The Evolution Of Prototype Architectures Developed For The Scheduling Software Integration Project

Frank Riddick
Computer Scientist
Manufacturing Engineering Laboratory
National Institute of Standards and Technology
Gaithersburg, MD USA

Christophe LeCapitaine
Guest Researcher
Manufacturing Engineering Laboratory
National Institute of Standards and Technology
Gaithersburg, MD USA

Abstract

The National Institute of Standards and Technology (NIST) is collaborating with vendors, users, and university researchers to develop a virtual manufacturing testbed. This testbed contains a number of software research prototypes, which address both optimization and integration issues for a wide range of software applications, including scheduling. This report describes the evolution and current state of a prototype that integrates scheduling applications with data collection systems.

1 Introduction

Manufacturing plays a vital role in the world's economy. In the United States alone, there are tens of thousands of factories producing metal-fabricated parts. These factories employ millions of people and ship billions of dollars worth of products every year [17]. One of the daily problems faced by the managers of these factories is scheduling. Discrete-event, simulation-based scheduling packages are used widely by factory managers around the world. These packages include a large number of pre-defined dispatching rules that will produce schedules for both simple and complex production environments. Users can extend these canned rules by adding their own factory-specific rules.

The effectiveness of these packages is limited in two ways. First, the dispatching rules are only guaranteed to produce feasible schedules. Typically, the notion of optimality, with respect to one or more performance measures, is not even considered. In fact, the user must supply all knowledge about which rule(s) to use to achieve a desired performance measure(s). Currently, this knowledge can only be derived from extensive experimentation with combinations of the pre-defined rules provided by packages. Second, it is difficult to integrate these packages with current shop floor data collection packages. This means that an extensive data translation effort is necessary to import current information into the scheduling application. Such efforts are costly, error-prone, and time consuming.

The National Institute of Standards and Technology (NIST) is collaborating with vendors, users, and university researchers to develop a virtual manufacturing testbed. This testbed contains a number of software prototypes that address both optimization and integration issues for a wide range of software applications, including scheduling [1]. The approach used in and the status of the optimization work related to scheduling can be found in [2] and [3] respectively.

This report describes the research that has led to the current scheduling prototype. It is organized as follows. Section 2 provides some background information. Section 3 describes some initial implementations done by NIST and other project participants. Section 4 describes the architecture of the current prototype. Section 5 summarizes the work and discusses possible future directions for the research.

2 Background

NIST has been involved in manufacturing integration research for many years. This research began in the early 1980s, when automated manufacturing equipment and computers were first introduced into flexible manufacturing systems. The impetus for NIST's research program was the need for a generic factory control model upon which to base the integration of computers and equipment. NIST developed a hierarchical control model [4] and implemented that model in the Automated Manufacturing Research Facility (AMRF) [5]. Initial efforts to integrate scheduling into that model are described in [6,7].

The physical layout of the AMRF consisted of several robot-tended machining stations, a cleaning and deburring station, an inspection station, and a material handling system. To control the flow of jobs and materials through the facility, NIST developed a factory control system [4], which was supported by a distributed database management system and a network communications system [8]. The AMRF incorporated commercially available manufacturing equipment and computer hardware from many vendors, but much of the software was developed internally. The intention was that substitutions of commercial software would occur whenever it became available. Although it was highly innovative and immensely successful, the AMRF was subject to the same kinds of equipment problems faced by factories across the country. These problems increased both the time and the cost of integration testing.

To continue research in the area of manufacturing system integration, in 1994 NIST embarked on a new program called SIMA, Systems Integration for Manufacturing Applications [9-11]. SIMA is building upon the experience gained in the AMRF to accelerate the application of information technology to manufacturing environments. The SIMA Program works with U.S. industry to:

- develop standards for information exchange and interface protocols addressing interoperability problems in manufacturing systems.
- provide online access to NIST-resident capabilities supporting manufacturing technologies.
- develop collaboration technologies enabling industry researchers, practitioners, and NIST staff to remotely work together.

These efforts will allow manufacturing industries to make use of computer networks as a mechanism for communicating product and process data among various manufacturing activities such as product/process design, analysis, planning, scheduling, production, and quality control. The development of information interfaces between different manufacturing applications and between applications and their users will improve integration and thereby usability of these systems.

3 Scheduling Integration Project

During the last 10 years, many manufacturing companies have invested heavily in computer hardware and software that make it possible to collect data about the events on the shop floor literally as they are happening. This real-time data collection makes it possible to “close the loop” for operations management just as concurrent engineering is “closing the loop” for product design. The integration of computer software that collects real-time data with the computer software that uses the collected data continues to be a major problem. While there is a general understanding about which tools need the collected information, there is no agreement on the content or format of that information.

A two-year project was created within SIMA and jointly funded by the Navy Manufacturing Technology (MANTECH) program to develop generic interface specifications for, and to demonstrate the integration of, scheduling and shop floor data collection tools. In this project, shop floor operations will be demonstrated through the use of a discrete-event simulation of a job shop. This differs from the approach used in the AMRF of creating a real job shop as a part of the integration project. This project is a collaborative effort between vendors, users, academia, and NIST. It has two major benefits. First, the interface specifications will reduce the cost of integration for both manufacturers and vendors. Second, the integrated prototype will allow manufacturers, vendors, and academic researchers the opportunity to test the capabilities of a single scheduling technique on different simulated shops, and to test different scheduling techniques on the same simulated shop. Realizing these benefits has the potential to provide enhanced capabilities to manufacturers to improve shop performance and throughput.

3.1 Conceptual Model Development

Project participants met to develop a conceptual model of the factory entities necessary to integrate scheduling applications with shop floor data collection applications. The goal of this model was to specify a minimal set of the entities associated with factory operations, and not to define a comprehensive model of all factory operational information. This approach was required partly by the limited reporting capabilities of commercial-off-the-shelf (COTS) shop floor data collection systems and partly because

much of the information that is needed for scheduling does not need to be communicated to other factory applications.

The project team decided that a significant level of integration could be achieved through: (1) the definition of entities to represent elements of the factory (Loads, Resources, Buffers, and Materials); and (2) the definition of status message entities that define requests to create, change, or delete instances of the factory entities. Loads are groups of parts that are being operated upon by the Resources of the factory. Resources are the operators, machines, tools, and fixtures used in production to change the parts contained in the Loads into finished products. Buffers are temporary holding or staging areas for Loads. Materials represent quantities of consumable items that are tracked by the level available for production. Twelve status message entities, such as CreateLoad and ChangeResource, were defined to specify that instances of the four factory entities could be created, changed, or deleted. Additional information concerning these models can be found in [12].

3.2 Initial Prototype Development Efforts

Three prototype implementations were done by project participants, two by vendors and one by NIST personnel. The factory entities and status messages defined in [12] were used as a basis for the architectures for the prototypes. The use of COTS software in developing the prototypes was encouraged rather than developing the applications completely from scratch. Each prototype provided functional components for scheduling, dispatching, and data collection/reporting. Some mechanism for persistent storage of status information was also required so that successive executions of the scheduling application could use current shop floor information as the basis for creating new schedules. In addition, functionality to simulate the execution of the schedule information by the job shop was required. Finally, consideration was given to providing mechanisms for exchanging the factory information with other manufacturing systems, such as enterprise resource planning (ERP) systems. In the following section, the three implementations will be examined.

3.2.1 Vendor 1 Prototype Architecture

Figure 1 shows the architecture of the vendor 1 prototype. In this prototype, vendor 1 used its COTS scheduling application to provide the scheduling functionality. When schedules are produced, a file passing mechanism is used to pass the information to a commercial manufacturing execution system (MES) that provides the job dispatching functionality for the shop floor. The MES system also provides

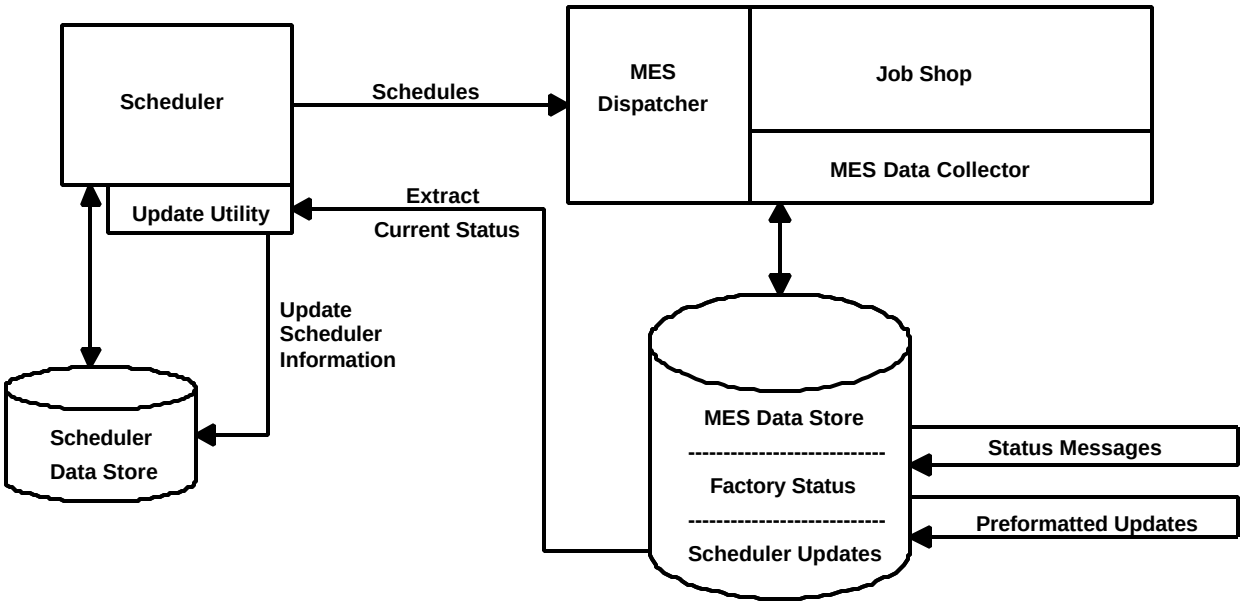


Figure 1 - Vendor 1 Prototype Architecture

the data collection functionality. The functionality of the job shop is simulated by the execution of a small set of canned routines that translate the manufacturing instructions (work orders) produced by the dispatcher into the events that would occur from the execution of those instructions. The MES Data Collector stores these events in the MES system database. The MES system database has been extended with tables to hold factory status information. Database triggers are employed to automatically translate MES data updates into status messages that cause the factory status extensions to the MES database to be updated. In addition, when any of the tables of the factory status extensions are updated, database triggers fire that create preformatted schedule update instructions in another extension to the MES database. These schedule update instructions are accessed by the update utility, an extension to the scheduling application, which uses them to update the scheduling application's database with current information about the status of the shop floor. The update utility automatically runs to update the database before the creation of new schedules by the scheduling application.

The key feature of this architectural approach is its extensive use of COTS software and its leveraging of existing database technologies. A drawback with this approach is the factory status information is somewhat hidden as an extension of the MES system's database, which makes it harder to integrate with other manufacturing applications.

3.2.2 Vendor 2 Prototype Architecture

Figure 2 shows the architecture of the vendor 2 prototype. In this prototype, vendor 2 used its COTS scheduling application to provide the scheduling functionality. When schedules are produced, a file passing mechanism is used to pass the information to a simple dispatching program. The dispatching program creates work orders for each machine resource in the job shop. The functionality of the job shop is simulated by the execution of a small set of canned routines that translate the work orders produced by the dispatcher into the events that would occur from the execution of those work orders. These events are passed to a COTS data collection system that converts the information into status messages and stores the status messages in a file as ASCII strings. This file of status messages is read by the update utility, an extension to this vendor's scheduling application. In this prototype, the update utility uses the file of status messages to update both the scheduling application's database and a separate relational database that holds the factory status information in accordance with the conceptual model described above. The update utility automatically runs to update the database before the creation of new schedules by the scheduling application.

This architecture also makes extensive use of COTS software and leverages existing database

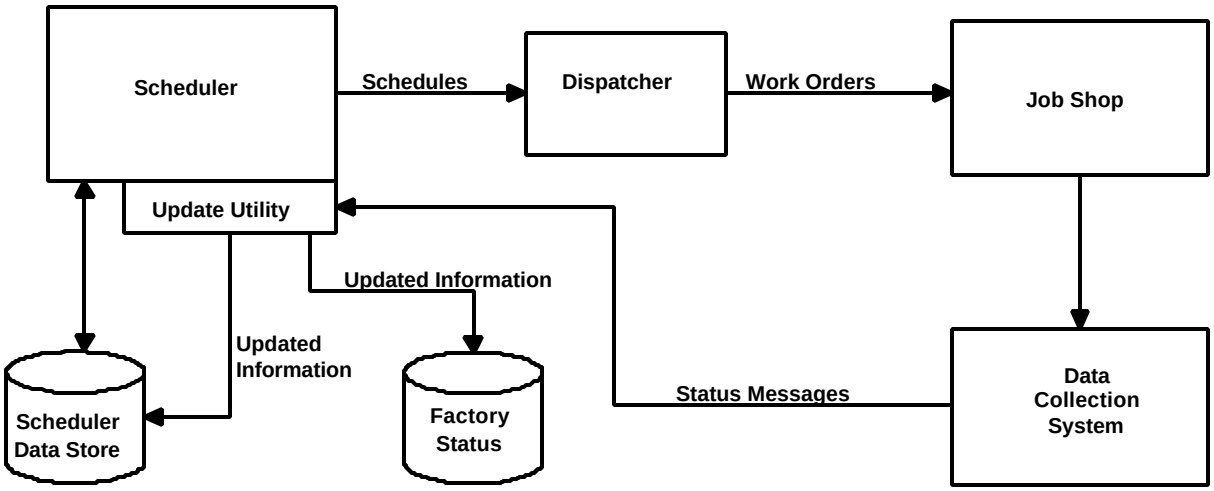


Figure 2 - Vendor 2 Prototype Architecture

technologies. It overcomes a drawback of the vendor 1 architecture by having a more open, modular approach that is conducive to integration with other manufacturing applications.

3.2.3 Initial NIST Prototype Architecture

Figure 3 shows the architecture of the initial NIST prototype. NIST levied upon itself the additional requirement of being able to work with multiple scheduling applications. So, in this prototype, the scheduling functionality can be provided by either vendor's COTS scheduling application. When schedules are produced, a file passing mechanism is used to pass the information to the same dispatching program used in the vendor 2 prototype that creates work orders for each machine resource in the job shop.

Instead of using simple routines that just translate inputs into expected outputs, the NIST prototype employs a discrete-event simulation of a job shop and an integrated data collection system. A commercial discrete-event simulation engine was modified to:

- Create Loads (the groups of parts being produced via the manufacturing process) based on order-related information provided by the dispatcher.
- Use shop floor Resources to process Loads according to the work orders produced by the dispatcher.
- Produce status messages describing the events that occurred as a result of the simulated manufacturing activities.

Three different COTS discrete-event simulation systems were modified in this manner. Since two scheduling systems are also supported, six different prototype configurations can be run. Further information about the modification to the simulations can be found in [13,14].

When changes in the state of the simulated job shop occur, the simulated job shop creates status messages and stores the status messages in a file as ASCII strings. A software component called the Status Manager reads the status messages and updates a database that maintains the current factory status. An update utility is used to interact with the Status Manager to extract the current state of the job shop and use this information to update the database of the scheduling application being used in this implementation. The update utility automatically runs to update the database before the creation of new

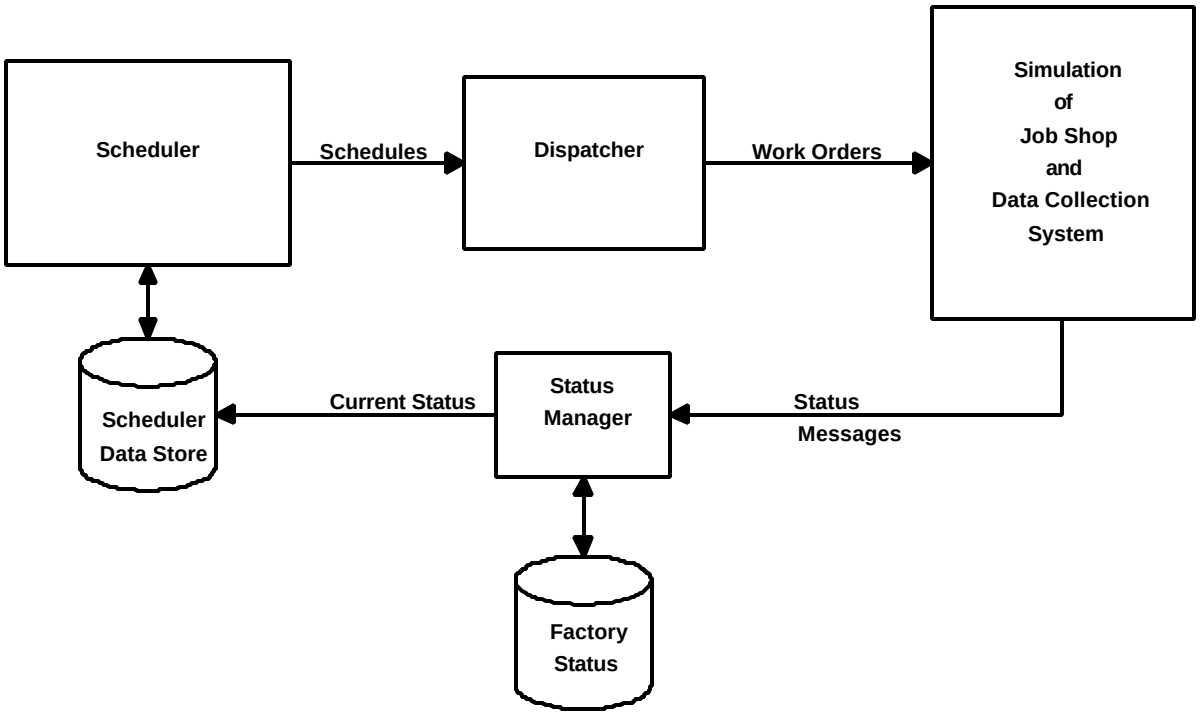


Figure 3 - Initial NIST Prototype Architecture

schedules by the scheduling application.

The motivation for this architecture is to provide a flexible, modular infrastructure that supports the integration of different scheduling and simulation applications.

3.3 Current NIST Prototype Architecture

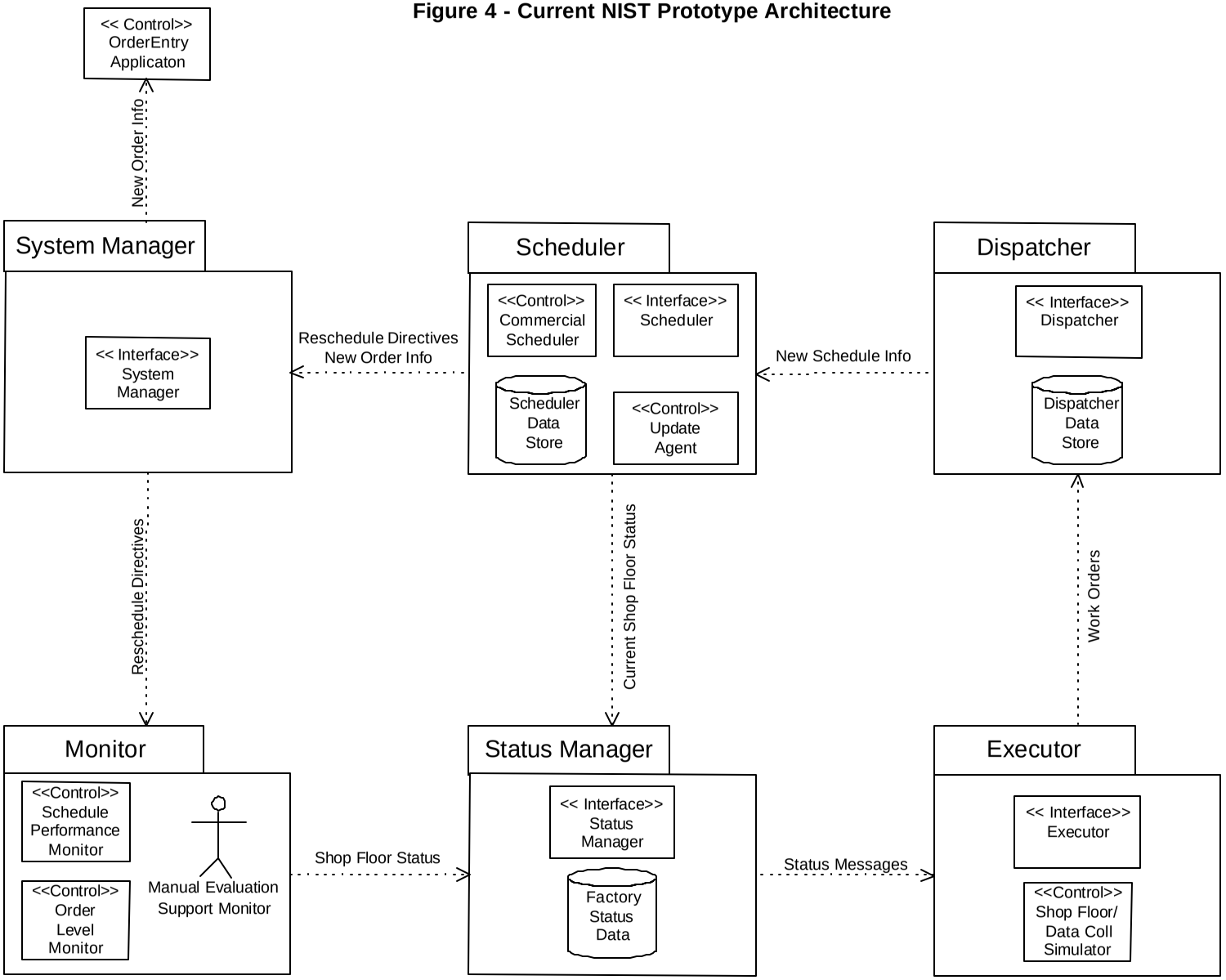
Once the three prototypes had been developed and demonstrated, their architectures were analyzed to assess the effort that would be required to integrate the systems with other manufacturing applications. The use of file passing mechanisms for information exchange and the lack of clear functional boundaries for system components were identified as common characteristics of the architectures that were possible impediments to further integration. NIST made significant modifications to its prototype architecture to mitigate these impediments to integration and to provide improved support for manufacturing system automation and performance evaluation.

Figure 4 shows the current NIST prototype architecture as a Unified Modeling Language (UML) class diagram [15]. The architecture has been divided into subsystems that represent functional components that collaborate to perform the system's work. The Scheduler, Dispatcher, and Status Manager subsystems provide capabilities similar to their like-named functional components from the initial prototype architecture. The Executor subsystem provides similar capabilities to the discrete-event simulation of the integrated shop floor/shop data collection system from the initial prototype. In the current architecture, a new subsystem called the System Manager is present. It coordinates the activities of the other subsystems and provides an interface through which external systems can interact with this system. Another new subsystem, called the Monitor, provides for the implementation of automated performance evaluation and rescheduling capabilities. Several concurrent, interacting monitoring applications can be supported by this subsystem. Each application may use different criteria for evaluating shop floor performance and automatically generate reschedule requests for processing by the System Manager. Additionally, monitoring applications can be developed that extract and present the status-related information necessary for shop floor performance evaluation by human shop floor managers. Finally, to illustrate how other applications can be integrated with the subsystems of the scheduling system, an Order Entry Application was developed. It has been included in the diagram, though such an application is logically external to the system boundary.

In the Figure 4 diagram, the type of information that is exchanged between subsystems is specified by named dependencies, which are denoted by the arrows with dotted lines between subsystems. This differs from the presentation of information exchange requirements in the previous architecture diagrams where data flow between components was emphasized. Internally, each subsystem contains one or more software objects. These objects may be: databases; applications that provide the mechanisms for implementing the requirements of the subsystem (indicated with the control stereotype); or objects which implement the interface through which other applications may interact with this subsystem (indicated with the interface stereotype). Additional information about the interface objects is presented below.

With the exception of the Monitor subsystem, each subsystem contains an interface object that is a Common Object Request Broker Architecture (CORBA) server that implements that subsystem's interface [16]. Components of other subsystems, and external applications such as the Order Entry Application, may only interact with a subsystem through the methods provided by its interface. Each subsystem's server contains the logic to collaborate with the other objects in the subsystem to provide the subsystem's required functionality. To provide their functionality, the objects in the Monitor subsystem need only be clients to the services provided by the server objects in the other subsystems. Through the extension of the architecture with the System Manager subsystem and the use of CORBA technology as the mechanism for subsystem communication, integration of the system with other manufacturing applications can be accomplished through a widely accepted integration technology. Platform and language dependencies have been reduced, and integration and implementation possibilities have been increased. In addition, the Monitor subsystem provides support for implementing system performance evaluation functionality.

Figure 4 - Current NIST Prototype Architecture



3.3.1 Interface Classes and an Interaction Diagram for the Current NIST Architecture

Figure 5 is a UML class diagram that shows the details of the classes that define the interface of each subsystem. Prototype implementations of the NIST architecture were developed using CORBA servers whose interfaces were based on these classes. Figure 6 is a UML interaction diagram that shows how the servers interact to incorporate new order information into the system by creating and implementing a new schedule.

4 Summary

During the last 10 years, many manufacturing companies have invested in modern computer-based technologies. These technologies have made it possible to gather data about the events on the shop floor as they are happening. The availability of this data has spawned a new challenge…how to distribute and analyze this information so that improved manufacturing processes can result. Towards this end, a project at NIST has focused on developing methods for integrating commercial scheduling software with commercial shop floor data collection software. Achieving this integration will require the development of new interface and information exchange standards. NIST is working with industry, vendors, and users to develop these standards. An analysis of the architectures of several prototype systems developed to test integration approaches and to foster standards development and adoption has been presented in this paper.

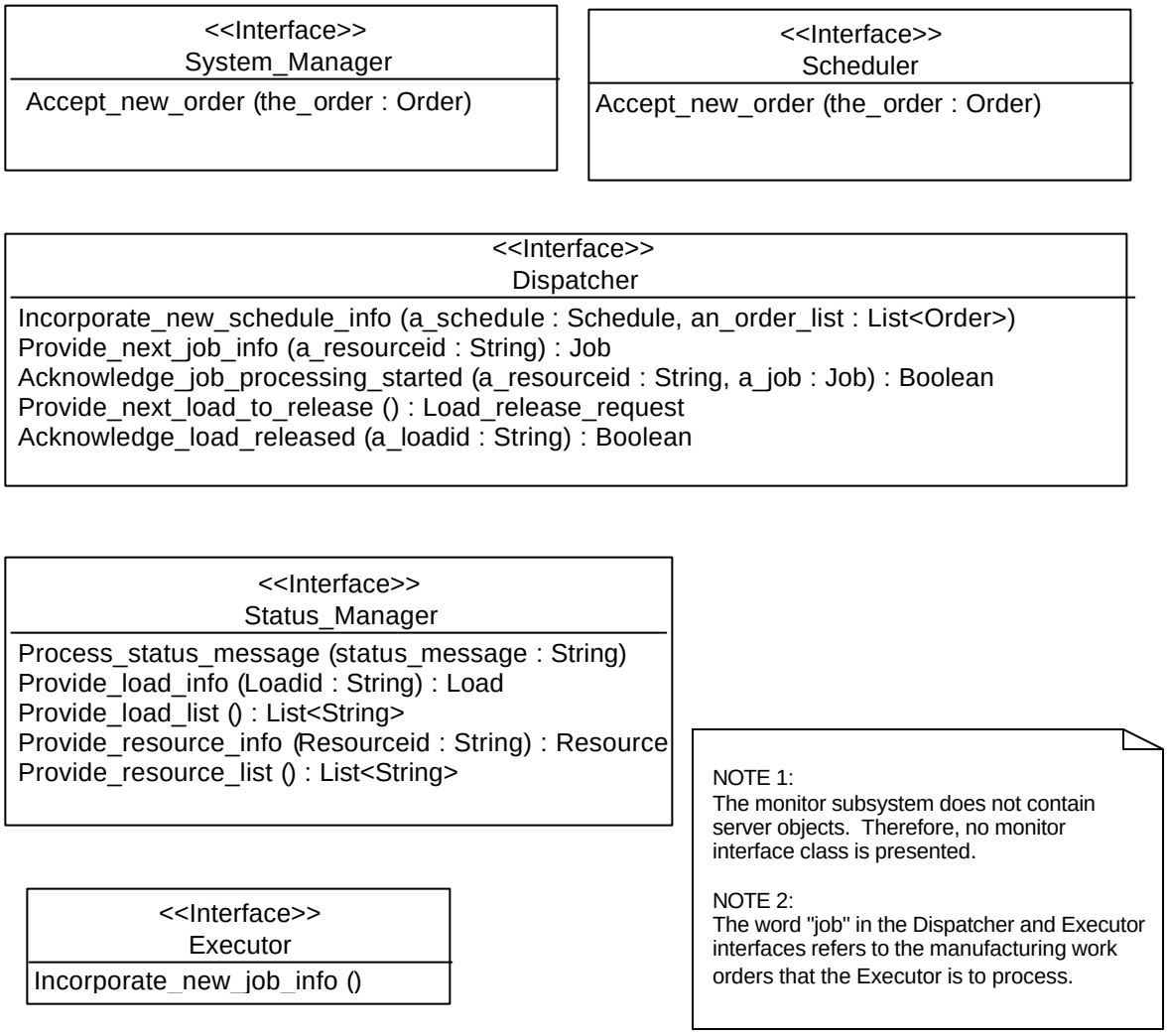


Figure 5 - Subsystem Interface Classes

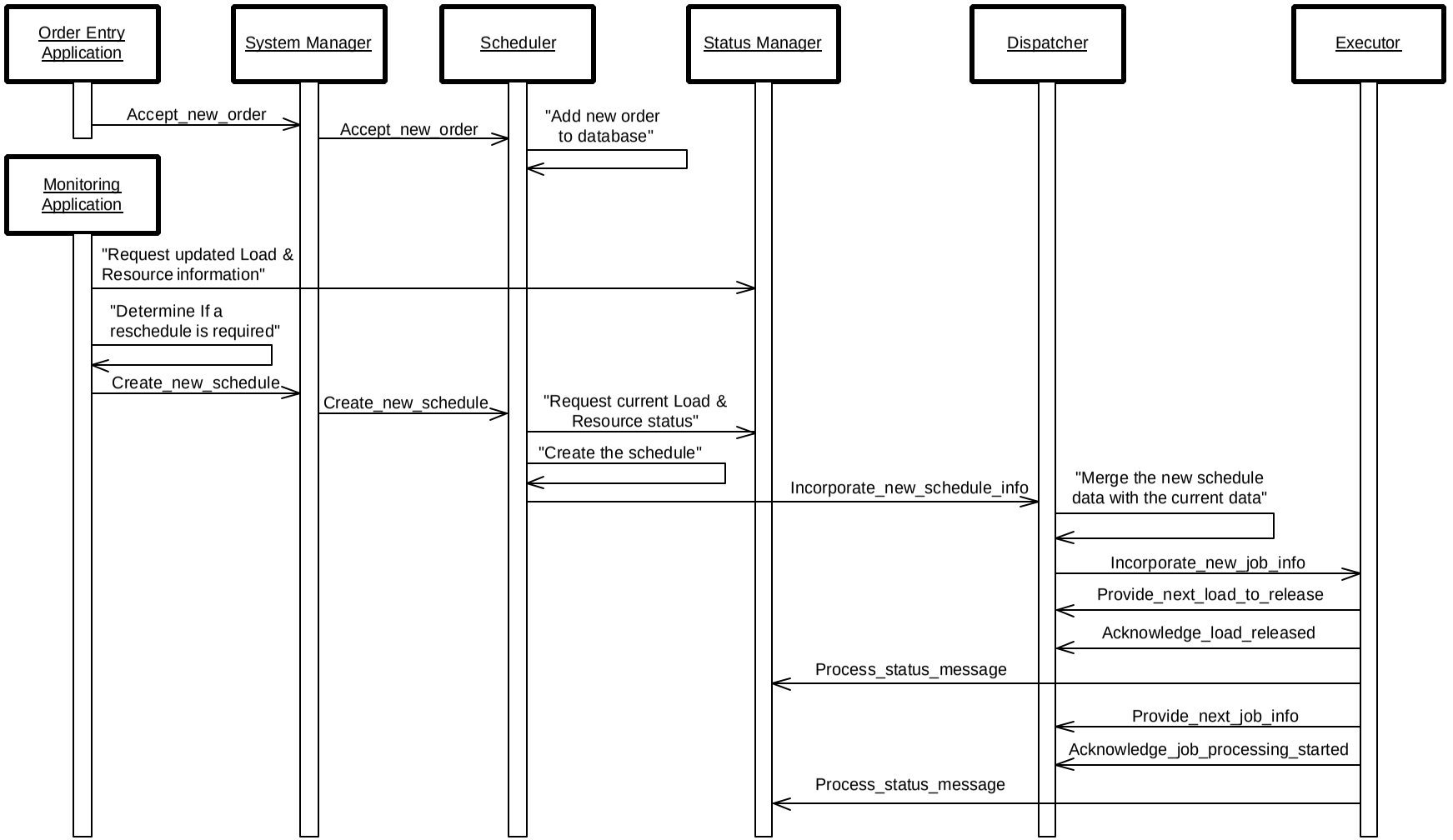


Figure 6 - Sequence diagram showing new order incorporation via Monitor-initiated new schedule generation

5 Acknowledgements

Work described in this paper was sponsored by the U.S. Navy Manufacturing Science and Technology Program and the NIST Systems Integration for Manufacturing Applications (SIMA) Program. The work described was funded by the United States Government and is not subject to copyright.

6 Disclaimer

Certain commercial equipment, instruments, or materials may be identified in this paper in order to facilitate understanding. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

7 References

[1] Jones, A. and Iuliano, M, "Virtual Manufacturing Testbed," Proceedings of the SMC'97, Vol. 1, 737-742, Orlando, Florida, October, 1997.

[2] Jones, A., Rabelo, L., and Yih, Y., "A hybrid approach for real-time sequencing and scheduling," International Journal of Computer Integrated Manufacturing, Vol. 8, no.2, 145-154, 1995.

[3] Jones, A., Riddick, F., and Rabelo, L., "Development of a Predictive-Reactive Scheduler Using Genetic Algorithms and Simulation-based Scheduling Software," Proceedings of the AMPST'96, 589-598, Bradford, England, March, 1996.

[4] Jones, A. and McLean, C., "A Proposed Hierarchical Control Model for Automated Manufacturing Systems," Journal of Manufacturing Systems, Vol. 5, No.1, 15-25, 1986.

[5] Simpson, J., Hocken, R., and Albus, J., "The Automated Manufacturing Research Facility," Journal of Manufacturing Systems, Vol. 1, No. 1, 17-31, 1982.

[6] Jackson, R. and Jones, A., "An Architecture for Decision Making in the Factory of the Future," ORSA INTERFACES - Special issue on Manufacturing, Vol. 17, No. 6, 15-28, 1987.

[7] Davis, W. and Jones, A., "A Real–Time Production Scheduler for a Stochastic Manufacturing Environment," International Journal of Computer Integrated Manufacturing, Vol. 1, No. 2, 101-112, 1988.

[8] Jones, A., Barkmeyer, E., and Davis, W., "Issues in the Design and Implementation of a System Architecture for Computer Integrated Manufacturing," International Journal of Computer Integrated Manufacturing - Special issue on CIM Architecture, Vol. 2., No. 3, 65-76. 1989.

[9] Bloom, H., "Technical Program Description: Systems Integration for Manufacturing Applications," NISTIR 5476, National Institute of Standards and Technology, Gaithersburg, MD, July, 1994.

[10] Fowler, J., "Systems Integration for Manufacturing Applications Technical Program Plan," NISTIR 5986, National Institute of Standards and Technology, Gaithersburg, MD, March, 1997.

[11] Fowler, J., "Systems Integration for Manufacturing Applications 1998 Annual Report," NISTIR 6339, National Institute of Standards and Technology, Gaithersburg, MD, April, 1999.

[12] Riddick, F. and Loreau, A., "Models for Integrating Scheduling and Shop Floor Data Collection," Proceedings of IASTED MIC'97, 276-279, Vienna, Austria, February, 1997.

[13] Riddick, F.,"Using Simulation as a Proxy for a Real Shop Floor and Data Collection System," NISTIR 6173, National Institute of Standards and Technology, Gaithersburg, MD, 1998.

[14] Riddick, F., "Reactive Scheduling System Implementations Using a Simulated Shop Floor," Proceedings of IASTED Applied Modelling and Simulation Conference, 104-109, Honolulu, Hawaii, August, 1998.

[15] Erikson, H., and Penker, M., UML Toolkit, (New York; Wiley, 1998)

[16] Object Management Group, The Common Object Request Broker: Architecture and Specification, 2.2 ed., (OMG, 1998)

[17] Jones, A. and McLean, C., "Industrial Need: Production Standards," NISTIR 6058, National Institute of Standards and Technology, Gaithersburg, MD, 1997.