CONTROLLING ACTIVITIES IN A VIRTUAL MANUFACTURING CELL

Michael Iuliano
Albert Jones

National Institute of Standards and Technology
Metrology Bldg., Room A127
Gaithersburg, MD 20899

ABSTRACT

Researchers at the National Institute of Standards and Technology are developing a virtual manufacturing cell. This cell will contain simulation models of a wide range of manufacturing equipment, processes, and systems. It will have commercial and prototype applications software which implement production functions from order release to final inspection. There will be an information base and a collection of interfaces which will provide the integrating infrastructure for these applications. These interfaces will be based on information models and exchange protocols, and will specify what information is shared across those applications and how it is exchanged. This paper describes the current virtual manufacturing cell, with special emphasis on the hierarchy we are developing to control activities within the cell.

1. INTRODUCTION

Since the mid 1980s, the National Institute of Standards and Technology (NIST) has been involved in research and development related to manufacturing systems integration. The Automated Manufacturing Research Facility (Simpson, Hocken, Albus, 1982) was designed and built as a development and demonstration testbed for a wide range of systems integration projects. Hierarchical control principles (Albus, Barbera, Nagel, 1981), together with contem- porary information management and communication strategies were used to design and build these integrated systems. The computer technology available at that time was used to implement these principles and strategies on top of a real manufacturing cell. As a result, the AMRF was subject to the same kinds of problems faced by factories across the country: difficult and time consuming development of new ideas; and, frequent breakdowns, software glitches and communications problems.

To overcome these equipment-related problems, NIST is collaborating with vendors, users, and university researchers to develop a virtual manufacturing cell. This cell will contain simulation models of manufacturing equipment, processes, and systems. These models will be used for off-line programming of individual pieces of equipment and on-line control of processes and systems. There will also be commercial and prototype applications software and a manufacturing information base. The applications software will implement production functions from order release to final production and inspection. A major effort will be the development of the interfaces between the applications and this information base. These interfaces will provide the methods for creating, updating, inserting, and extracting required information. Together, they will provide the integrating infrastructure for both the simulation models and the software applications.

This virtual manufacturing cell will provide a highly flexible and cost effective means for extending the earlier AMRF work in integration and control . We intend to investigate both traditional and emerging control strategies. We will examine a wide range of integration mechanisms including file transfer, messages passing, object request brokering, and databases . As it evolves, this virtual manufacturing cell will provide a place for testing the robustness of interface standards, and conformance of software products to those standards. It will be a vehicle for cooperative research and development on a wide range of manufacturing problems. Initially, the virtual manufacturing cell will be housed in a laboratory at NIST. Our long term goal, however, is to use the Internet and emerging web technologies to integrate models, applications, and information bases from locations around the country and the world.

This paper is organized as follows. Section 2 provides a short description of the cell including software applications and hardware platforms. Section 3 presents an overview of the proposed control hierarchy - a cell controller and several machine controllers. Section 4 gives details about the current capabilities of the controllers. Finally, Section 5 looks at the future evolution of the cell.

2. VIRTUAL MANUFACTURING CELL

The cell contains software packages which run on a wide range of hardware platforms and operating systems. These packages support the applications which implement the following functions: cell control, machine control, cell simulation, machine simulation, routing, operations planning, scheduling, NC verification, and shop floor data collection. Prototype applications under development at NIST include order entry, job routing, cell control, machine control, and process plan editing. Commercial software packages include:

- product data management, MATRIX™
- operations planning, ICEM™ PART
- shop floor simulation, QUEST™
- NC program simulation, VNC™
- scheduling, AUTOSCHED™

These packages run on a number of hardware platforms under a number of operating systems including: IRIX™ on SGI machines, SOLARIS™ on SUN workstations, and WINDOWS NT™ on desktop and notebook PCs.

We will show a high degree of interoperability by replacing these particular packages and applications with others that have the same functionality. This will give vendors, manufacturers, and university researchers the opportunity to test the capabilities of a single software application on vastly different simulated manufacturing systems, or different software applications on the same simulated manufacturing system. Beginning next year, we plan to add some new applications such as Manufacturing Resource Planning (MRP), Manufacturing Execution Systems (MES), inventory control, tool management, and quality control.

3. THE CONTROL HIERARCHY

As noted in section 1, NIST has been involved in the theory, implementation, and demonstration of hierarchical control systems for a number of years. The control system being implemented in the virtual manufacturing cell is based on the approach described in (Davis, Jones, Saleh, 1992). The system being controlled contains a number of simulated machines grouped into a single cell. These machines may be machine tools, robots, or coordinate measuring machines. The control hierarchy contains a cell controller and one machine controller for each machine. Each of the controllers in this hierarchy performs the same four functions: planning, scheduling, execution, and monitoring. The cell controller performs these functions to control the movement of parts around the cell. Each machine controller performs these functions to control the parts at a single machine.

Our goal is to have this virtual cell emulate a real factory as closely as possible. At the moment, there are some mnior differences (see Figure 1). In a real factory, the cell controller would issue a collection of commands to each machine controller. The machine controller, which could be a computer or a human operator, would issue commands to the commercial controller that controls the motions of the real equipment. Currently, the controllers in the hierarchy for the virtual cell behave in a slightly different way. First, the cell controller interfaces directly with the cell simulator. It does not issue commands directly to machine controllers. Second, machine controllers are embedded in the machine simulators. These controllers can be activitated in three ways: by the cell simulator, by an external operator, or by the workflow manager. Activation occurs whenever parts arrive at a machine to be processed. These behaviors will be modified in the coming year so that the controllers in the virtual cell behave more like controllers in a real cell.

Currently, this control hierarchy is supported by several prototype applications developed at NIST that implement the following functions: order entry, routing, and operations planning. The order entry application creates a file that contains the orders to be manufactured in the virtual cell. Each order contains the following information: Order ID, Part ID, Part Description, Order Size, Routing ID, Lot Size, Release Date, and Due Date. The routing application generates the cell level process plan. This plan contains all the possible routings through the cell for a given order. Each plan contains the following information: Routing ID, Part ID, Step ID, Step Description, Alternate Step ID, Next Step ID, Machine ID, and Processing Time. The operations planning application generates the machine level process plan. This plan contains all of the tasks that the machine is expected to perform and all of the data needed to perform those tasks. The structure and information content for this plan are described in detail in section 4.1. The output files generated by these applications are managed by MATRIX™.

3.1 The Cell Controller

As noted above, the cell controller is responsible for planning, scheduling, executing, and monitoring the movement of parts around the cell. The planning function generates a run-time routing that will enable each order to be completed by its due date. This plan contains: 1) the list of machines to be used in the production of each part, 2) precedence relations among those machines, and 3) expected duration at each machine. The generation of this plan involves the following steps.

1) retrieve the process plan (see below) for this order
2) parse the plan to identify all alternative routings
3) select feasible routings based on the current state of the system

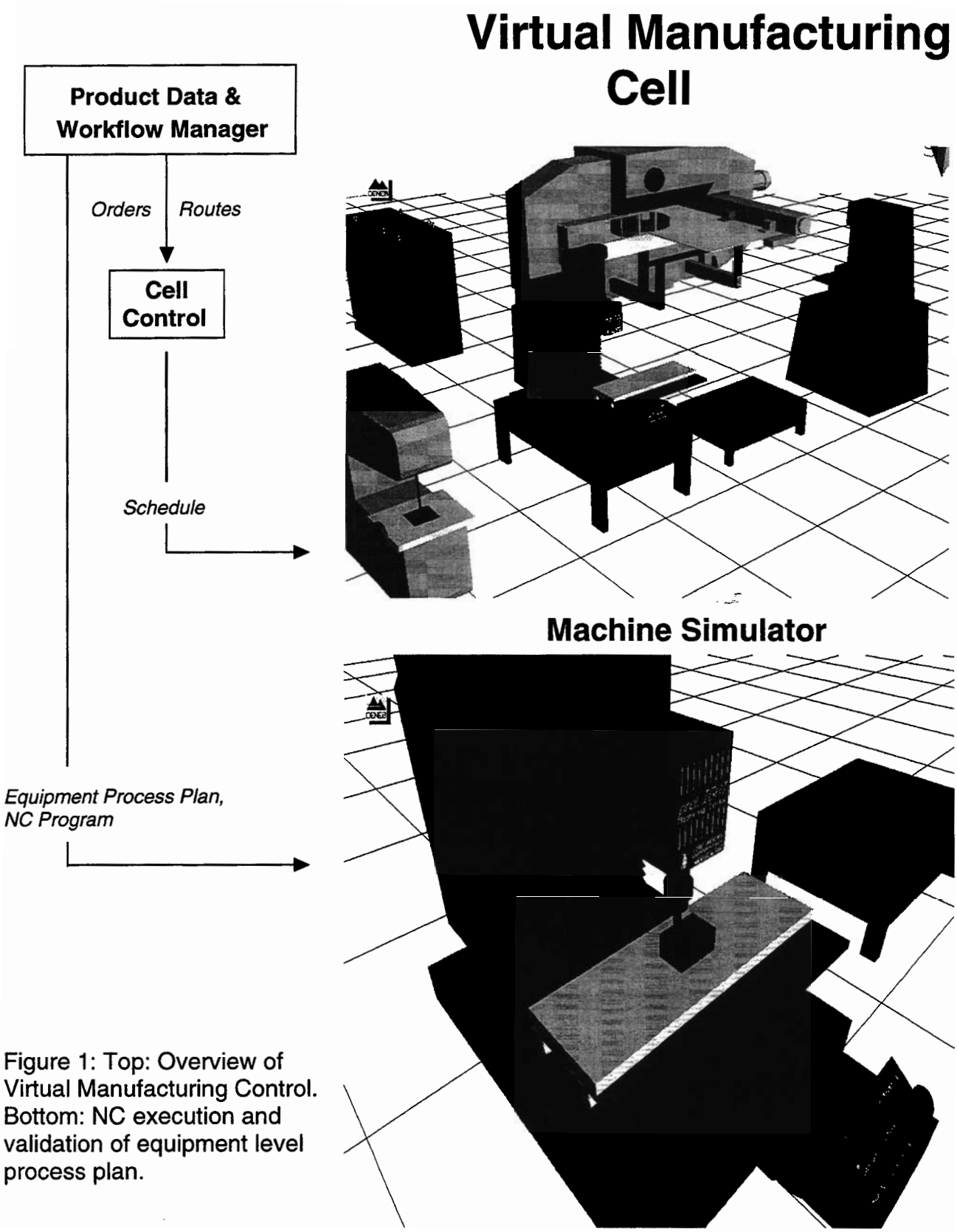

Figure 1: Top: Overview of Virtual Manufacturing Control. Bottom: NC execution and validation of equipment level process plan.

4) rank all feasible routings based on current performance criteria and process times
5) select routing which optimizes those criteria
6) pass this run-time routing to the scheduler.

Whenever conditions in the cell evolve to the point that the due date for this order cannot be met with the current routing plan, a new one must be created. The planner will repeat the preceding steps to select one of the remaining alternatives.

For each new order, the scheduling function uses its due date, routing, processing times, current cell status, and the current performance measures to determine predicted start and finish times at each machine in routing. The scheduling function also uses the updated state information from the monitoring function (see below) to determine whether or not to generate a revised schedule.

The execution function formulates the data package that drives the cell simulator. This package contains the orders, the routing for each order, and the dispatch list for each machine. The order file is used to create an initialization file for the simulation. Each time a job is finished at one machine, the simulator determines its next destination from the routing file. The dispatch list contains the exact sequence of jobs to be performed at every machine.

The monitoring function keeps an up-to-date model of cell status. This model can be compared to the predicted model that comes from the scheduler to determine if everything is "on-track". This comparison can be done automatically by the monitoring function or by a human operator. Whenever a problem occurs, the scheduler can be invoked using the model from the monitoring function.

3.2 Machine Controllers

Each machine controller is responsible for planning, scheduling, executing, and monitoring operations at a single machine. For each job that arrives at the machine, the planning function creates a run-time operations plan. This plan contains the tasks that must be performed in order to complete the machining, inspection, or material handling for that part. There are two types of scheduling that must be done by the machine controller: sequencing of parts at the machine and sequencing of tasks within the operations plan. As noted above, the sequencing of the parts at the machine is provided by the cell execution function. Since the operations plan contains only one feasible sequence of tasks at this time, no additional scheduling is required. The execution function formulates the data package that the machine simulator needs to execute the tasks in that plan. Since all of these tasks are executed automatically within the machine simulator, the only feedback occurs when all tasks have been completed. This feedback is sent to the cell simulator that forwards the appropriate the message to the cell level monitoring function.

4. MORE ON IMPLEMENTATION

4.1 The Cell Controller

In the current implementation, each cell level process plan contains only one feasible routing. The planning function simply retrieves this plan from MATRIX™, parses it and sends it to the scheduling function. The scheduling function, implemented in AUTOSCHED™ , computes the start and finish times at each of the machines in this routing.

The execution function formulates the data package that drives the cell simulator implemented in QUEST™. This data package contains a list of orders, the routing for each order, and the dispatch list for each machine. The order and the routing files are read into the cell simulator from MATRIX™. The order file is used to initiate the simulation. Each time a job is finished at one machine the cell simulator determines its next destination from the routing file. The dispatch list is derived from the schedule produced by AUTOSCHED™. It contains the exact sequence of jobs to be performed at every machine. This sequence overrides the normal queue selection rules used by the cell simulator to pick the next job.

The monitoring function maintains an up-to-date model of cell status in the relational database which is implemented in ACCESS™ . A preliminary information model describing the entities and their relationships can be found in (Jones, Riddick, Rabelo, 1996). Updates to this database are based on a collection of "change" messages issued by the cell simulator each time an event occurs in the simulation. These messages are written to a file which resides in a public directory on the network server. This file is retrieved by a file handler that parses the file and creates the necessary Standard Query Language (SQL) update queries. These queries are used to update the model in the database.

Machine controllers implement the planning, scheduling, and execution functions. The implementation of the scheduling function is discussed above. The planning and execution functions are implemented in the same prototype application within the simulation package, VNC™. This simplifies prototype development because of the native bindings with the machine simulator in the simulation environment. Moreover, since they are implemented sequentially within the same application, no feedback is needed on the completion of individual work elements within the plan. Hence, we have no need for a monitoring function at this time. In the following sections, we provide more details on the implementation of the planning and execution functions.

4.2 The Machine Controller

In the following sections we provide more details on the implementation of the machine controllers.

4.2.1 The Planning Function. As noted above, the planning function must retrieve and parse an operations

plan. The current operations plan structure is based on work described in (Mclean, 1987). The plan consists of keywords, work elements, and associated attribute/value pairs. There are presently three keywords corresponding to the three sections in the plan: HEADER, RESOURCE, and PROCEDURE. The HEADER section contains administrative information about the plan such as the part_id, machine_id, date, revision_number, etc. The RESOURCE section lists all of the data files that appear in the PROCEDURE section. The PROCEDURE section can contain an arbitrary number of work elements, which are listed in a sequential manner indexed by a step number . These work elements are directives indicating a type of task to be completed.(see Figure 2).

Figure 2. Simple Operations Plan

```
HEADER
    part_id = test_part1
    plan_name=setup2
    machine_id = vmc100
    date=July, 6, 1996
    revision_number=7
    end_header

RESOURCES
    machine_file = WORKCELLS/VMC_100
    tool_file = TOOLS/TWIST_DRILLS_225
    fixture_file = FIXTURES/FIXTURE_75
    workpiece_file = WORK/WORKPIECE_1
    nc_program_file= NC/test_part1.cnc
    end_resources

PROCEDURE
Step 1  LOAD_TOOL
    tool_name = TWIST_DRILLS_225
    tool_id = T7
    slot = 7
    end_step
 Step 2  LOAD_FIXTURE
    fixture_name = CLAMP
    fixture_id = 75
    location = x_axis
    x_offset = 152.4
    y_offset = 101.6
    z_offset = 44.45
    end_step
Step 3  LOAD_WORKPIECE
    workpiece_name = BAR_3x5x2
    workpiece_id = WORKPIECE_21
    location = Fixture
    x_offset = 0
    y_offset = 0
    z_offset = 0
    end_step
Step 4  LOAD_NC_PROGRAM
    nc_program_name= test_part1
    machine_id  = vmc100
    end_step
Step 5  RUN_NC_PROGRAM
    end_step
```

Each work element contains attribute/value pairs that represent the data files required to implement that particular work element. At the moment, we support LOAD_TOOL, LOAD_FIXTURE, LOAD_ WORKPIECE, LOAD_ NC_ PROGRAM, and RUN_NC_ PROGRAM. LOAD_TOOL requires a tool_name, tool_id, slot. LOAD_FIXTURE requires a fixture_ name, fixture_ id, location, x_offset, y_offset, and z_offset. LOAD_ WORKPIECE requires a workpiece_ name, workpiece_id, location, x_offset, y_offset and z_offset. LOAD_ NC_ PROGRAM requires an nc_program_ name and machine_id. RUN_NC_ PROGRAM has no attributes.

4.2.2 The Execution Function We are using the VNC™ simulation package to execute the operations plan. To accomplish this, simulation models of several entities are required including the machine, cutting tools, a fixture, a workpiece, and the machine controller. These models are generated in the simulation environment and saved into individual files that are managed by the product data manager.

For a machine, the model consists of the following: geometry of the physical parts of the machine, the degrees of freedom of the machine, the association of each machine physical part with each degree of freedom, the definition of the degree of freedom as translational/rotational and the direction of translation/rotation, the translation/ rotation travel limits, the axis translation/rotation travel speeds/ accelerations.

The model of the machine controller consists of a MIMIC™ configuration file that is the controller emulator within VNC™. This emulator is capable of reproducing the functionality of any CNC controller in the simulation environment. Unique configuration files have been created and configured for several machine tool controllers.

For a tool, the model consists of the following: geometry of the physical parts of the tool, the definition of which geometry is the cutting edge, and the definition of the name of the tool as it will be referred to in the simulation. The tool geometry is composed of two sub-geometries: a geometry of the shank and a geometry of the cutting edge. For fixtures and workpieces, the model consists of the geometry of the fixture/workpiece and the definition of a name.

Once all of these models have been retrieved into the environment, the NC program is executed. No feedback is provided until either an error occurs or the program runs to

completion. If any errors are detected, they are reported back to the process planner, who must make modifications to the operations plan. Typical errors include missing files and a variety of cutting problems (for example, tool cuts into fixture). Once the simulation has run to completion, the final simulated part can be compared to the original designed part.

5. FUTURE PLANS

The virtual manufacturing cell will continue to evolve over the next several years. During that time we will focus on four areas: interoperability testing, capability testing, expanded functionality, and distribution over the Internet. Because of the focus on interoperability, we will spend considerable time interchanging software applications that implement the same functions. Furthermore, we can begin to measure the impact of various types of integration mechanisms including message passing, file transfer, databases, common object request brokers, and work flow management systems. The virtual manufacturing cell will provide a unique opportunity for testing the capabilities of the existing software applications and adding new ones. Manufacturers, vendors, and university researchers will be able to test the capabilities of a single software application on vastly different simulated manufacturing systems, or to test different software applications on the same simulated manufacturing system. We plan to add new software applications such as MRP, MES, inventory control, tool management, and quality control. Finally, we want to make the various testing capabilities of the virtual manufacturing cell available to other researchers around the world using emerging Internet tools such as JAVA (Gosling, Joy, Steele, 1996).

** No approval or endorsement of any commercial product by the National Institute of Standards and Technology is intended or implied by these selections.*

ACKNOWLEDGMENTS

Work described in this paper was sponsored by the US Navy Manufacturing Technology Program and the NIST Systems Integration for Manufacturing Applications (SIMA) Program and is not subject to copyright.

The authors wish to thank Chuck McLean for his advice on process plan work elements.

REFERENCES

Albus, J., Barbera, A., and Nagel, R., Theory and Practice of Hierarchical Control, *Proceedings of 23rd IEEE ComputerSociety International Conference*, 18-39, 1981.

Davis, W., Jones, A., and Saleh, A., A Generic Architecture for Intelligent Control Systems, Computer Integrated Manufacturing Systems, Vol. 5, No. 2, 105-113, 1992.

Gosling, J., Joy, W., and Steele, G., The JAVA Language Specification, *Addison-Wesley*, 1996.

Jones, A., Riddick, F., and Rabelo, L., Development of a Predictive/Reactive Scheduler using Genetic Algorithms and Simulation-based Scheduling Software, *Proceedings of AMPST'96*, 589-598, 1996.

McLean, C., Interface Concepts for Plug-Compatible Production Management Systems, *IFIP WG 5.7: Information Flow in Automated Manufacturing Systems*, North Holland, 307-318, 1987.

Simpson, J., Hocken, R., and Albus, J., The Automated Manufacturing Research Cell of the National Bureau of Standards, *Journal of Manufacturing Systems*, Vol. 1, No. 1, 17-32, 1982