Journal of Southeast University (English Edition) Vol.19 No.1 Mar. 2003 ISSN 1003—7985

Development of a distributed and integratable manufacturing execution system framework

Yang Hao[1] Zhou Na[1] Zhu Jianying[1] Luo Xiang[2]

([1]The Mech-Electronic Engineering Research Center, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

([2]Department of Mechanical Engineering, Southeast University, Nanjing 210096, China)

Abstract: Using remote method invocation (RMI) and a distributed object-oriented technique, this paper presents a systematic approach to developing a manufacturing execution system (MES) framework, which is open, modularized, distributed, configurable, interoperable and maintainable. Moreover, the design patterns for the framework are developed and a variety of functional components are designed by inheriting appropriate patterns. And then an application is constructed by invoking corresponding methods of related components. An MES system implementing the framework and design patterns can be facilely integrated with other manufacturing applications, such as enterprise resource planning (ERP) and floor control system (FCS).

Key words: manufacturing execution system framework; design pattern; distributed object-oriented; remote method invocation

With greater demand for high quality, low cost, high performance products, manufacturers have to reduce costs, cut production time, improve quality, increase asset utilization, decrease inventory and guarantee on-time delivery of their products. To achieve these goals, manufacturing execution systems (MES) emerged in the 1990's.

The MESA international definition of MES is, "Manufacturing execution systems (MES) deliver information that enables the optimization of production activities from order launch to finished goods. Using current and accurate date, MES guides, initiates, responds to, and reports on plant activities as they occur. The resulting rapid response to changing conditions, coupled with a focus on reducing non value-added activities, drives effective plant operations and processes. MES improves the return on operational assets as well as on-time delivery, inventory turns, gross margin, and cash flow performance. MES provides mission-critical information about production activities across the enterprise and supply chain via bi-directional communications."[1]

Many industries currently use an MES for managing factory floor information and activities to increase productivity and improve quality for the advantages of MES systems[2]. But the MES system is very complex; it's not that the code is necessarily complex, but rather that the whole system itself is. Therefore, most of the MES systems are monolithic, insufficiently configurable and integratable, and difficult to modify and expand.

Using remote method invocation (RMI), a distributed object-oriented technique, this paper presents a systematic approach to developing an MES framework, which is open, modularized, distributed, configurable, interoperable and maintainable. Moreover, the design patterns for the framework are developed and a variety of functional components are designed by inheriting appropriate patterns. And then an application is constructed by invoking corresponding methods of related components. An MES system implementing the framework and design patterns can be facilely integrated with other manufacturing applications, such as enterprise resource planning (ERP) and floor control system (FCS).

1 Basic Foundations

The design of the MES framework is based on a number of foundations. These foundations are used to develop the overall architecture. This paper introduces the most important foundations as follows.

1.1 Distributed objects and RMI

Distributed objects are objects in the client/server system. Distributed objects are packed as independent pieces of code that can be accessed by remote clients via method invocations. Clients do not need to know where the distributed object resides or what operating system it executes on; it can be the same machine or on a machine that sites across a network.

Received 2002-10-14.

Foundation item: The National Natural Science Foundation of China (59990470).

Biographies: Yang Hao (1975—), male, graduate; Zhu Jianying (corresponding author), male, professor, zjyao@nuaa.edu.cn.

Distributed objects have the inherent potential to allow granular components of software to plug-and-play, interoperate across a network, run on different platforms, coexist with legacy applications through object wrappers, roam on networks, and manage themselves and the resources they control[3].

To create a distributed object-oriented infrastructure, there are three standards: OMG's common object request broker architecture (CORBA), Microsoft's distributed component object model (DCOM), and Sun's Java remote method invocation (RMI)[4]. RMI is the latest distributed computing technology broadly used in distributed systems. In this paper, the RMI is adopted to develop the infrastructure of the MES framework. And the unified modeling language (UML) model is used to express the MES Framework.

Java RMI technology, which is the part of the core Java platform and is therefore everywhere, is the basis of distributed computing in the Java environment. Java RMI is created after broad acceptance of the Internet and object-oriented design. RMI enables applications running in different processes on different machines to communicate with one another in a way that preserves the object-oriented paradigm; thus it provides a dynamic and flexible environment for building robust distributed applications[5,6].

As Fig.1 illustrates, RMI architecture consists of three layers: the stub/skeleton layer, the remote reference layer and the transport layer. The boundary at each layer is defined by a specific interface and protocol; each layer is independent of the next and can be replaced by an alternate implementation without affecting the other layers in the system. The application layer sits on top of the RMI system. A remote method invocation from a client to a remote server object travels through the layers of the RMI system to the client-side transport, then up through the server-side transport to the server. A client invoking a method on a remote server object actually uses a stub

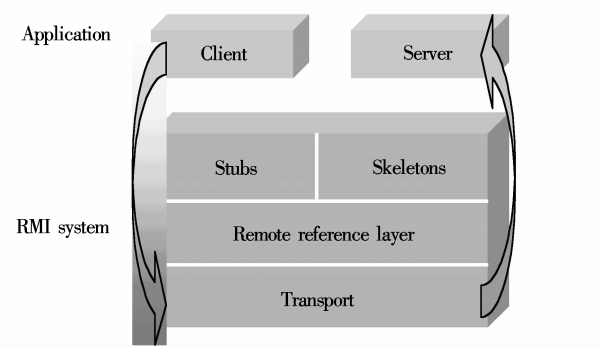

Fig.1 Java RMI system architecture

or proxy for the remote object as a conduit to the remote object. The remote reference layer is responsible for carrying out the semantics of the invocation, and the transport layer is responsible for connection setup, connection management, and keeping track of remote objects residing in the transport's address space[5].

Fig.2 illustrates the interaction of the RMI client and server[7]: ① The RMI server creates an instance of the remote object implementation and passes a serialized form of its stub to the RMI registry with which the stub is registered; ② The RMI client attempts to get a handle on the remote object from the registry; ③ The RMI registry returns a serialized copy of the stub to the client that sequentially de-serializes the stub to create an instance; ④ The client calls one of the remote object's methods through the stub; ⑤ The stub contacts the skeleton within the server; ⑥ The skeleton invokes the method on the implementation of the remote object; ⑦ The remote object implementation returns the result to the skeleton; ⑧ The skeleton returns the result to the client stub; ⑨ The client stub returns the result to the RMI client.

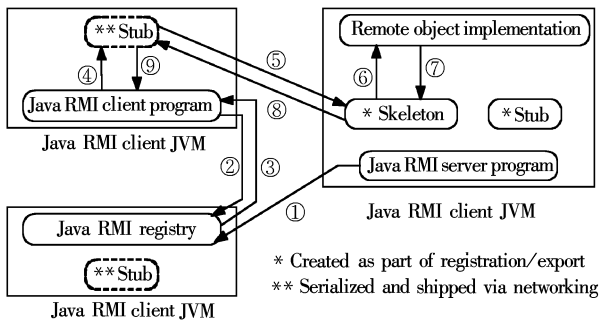

Fig.2 Java RMI client, server, and registry interaction

1.2 The concepts of frameworks and design patterns

A framework provides an organized environment for running a collection of objects. It also provides tools that let you construct components that are willing to play by the framework's rule of engagement. The net effect is to amplify the benefits obtained at the module, application and system level as well[8].

Frameworks can offer simple patterns that guide the collaboration of objects so that they can start to model their real world. The framework objects collaborating paradigm often reflects a design pattern that leverages reusable object-oriented software. Design patterns are descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context. One person's pattern can be another person's building block[7,8].

There are many differences between framework and design patterns as well as relationships; the main aspects are as follows[8,9]:

- A framework can contain several design patterns but the reverse is never true.
- A framework can be embodied in code and used directly. In contrast, design patterns, just examples of code, must be implemented each time when they are used.
- A framework always has a particular application domain. In contrast, design patterns can be used in any kind of application.
- Frameworks are what we need to get to business objects and more intelligent components. Design patterns can be used to document certain elements of the framework design.

Manufacturers are shifting to framework-based MES systems architectures as distributed applications become mainstream. These architectures offer manufacturers improved connectivity, extensibility and scalability of their MES components, faster IT development times by reusing application components, and a way to tie disparate heterogeneous environments together using the power of the Web.

2 Development of the MES Framework

2.1 Constructing the abstract object model from the real systems

The development of the MES framework begins with collecting the domain requirements to construct the abstract object model of the real systems. Considering the interface between MES and ERP and that between MES and FCS, a typical distributed architecture for a manufacturing entity contains the following requirements[2,8,10,11]:

1) Common database: stores customer orders, job assignments, equipment status, recipes, bills of materials and engineering data, human resource (HR) data and other resource data.

2) System manager: monitors and controls the status of the whole factory.

3) Scheduler: is in charge of scheduling and dispatching job assignments.

4) Equipment manager: monitors and controls equipment.

5) Material manager: handles the movement of AGVs, AS/RS, robots and other material.

6) WIP manager: tracks the work-in-process (WIP).

7) SPC manager: handles the statistical process control (SPC).

According to the above requirements, the abstract object model of the real system is constructed, as shown in Fig.3.

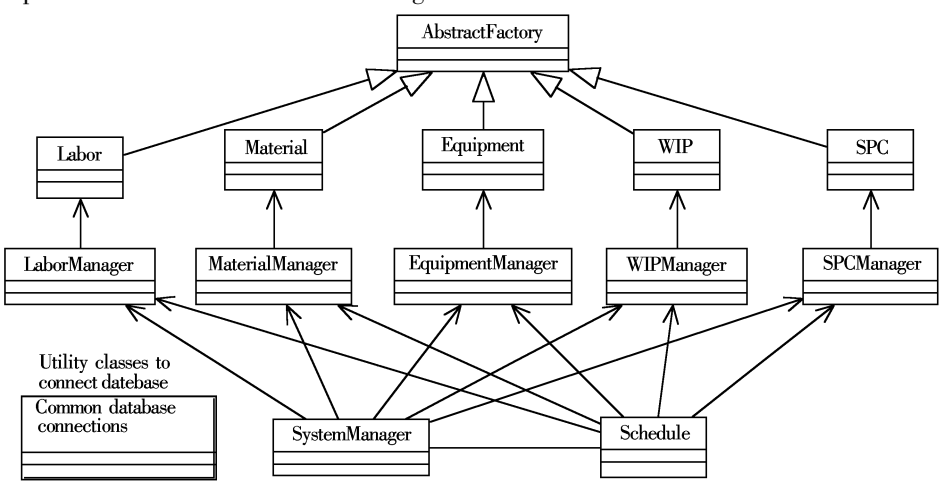

Fig.3 Abstract object model for the MES

2.2 Partitioning the abstract objects into components

The abstract objects should be partitioned systematically and methodically into components to design an integratable MES that is highly distributed. The system is partitioned into eight components according to the principle that different functional modules will be grouped into different components. They are: system management, schedule, equipment management, material management, labor management, WIP management, SPC management and common database. Due to space limitations, this paper doesn't discuss the WIP and SPC management components. The other six components are depicted in Fig.4(a). Labor, equipment, and material, three key elements of a factory, are similar and can be defined as system resources. Their management components have the same design pattern shown in the latter subsection. Therefore, these three components are also considered

as resource management components, as shown in Fig. 4(b).

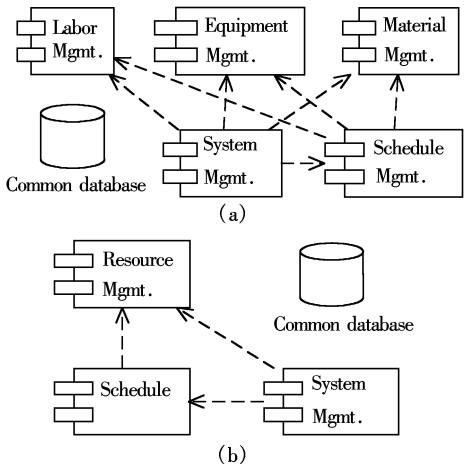

Fig.4 Partitioning abstract objects into components

2.3 Defining the MES framework messages

The MES framework messages are defined as in Fig.5 according to the principle that only framework messages are allowed to pass into and out of the MES framework components. The resource management components will accept Initiate, StartUp, ShutDown and StandBy messages from the system manager and reply EventReport and AlarmErrorReport messages; the scheduler will accept DispatchOrder and CancelOrder from the system manager and reply OrderDoneReport to it; at the same time, the schedule will send DispatchJob-CancelJob to resource management components and accept JobDoneReport message from it; the system manager also provides a CreateOrder message for the outside world, as shown in Fig.5(a).

The messages among the resource components are depicted in Fig.5(b). There are various requestresponse messages which are sent and replied to each of the components. The common database component provides the data processing messages, such as storing, updating and getting data, as shown in Fig.5(c).

Through these messages, the MES system can monitor the product activities, and transfer or feed back the operational information rapidly. After the managers/users receive this information, they can make a rapid response, such as adjustments, new decisions, and so on, to various changing conditions.

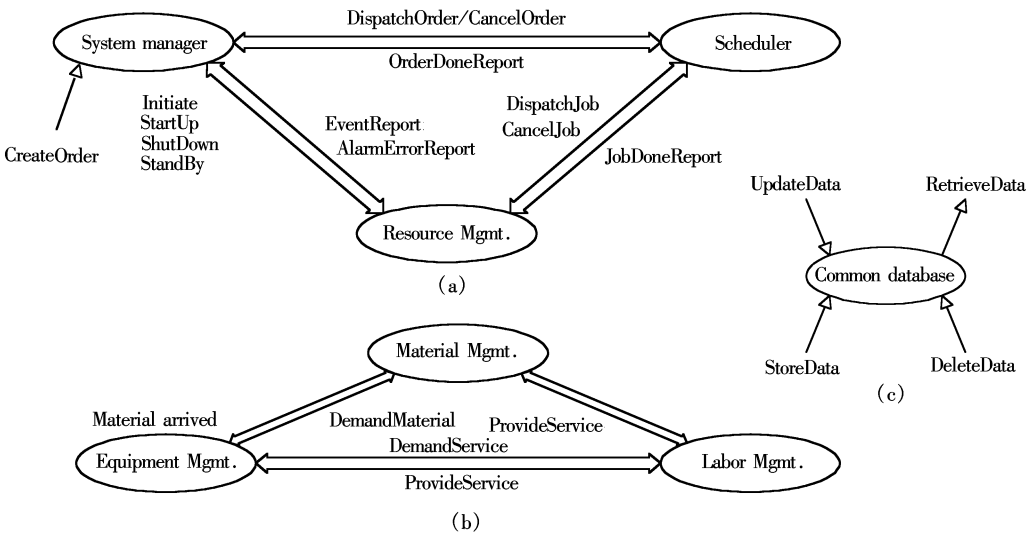

Fig.5 Defining the MES framework messages

2.4 Developing design patterns for the MES framework

The fundamental structure of the MES framework is that the system manager controls several component managers and a component manager manages several resources. Therefore, there is a design pattern for the system manager and resource managements. The design pattern of the scheduler is a different one. Due to space limitations, this paper just describes the former with UML[12], as shown in Fig.6.

The MESAbstractObject is the abstract superclass of the MESManager and MESResource. It specifies the common attributes and operations for these two subclasses to inherit. The MESManager and MESResource inherit all members of the superclass and extend their own attributes and operations. The MESManager manages several MESResource instances by MESResourceCollection class. Similarly, the MESComponentManager, a subclass of MESManager, processes the jobs of an MES Resource by MESJobHandler.

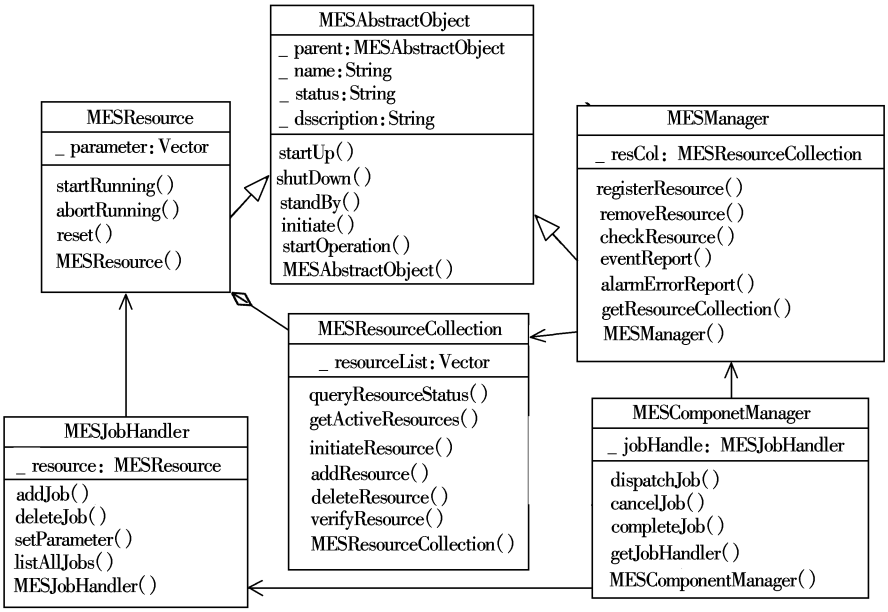

Fig.6 A design pattern for the MES framework

2.5 Developing the MES framework architecture

After all of the components and design patterns have been developed, the backbone of the MES framework is established, as shown in Fig.7.

The presentation tier of the MES framework arch-

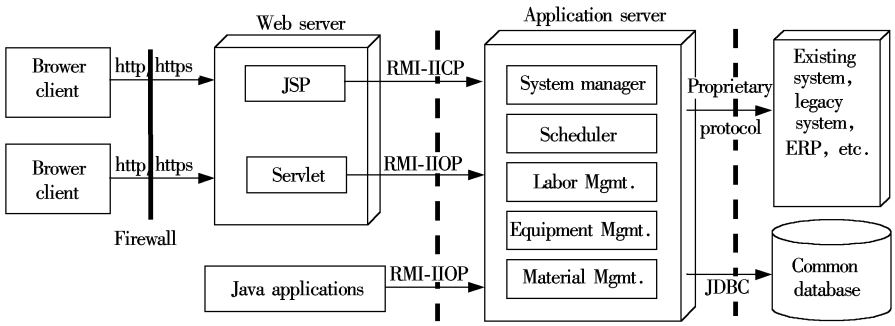

Fig.7 MES framework architecture

itecture can be an HTML or Java applet browser client that connects to a web server by http/https protocol. The web server communicates with the application server by RMI or RMI-IIOP standard protocol described in subsection 1.1. The client can also use a Java application that communicates directly with the application server by RMI or RMI-IIOP. The application server, also called the middleware tier or logic business tier, is the core of the MES framework architecture. It can access the database by JDBC and integrate with the legacy system, existing system, and ERP system using proprietary protocols.

This MES framework is a distributed, integratable, and open architecture. And it is expandable. When a new component needs to be added this system, one may simply follow the component design patterns mentioned in the above subsections and he will have a highly pluggable component. Thus the component will easily be deployed into the application server.

3 Application Example

Using the MES framework architecture mentioned above, we rapidly developed a web based distributed and integratable MES system in 8 months (usually, to develop and deploy an MES system requires 12 – 24 months) to exchange information and manage transactions in the plant. This system uses Tomcat 4.0 as the web server, Websphere 4.0 as the application server, and Oracle 8i as the database. The server end operation system can be Windows NT 4.0/Windows 2000 Server or an other server operation system; and the client end operation system can be Windows 98/2000 or other client operation systems. This MES system focuses on quick response to changing conditions, improved communication capability with ERP system, monitor production to control operations

within desired performance parameters, and a decrease in the gap between ERP and FCS. It dramatically extremely increases the productivity, improves the quality, reduces the cost, and guarantees on time delivery.

4 Conclusion

Applying the distributed object-oriented technique, an approach to develop an open and integratable MES framework based on RMI industry standard was proposed. Firstly the objects of the real system were abstracted; then these objects were partitioned into components, and their messages and design patterns were defined; finally, the MES framework architecture was developed from these fundamentals. Following the design patterns, a new component is integratable into the MES framework in a plug-and-play fashion.

From the description and application example, it is believed that the proposed approach to develop an open and integratable MES is indeed a viable and efficient method.

References

[1] MESA International. MES explained: a high level vision [EB/OL]. http://www.mesa.org/whitepapers/pap6.pdf. 1997 - 9/2002 - 10 - 8.

[2] Feng S C. Manufacturing planning and execution software interfaces [J]. *Journal of Manufacturing Systems*, 2000, **19**(1): 1 - 17.

[3] Orfali R, Harkey D, Edwards T. *The essential distributed objects survival guide* [M]. New York, NY: John Wiley, 1996. 1 - 42, 220 - 342.

[4] Campbell A T, Coulson G, Kounavis M E. Managing complexity middleware explained [J]. *IEEE IT Professional*, 1999, **1**(5): 22 - 28.

[5] Sun Microsystems, Inc. Java remote method invocation — distributed computing for Java [EB/OL]. http://java.sun.com/marketing/collateral/javarmi.html. 2001 - 3 - 11/2002 - 10 - 9.

[6] Cattell R, Inscore J. *J2EE™ technology in practice — building business applications with the Java™ 2 platform, enterprise edition* [M]. Boston Massachusetts: Addision-Wesley Publishing Company, 2001. 1 - 30.

[7] Sun Microsystems. *Java™ programming language workshop SL-285 Instructor guide* [M]. California, CA: Sun Microsystems, Inc, 2001. 1 - 22.

[8] Cheng F, Shen E, Deng J Y, et al. Development of a system framework for the computer integrated manufacturing execution system [J]. *I NT J Computer Integrated Manufacturing*, 1999, **12**(5): 384 - 402.

[9] Gamma E, Helm R. *Design patterns: elements of reusable object-oriented software* [M]. Boston Massachusetts: Addision-Wesley Publishing Company, 1995. 1 - 29.

[10] MESA International. The controls layer: controls definition & MES to controls data flow possibilities [EB/OL]. http://www.mesa.org/whitepapers/pap3.pdf. 2000 - 2/2002 - 10 - 9.

[11] MESA international. MES functionalities & MRP to MES data flow possibilities [EB/OL]. http://www.mesa.org/whitepapers/pap2.pdf. 1997 - 3/2002 - 10 - 10.

[12] Boggs W, Boggs M. *Mastering UML with rational rose* [M]. Alameda, CA: SYBEX Inc, 1999. 13 - 30.

一种分布式、可集成的制造执行系统框架的开发

杨 浩[1] 周 娜[1] 朱剑英[1] 罗 翔[2]

([1] 南京航空航天大学机械电子工程研究所, 南京 210016)

([2] 东南大学机械工程系, 南京 210096)

摘 要 采用面向分布式对象的远程方法调用 (RMI)技术,提出了一种开发开放式、模块化、分布式、可配置、可集成和可维护的制造执行系统 (MES)框架的系统方法.同时,为 MES 框架开发了相应的设计模式及其组件,根据这些设计模式并调用相应组件的方法就可以生成应用程序.由此生成的 MES 系统将能够很容易地与 ERP,FCS 等其他制造系统相集成.

关键词 制造执行系统框架; 设计模式; 面向分布式对象; 远程方法调用

中图分类号 TH166; TP393