ANNALS of the ORADEA UNIVERSITY.

Fascicle of Management and Technological Engineering, Volume VII (XVII), 2008

CONSIDERATION REGARDING A GENERIC REFERENCE ARCHITECTURE MODEL FOR FMS

Ilie Octavian POPP
"*Lucian Blaga*" University of Sibiu, Faculty of Engineering "*Hermann Oberth*"
Departments of Machine and Equipments, Emil Cioran St. no. 4, Sibiu
E-mail: ilie.popp@ulbsibiu.ro, Tel. 0269-216062/454

Key words: work cell, reference architecture model, virtual manufacturing device.

Abstract: The controller architecture proposed in our system use a client - server architecture where all the devices/resources in the system synchronize their activities by sending messages to the central controller. Each device in the system has a corresponding module in the cell-control system, called the virtual manufacturing device (VMD). The VMD handles the resource-specific control tasks.

.

1. INTRODUCTION

The devices or the resources that may comprise a work cell can be classified as follows based on their functionality:

- Processors that include the machine tools or measuring stations of a flexible manufacturing system.
- Transportation units that move the products between processors and storage units. These include the conveyors, AGV's, etc.
- Storage units that hold the products between the different processing stages. These include automatic storage and retrieval systems (ASRS), queuing carousels, recirculating buffers.
- Ports that act as material interface places for the jobs.

Depending on the type of device the VMD is controlling, they are classified as processor modules, storage modules, transport modules or port modules. From a control point of view, there is no difference between (say) storage module and a processor module. However, from a functional point of view the difference between the various modules is obvious. The processor module is controlled by the cell controller and is responsible for the correct commands being sent to the physical device. The module supports the following functions:

- Keeping track of the current state of the device.
- Determining the NC programs or the device programs to be executed.
- Downloading NC programs or the device programs to the device.
- A set of control functions that control the device operation.
- An interface for communicating to/from the device.
- Keeps track of the different parts in the device and their states.
- Keeps a log of all activities that have taken place in the device.
- Upload processing data from the device.

The transport module is also controlled by the cell controller and is responsible for the transportation of the part to be sent to the processor/storage unit that performs the next operation. The transport module supports the following functions.

- A set of control functions that control the device operations.
- An interface for communicating to/from the device.
- Keeps track of the different parts in the device and their states.
- Keeps a log of all activities that have taken place in the device.
- Determining the way the move operation is to be carried out, for example which program to execute and which gripper to use.
- Downloading of robot programs to the transport device.

The virtual manufacturing device (VMD) can be split into two parts, one generic and another specific. The generic part provides a uniform interface to the cell controller and enables it to be reused from previous applications. The generic part keeps track of the current state of the external resource and is used to send and receive messages from the cell controller. The specific part of the module is actually a wrapper around the device that translates the high-level messages handled by the generic part to the proprietary protocol handled by the physical device. This part of the module can also be thought of as a device driver. The device driver also provides some of the functions that are not provided by the physical device but promised by the generic interface. For example, the device driver incorporates the notion of input and output ports that act as transfer points for the parts flowing into the system.

2. GENERIC RESOURCE MODELS

The generic part of the resource model is based on the behavior of the devices that belong to a certain class. The devices in the work cell are broadly divided into four classes as mentioned earlier. We have implemented generic resource models for each of these. Since all devices belonging to the same class have similar functionality, they can be controlled through the same generic interface. We present the behavior models of the main resource models in our system viz. storage, transport, and processor module in the section below. A finite state model is used to depict the behavior model of each of these resources. The finite state model for the various resource modules are similar except for the behavior when the resource is in the running (or initiated) state. Thus at the highest level, the generic behavior of all resources is the same and is expressed as a finite state machine shown in figure 1.

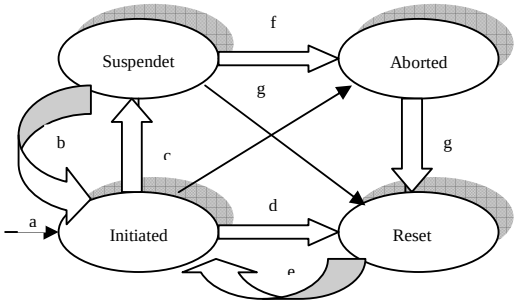

Fig. 1 ***Generic behavior of a resource expressed as a finite state machine***

Label	Event	Activation Agent
a	Start	Cell Controller
b	Resume	Cell Controller
c	Suspend	Cell Controller
d	Reset	Cell Controller
e	Restart	Automatic
f	Abort	Cell Controller
g	Reset	Cell Controller

The initiated state for each of the resource modules is depicted separately in figures 2a and 2b.

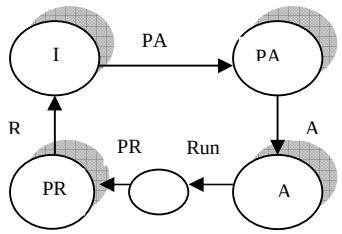

Fig. 2a: ***Finite state model of a Processor Module with unit capacity***

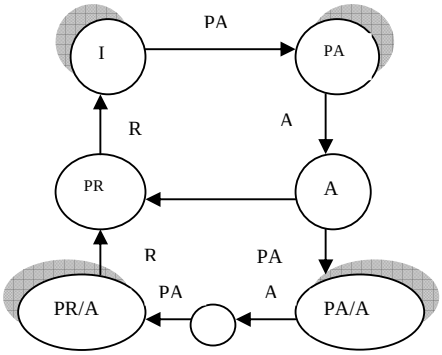

Fig. 2b: ***Finite state model of a Storage Module with a capacity of two***

Table 1 is the event table for this finite state machine.

Table 1: Event Table

Events	Actions and Resulting States
PA	Resource receives prepare to accept message. Device runs the appropriate program and to make the port read to accept a job. The resulting resource state is PA (Prepared to Accept).
A	Resource receives an accept message. It runs a n appropriate program to accept a job at the port. The resultant state is A (Accepted a Job)
Run	Resource receives a message to start processing a job. It runs the appropriate program. The resultant state is C (The processing of the job on this resource is complete)
PR	Resource receives prepare to remove message. Device runs the appropriate program and to make the port read to remove a job. The resulting resource state is PR (Prepared to Remove).
R	Resource receives an remove message.

[4]. Popp I. - *Consideration regarding a Work cell and Resource Model implementation in FMS*, Annals of the Oradea University, Fascicle of Management and Technological Engineering, CD-ROM Edition, Oradea, 2006

[5]. Wyns J., Brussel H., Valckenaers L., *WorkStation Architecture in holonic manufacturing systems*, Cirp Journal on Manufacturing Systems, Vol**.** 26, 220-231, (1996).

[6]. Reveliotis S., *Structural Analysis and Control of Flexible Manufacturing Systems with a Performance Perspective*, PhD Thesis, Univ. of Illinois at Urbana-Champaign, 1996

[7]. * * * *Computer Integrated Manufacturing (CIM)* Framework Specification Version 2.0", SEMATECH. INC, 1998.

event and send back an acknowledgement to the controller when it has completed, indicating that its internal state has changed. In the public interface of the generic resource models, `send Message' and `receive Message' procedures have been included in order to facilitate specific message passing between devices and the controller. The message passing is in asynchronous mode so that the resource modules are not blocked waiting for acknowledgements. A message format that is used in our resource model is described below.

The Message Format is as follows:

Message Type: Identifies the message type of the command. Valid values are the following: command messages, reply messages.

Message ID: See list of message IDs below:

Sender ID: The unique ID of the sender of the message that will enable the recipient to send the acknowledgements.

Transaction ID: A transaction id that will enable the application to associate the message with the appropriate transaction. Every asynchronous call will include a transaction id with it.

Job ID: Optional. The job associated with each message, if present, is included in the message definition.

Table 2: List of Message ids

MESSAGE	ID option
PREPARE_ACCEPT_ACK	Device is ready to accept (load) job.
ACCEPT_ACK	Device has accepted job and is ready to start processing
B_COMPLETED	Completed (finished processing).
PREPARE_REMOVE_ACK	Device is ready to unload job.
REMOVE_ACK	Device is unloaded.

4. CONCLUSIONS

It can be concluded that in spite of the enabling hardware and software technologies available today, the current practice of designing, building and operating manufacturing systems seems to preclude any possibility of exploiting its inherent flexibility and configurability. In short, such systems while allowing some product mix flexibility leave little or no room for volume and configuration flexibility. This research and development effort will focused on the concept of developing a generic reference architecture model for the specification, development, control and reconfiguration of a manufacturing enterprise at a work cell level. It introduces the concept of scalable flexibility in manufacturing from the shop floor to enterprise level.

REFERENCES

[1]. Adlemo A., and S.-A., *Models for Specification and Control of Flexible Manufacturing Systems*, Technical Report, School of Electrical and Computer Engineering, Chalmers University of Technology, Goteborg, Sweden, 1997.

[2]. Lawley M., *Structural Analysis and Control of Flexible Manufacturing System*, PhD Thesis, University of Illinois at Urbana-Champaign, 1995.

[3]. Popp, I., *Consideration regarding model Architecture for Scalable Flexibility in Manufacturing*, International Conference on Manufacturing System, Buletinul Inst. Politehnic din Iaşi, secția Construcții de maşini, Iaşi, 2005.

3. FINITE STATE MODEL OF A RESOURCE MODULE

The state chart diagrams of the running state of the main resource modules are given in figures 2a and 2b.

The resource module shown in figure 2a is a unit-processor. It can accommodate only one job at a time. The states that are reached by the resource are given below:

- IP The idle state of the resource
- PA The resource is prepared to accept a new job at an appropriate port.
- AP The resource has accepted a new job and is waiting to process it.
- CP The resource has completed processing a job.
- PR The resource is prepared to unload a job at an appropriate port.

The resource module shown in figure 2b can accommodate two jobs simultaneously. The resource reaches the following states when it has than more than job.

- PA/A the resource has accepted one job and is prepared to accept a second job at an appropriate port.
- A/AA the resource has accepted two jobs.
- PR/A the resource has accepted one job and is prepared to unload one job out of the system.

Messaging Interface: The messaging interface provided by the resource modules to allow the cell controller to interact with the physical devices in a work cell is discussed in this subsection. The generic resource module presents a uniform control interface to the cell controller so that the controller can call an appropriate function on the resource module for moving parts from one device to another, loading a machine and the start of processing. As seen from the state charts above, we find that the resource can be in one of five internal states when it is in the initiated state.

A brief description of the internal states is presented below:

The events in this finite state machine are the same as those described in figure 2a. The additional state PA/A, A/A and PR/A indicate that the resource has already accepted a job and is preparing to accept, accepting and removing an additional job

- Prepare Accepted State: The device is ready to accept a new job at a particular port. The device would have run an appropriate NC program so that the preparatory steps required to accept a job at the port are completed. When a device is in a Prepare Accepted state, it only accepts an ACCEPT signal corresponding to the same transaction.
- Accepted State: The device has accepted the job at the port and is ready to begin processing on the job. If the device is not a processing device, it does not carry out any operations and it remains in this state until it receives a PREPARE_REMOVE signal. Otherwise, it stays in that state until it receives a RUN signal.
- Completed State: The device has completed a processing operation on the job. The device remains in this state until it receives a PREPARE_REMOVE signal.
- Prepare Removed State: The device is ready to unload a job from the appropriate port. The device would have run an appropriate NC program so that a job at the port can be unloaded on receiving a REMOVE signal. This would imply that the job would have been moved to the correct position and the port setup so that it interfaces with the `accepting' port (of the receiving device).
- Removed State/ Init State: The initiated state of the device where it is ready to accept new PREPARE_REMOVE signals from the cell controller.

The generic part of the resource model supports a set of functions like ‘prepare to accept', ‘accept Job', ‘run Program' so that the cell controller can call the appropriate functions (trigger the event) to change the internal state of the resource. The device driver handles the transformation of such calls to the proprietary protocol handled by the physical devices. The physical device has to perform appropriate actions on receiving such an