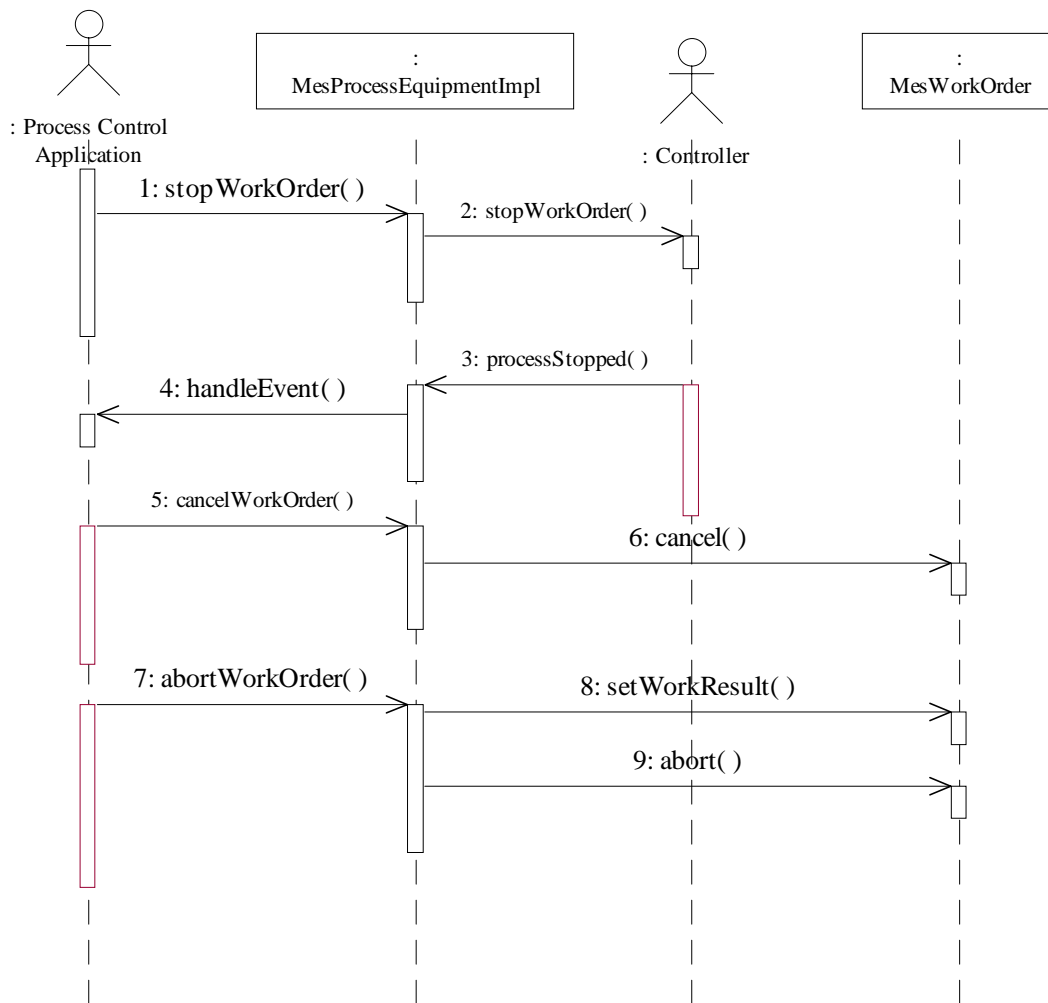


Figure 39 Sequence diagram for the suspend/resume of process equipment in semi-automatic execution

3.9. Schedule Management Function Group

3.9.1. Scheduler interface component

(1) MesScheduler

This interface defines the function for interfacing with a scheduler. Since an existing scheduler is to be connected, only an abstract method is defined.

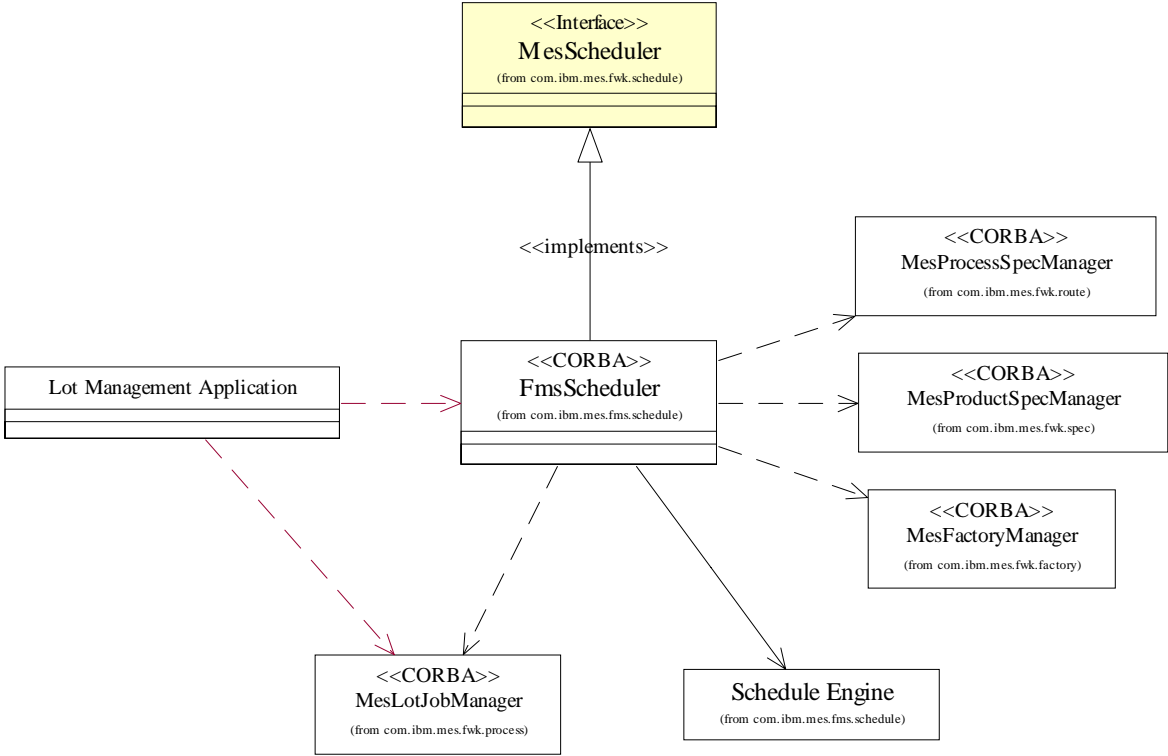


Figure 40 Class diagram of Schedule Management Function Group

3.10. Common Function Group

3.10.1. Event Notification Management Component

Event Service of CORBA is used. For details about an event model, refer to the CORBA specification.

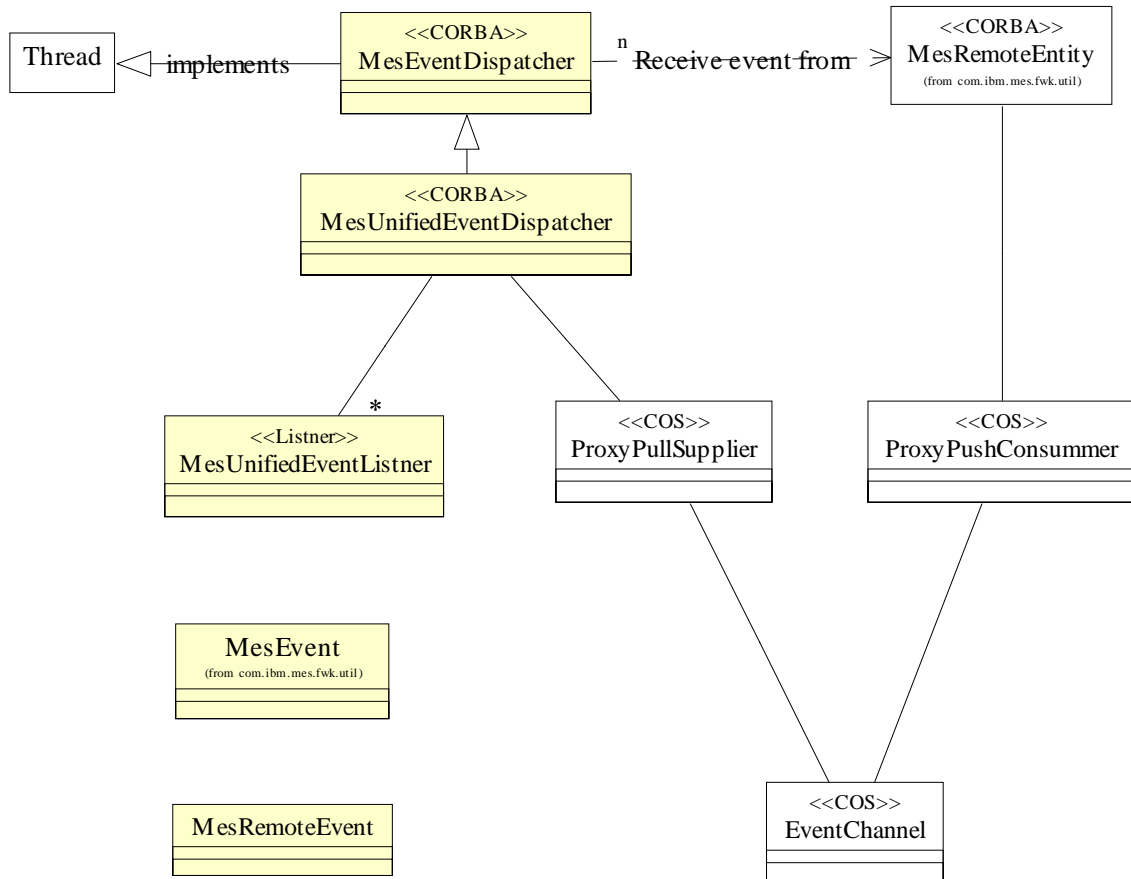


Figure 41 Class diagram of Event Notification Management Component

(1) MesEvent

This class defines events that occur within the component.

(2) MesEventDispatcher

This class defines a dispatcher that receives events from a CORBA server and distributes them to Java objects in the local space.

(3) MesRemoteEvent

This class defines a packet that reports events via Event Channel of CORBA. In terms of IDL mapping, this class is converted to struct. Since Event Channel accepts struct mapping only, MesEvent needs to be converted to this class before transmission.

(4) MesUnifiedEventDispatcher

This class distributes all events via a common interface.

- Register an event listener
- Distribute events (MesRemoteEvents).
- Delete a registered event listener

(5) MesUnifiedEventListener

This interface defines a listener that receives events from a CORBA server.

3.10.2. Remote entity component

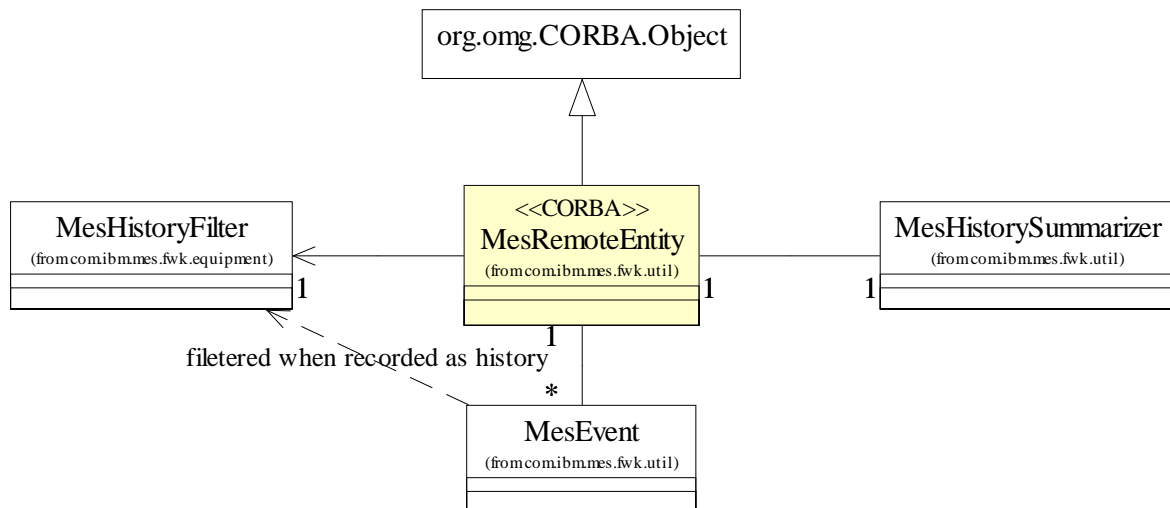


Figure 42 Class diagram of Remote Entity Component

(1) MesRemoteEntity

This interface defines the common functions of remote objects that comprise a framework. Various attributes acquired through this interface are copies of server internal attributes. Edits performed on acquired attributes will not be reflected in the attributes in a server.

(2) MesHistoryElement

This class defines the history items to be recorded within the component.

(3) MesHistoryFilter

This class defines the filter for checking whether or not to make history recordings.

(4) MesHistorySummarizer

This is an abstract class for making a summary of the history of a remote entity (MesRemoteEntity).

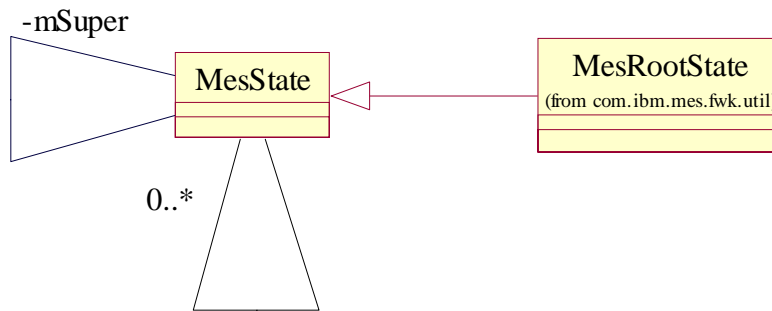


Figure 43 Class diagram of State Component

3.10.3. State management component

(1) MesState

A state object (MesState) retains the aggregate of states, which are hierarchically arranged in the form of a tree structure. It offers the following functions.

- Get the ID of a state
- Get the level of a state
- Set or get the name of a state
- Set or get the state of a high-level
- Set or get the state of a low level
- check whether a given state coincides with this state
- Get the information that indicates whether a low-level state is registered
- Delete a given low-level state together with all of its subordinate lower-states
- If a given state ID is registered as this state or a lower-level state, returns that state.
- If a given state ID is registered as this state or a higher-level state, returns that state.

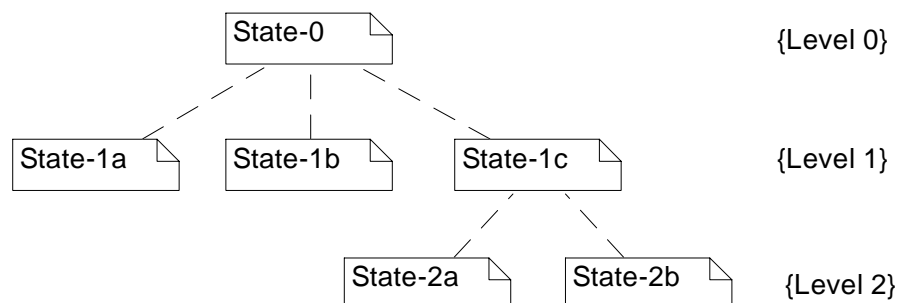
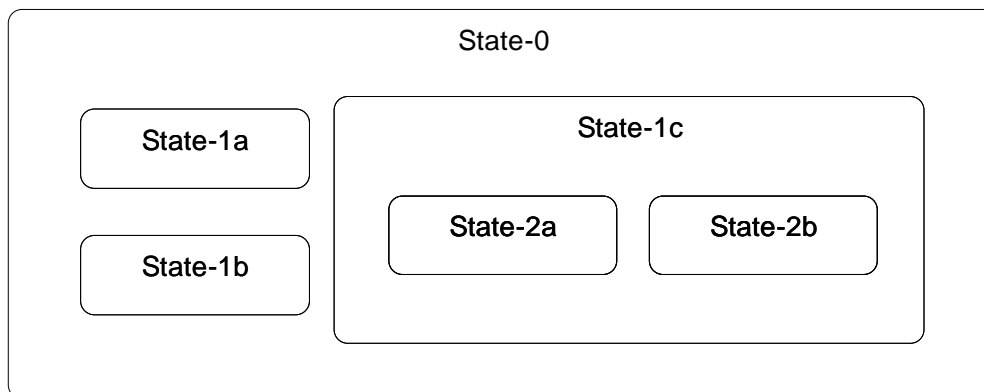


Figure 44 State diagram of a state representation

(2) MesRootState

A root state (MesRootState) is the highest-level state. It serves as a root for the aggregate of all states, which are hierarchically arranged in the form of a tree structure. This class is derived from a MesState class, which corresponds to an individual state. However, it can retain the state prevailing at that time. Further, a state belonging to this state aggregate is registered by addState (when it belongs to a level immediately below the highest level) or addStateTo (when a lower-level state below a registered state is to be registered). As regards the states registered by these methods, their uniqueness is assured within the state aggregate. An entity class (production order, production lot, etc.) that may be in various states is associated with a corresponding state aggregate. To such an entity class, public methods such as isInState, getCurrentStateId, or getRootState are offered. isInState replies to indicate whether the object is in a given state or its higher-level state. getCurrentStateId returns the prevailing state ID of the object. getRootState returns a root state object (MesRootState), which serves as the root of a tree structure with which all the elements of the state aggregate are associated, or a class derived from it (e.g., MesLotJobState). When you note a tree structure obtained by getRootState, you can reference the state names (e.g., names used for screen display) that correspond to all state IDs contained in the state aggregate.

-End of this document-