Proceedings of the 17th International Conference on Manufacturing Systems – ICMaS
ISSN 1842-3183

University POLITEHNICA of Bucharest, Machine and Manufacturing Systems Department
Bucharest, Romania

MODELING WORK CELLS AND MANUFACTURING SYSTEMS FOR SCALABLE FLEXIBILITY IN MANUFACTURING

Ilie Octavian POPP, Ioan Barsan, Dorin TELEA

Abstract *- A framework for organizing resources consisting of hardware devices (such as machine tools, robots, conveyors, etc) and software modules (such as cell controller, monitoring software) in a CIM environment has been developed. We focus on the basic building blocks of the framework and are given some sample configuration files for the resources and the work cell.*

Key words: *FMS, reference architecture, modeling, supervisory control/synchronization, configuration file.*

1. INTRODUCTION

The work cell can be setup to support different manufacturing environments. A work cell can be configured to have multiple buffers, separate I/O ports storage transportation devices and so on. The ports associated with each device and the connectivity information is defined in the work cell configuration file. The job routes supported at the work cell level are also specified in the configuration file. The operational logic of the work cell controller in such architecture is relatively independent of its configuration. The appropriate procedure calls (handshake messages) are made on the devices sequentially as given in the job definition. The cell controller waits on an acknowledgement from the resource module before making the next call as explained in paper Ref. [2] and paper Ref. [4].

The problem of supervisory control/synchronization in a flexible-manufacturing environment is one of the most difficult problems designers'face in the conceptualizing of a Flexible Manufacturing System. It is clear that manufacturing flexibility induces complexity. This research and development effort proposes such a reference architecture that will allow for the control and reconfiguration of a flexible manufacturing work cell. This architecture will address many different issues, such as, the type of resources in the system, system capabilities, system behavior, and architecture interfaces. It also develops the rules for synthesizing complex manufacturing systems.

The steps involved in modeling work cells and manufacturing systems are described in Ref. [1] and Ref. [5].

• The configuration files for each of the basic resource has to be written. These files define the capacity of the resource, the capabilities of the machine (programs, configurations) and the group to which it belongs amongst other things.

• The configuration file for the work cell has to be written. This file defines all the resources that a work cell is composed of. In addition, this configuration file contains defines the connectivity information and the various routes followed by the different job types.

• To attach machine simulators to each of the basic resources, device driver files have to be written that translates commands from the module controllers to the simulators and vice versa.

Note the use of a Finite State Automata class in Fig. 1. All transactions and messages in the work cell as well as resources are derived from a state Machine. This implies that all the resources have a strict notion of their current states and all the events are state driven.

The fields that are mandatory fields in a configuration file are the following:

• The name used to locate the resource in the distributed environment.
• The type of the resource
• The server kind field that serves as a de-multiplex key.
• The (buffer) capacity of the resource.
• The port numbers in a resource
• The total number of setups supported by the resource.
• The programs supported in each setup. A program Id (PID) and a filename describe a program.

The additional fields that have to be described in a configuration file for a composite member are described below:

• The members (resources) that the work cell is composed of.
• The capacity of each of the resource.
• The in-ports and out-ports of each of its resource.

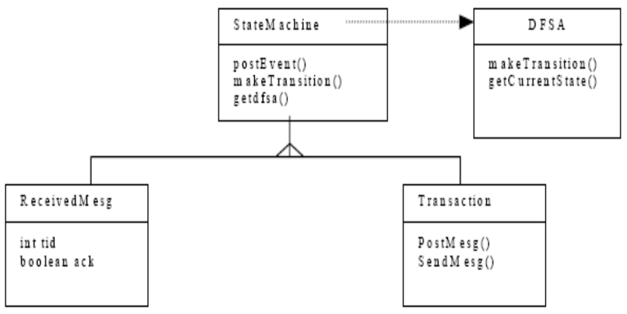

Fig. 1. The OMT diagram for the transaction class and the received message used in the work cell and the resource server class.

• The connectivity information that describes how the ports of the resources are connected to each other.

• There are two keywords used in defining the connectivity information. 'TO' is used to describe one-way connectivity between ports while 'ONTO' means that the connectivity between the ports is two-ways.

• The different job types (routes) supported in each configuration. A route is described by a sequence of stages each stage defined by the resource that the job needs at that stage.

The device drivers for a storage type and a processor type have been described above.

The steps involved in defining a device driver are given below:

• Define the ports of the device.

• The link of the mechanism associated with the port.

• The location of the port with respect to the link coordinates.

• Define the programs associated with accept, remove, prepareAccept and prepareRemove commands at various ports.

• Define the mapping between programs IDs (PID) and the programs that would be run by the devices.

However, the portmaps have to be defined differently if the capacity of the device is greater than one. The program associated with the accept, remove, prepareAccept and prepareRemove commands are indexed both by the port and the buffer ID.

2. TEST CASES MODELED USING THIS ARCHITECTURE

A. Case 1

A simple work cell composed of base resources has been tested successfully using our architecture. The work cell consists of a three-axis machine tool (a unit processor), a pallet changer and an input/output buffer (Fig. 2). The pallet changer is fed jobs/parts from the work cell buffer and it feeds the jobs into the machine tool. The pallet changer picks up a processed job from the machine tool and transfers the job into the output buffer of the work cell.

In the above setup, the capacity of the work cell is four while the processor is of unit capacity. In such a configuration, the interactions between the pallet changer and the processor are dynamic in the sense that they are dependent on the processing time spent by each job on the machine tool and the time at which different jobs enter the system. The controller of the work cell automatically allows/disallows different transitions thereby avoiding conflicts. Some sample configuration files for the resources and the work cell are given below. The device driver file for this example is also listed below.

Configuration file for the work cell (composite member):

```
WORKCELL work cell1 {OWNER: NONE
SERVERKIND: WorkStation
CONFIGURATIONS: 1
ME MT1
CURRENT_CONFIG: 1
```

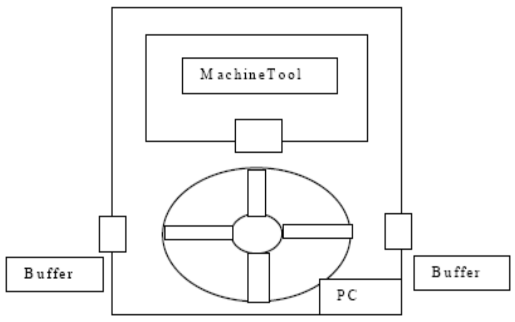

Fig. 2. Schematic sketch of a work cell consisting of a pallet changer and a machine tool.

```
RESOURCES {DEVICE: DEVICENA
CONFIGURATIONS: 1
CURRENT_CONFIG: 1
RESOURCES {DEVICE: DEVICENAME MT1
CAPACITY 1 TYPE PROCESSOR PORTS
{INPORTS: P1 OUTPORTS: P1}
DEVICE: DEVICENAME DUMMY CAPACITY 1000
TYPE STORAGE PORTS
{INPORTS: P1 OUTPORTS: P1}
DEVICE: DEVICENAME PC1 CAPACITY 4 TYPE
STORAGE PORTS
{INPORTS: P3: P2 OUTPORTS: P1: P2}}
CONNECTIVITY {MT1: P1 ONTO PC1: P2 DUMMY:
P1 TO PC1: P3 PC1: P1 TO DUMMY: P1}
ROUTE {CONFIGURATION 1 {JOB 1 {STAGE
DEVICE DUMMY CAPACITY 1 PROGRAM 1
GEOMETRY part1.iv SCALE 2 STAGE DEVICE PC1
CAPACITY 1 PROGRAM 1 GEOMETRY part1.iv
SCALE 2 STAGE DEVICE MT1 CAPACITY 1
PROGRAM 1 GEOMETRY part1.iv SCALE 2 STAGE
DEVICE PC1 CAPACITY 1 PROGRAM 3 GEOMETRY
part1.iv SCALE 2 STAGE DEVICE DUMMY
CAPACITY 1 PROGRAM 2 GEOMETRY part1.iv
SCALE 2}
JOB 2{STAGE DEVICE DUMMY CAPACITY 1
PROGRAM 1 GEOMETRY part1.iv SCALE 1 STAGE
DEVICE PC1 CAPACITY 1 PROGRAM 2 GEOMETRY
part1.iv SCALE 0.5 STAGE DEVICE DUMMY
CAPACITY 3 PROGRAM 2 GEOMETRY part1.iv
SCALE 3}}
```

Configuration file for pallet changer (Storage):

```
RESOURCE PC1 STORAGE {OWNER: work cell1
SERVERKIND: WorkStation
CAPACITY: 4
PORTS: P1: P2: P3
MODEL: FANUC1998
EQUIPMENTNAME: TRI_SPEC
CONFIGURATIONS: 1
CURRENT_CONFIG: 1
CONFIGURATION: 1
{PROGRAMS {PID: 1 FILENAME: gcode1.dat PID: 2
FILENAME: gcode2.dat }}
```

Configuration file for the machine tool (Processor):

```
RESOURCE MT1 PROCESSOR {OWNER: work cell1
SERVERKIND: WorkStation
```

CAPACITY: 1
CONFIGURATIONS: 1
PORTS: P1
MODEL: MITSUBSH79
EQUIPMENTNAME: MILL_END
CURRENT_CONFIG: 1
CONFIGURATION: 1
{PROGRAMS {PID: 1 FILENAME: gcode.dat}}

Configuration file for device driver of pallet changer (Storage):

DRIVER PC1 TYPE STORAGE {PORTS {INPORTS : P1 LOCATION : 0000000 LINK : dummy: P2 LOCATION : 0000000 LINK : dummy: P3 LOCATION : 0000000 LINK: dummy OUTPORTS : P1 LOCATION : 0000000 LINK : dummy : P2 LOCATION : 0000000 LINK : dummy : P3 LOCATION : 0000000 LINK : dummy}
PORTMAPS {PACCEPT P1: BUFFER 1: PID ge11.dat PACCEPT P2: BUFFER 1: PID ge12.dat PACCEPT P3: BUFFER 1: PID ge13.dat PACCEPT P1: BUFFER 2: PID ge21.dat PACCEPT P2: BUFFER 2: PID ge22.dat PACCEPT P3: BUFFER 2: PID ge23.dat PACCEPT P1: BUFFER 3: PID ge31.dat PACCEPT P2: BUFFER 3: PID ge32.dat PACCEPT P3: BUFFER 3: PID ge33.dat PACCEPT P1: BUFFER 4: PID ge41.dat PACCEPT P2: BUFFER 4: PID ge42.dat PACCEPT P3: BUFFER 4: PID ge43.dat PREMOVE P1: BUFFER 1: PID ge11.dat PREMOVE P2: BUFFER 1: PID ge12.dat PREMOVE P3: BUFFER 1: PID ge13.dat PREMOVE P1: BUFFER 2: PID ge21.dat PREMOVE P2: BUFFER 2: PID ge22.dat PREMOVE P3: BUFFER 2: PID ge23.dat PREMOVE P1: BUFFER 3: PID ge31.dat PREMOVE P2: BUFFER 3: PID ge32.dat PREMOVE P3: BUFFER 3: PID ge33.dat PREMOVE P1: BUFFER 4: PID ge41.dat PREMOVE P2: BUFFER 4: PID ge42.dat PREMOVE P3: BUFFER 4: PID ge43.dat ACCEPT P1: BUFFER 1: PID ret1.pgm ACCEPT P1: BUFFER 2: PID ret2.pgm ACCEPT P1: BUFFER 3: PID ret3.pgm ACCEPT P1: BUFFER 4: PID ret4.pgm ACCEPT P2: BUFFER 1: PID ret1.pgm ACCEPT P2: BUFFER 2: PID ret2.pgm ACCEPT P2: BUFFER 3: PID ret3.pgm ACCEPT P2: BUFFER 4: PID ret4.pgm ACCEPT P3: BUFFER 1: PID ret1.pgm ACCEPT P3: BUFFER 2: PID ret2.pgm ACCEPT P3: BUFFER 3: PID ret3.pgm ACCEPT P3: BUFFER 4: PID ret4.pgm REMOVE P1: BUFFER 1: PID ret1.pgm REMOVE P1: BUFFER 2: PID ret2.pgm REMOVE P1: BUFFER 3: PID ret3.pgm REMOVE P1: BUFFER 4: PID ret4.pgm REMOVE P2: BUFFER 1: PID ret1.pgm REMOVE P2: BUFFER 2: PID ret2.pgm REMOVE P2: BUFFER 3: PID ret3.pgm REMOVE P2: BUFFER 4: PID ret4.pgm REMOVE P3: BUFFER 1: PID ret1.pgm REMOVE P3: BUFFER 2: PID ret2.pgm REMOVE P3: BUFFER 3: PID ret3.pgm REMOVE P3: BUFFER 4: PID ret4.pgm}
PROGRAMMAPS {PID 0: PROGRAM gcode.dat PID 1: PROGRAM gcode.dat STARTPROGRAM: initpgm.dat}
BUFFERS {1 LOCATION: 1.2 -1 0 0 5 0 5 -0.5 -0.5 LINK: palletslide1 2 LOCATION: 1.2 -1 0 0.5 0.5 -0.5 -0.5 LINK: palletslide2 3 LOCATION: 1.2 -3 0 0.5 0.5 -0.5 -0.5 LINK: palletslide3 4 LOCATION: 1.2 -3 0 0.5 0.5 -0.5 -0.5 LINK: palletslide4}}

Configuration file for device driver of the machine tool (Processor):

DRIVER MT1 TYPE PROCESSOR {PORTS {INPORTS: P1 LOCATION: 1 -1 -1 0.5 0.5 0.5 0.5 LINK: L4 OUTPORTS: P1 LOCATION: 1 -1 -1 0.5 0.5 0.5 0.5 LINK: L4 }
PORTMAPS {ACCEPT P1: PID home.dat REMOVE P1: PID home.dat }
PROGRAMMAPS {PID 0: PROGRAM gcodemt1.dat PID 1: PROGRAM gcodemt1.dat PID 2: PROGRAM gcodemt1.dat STARTP ROGRAM: start.dat}}

Note that the configuration files for basic resources are very simple while the configuration file for the work cell is more involved. This is not specific to this test case but is a more general situation.

B. Case 2

A work cell consisting of two Universal high speed placement machines (HSP) in serial and a conveyor that shuttles jobs between the two were modeled using this architecture (Fig. 3). The work cell supports two different job types. The two job types are defined in terms of the devices that they visit.

• Job1 - < HSP1 Conveyor HSP2>
• Job2 - < HSP2 Conveyor HSP1>

Each of the HSPs is modeled as a unit capacity processor while the conveyor is modeled as a unit capacity transportation unit. Since there are counter flow jobs in the system and the conveyor is a shared resource of unit capacity, there is a potential for deadlocks. The work cell controller makes sure that such situations don't arise. Some sample configuration files for the resources and the work cell are given below.

The device driver files for the HSP and the conveyors are also listed below. Note the similarity between the configuration file of the Universal machine and the 3-axis machine tool in the previous example. This is because the control information for both the machines (unit capacity processors) is the same. The device driver files for each of these machines is obviously different and is specific to the simulator that is used to model the machine. The work cell configuration file for this system is written down exactly the same way as explained at the start of this section. Notice that the geometry at each stage has been defined as 'user defined' in the route definition. This is a keyword in the language that means that the geometry of the part at

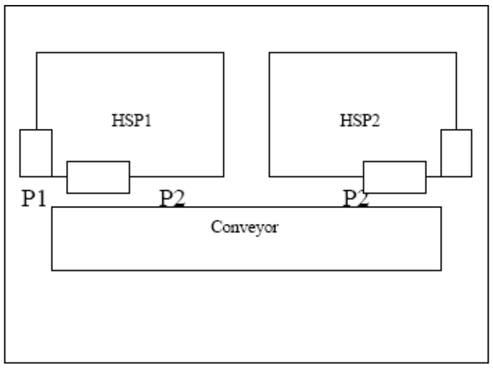

Fig. 3. Schematic Sketch of the HSP cell.

each stage would be redefined by the operation that takes place at that stage. That is, the part geometry has to be shared between stages.

Configuration file for the Universal HSP machine

```
RESOURCE MT1 PROCESSOR {OWNER: work cell1
SERVERKIND: WorkStation3
CAPACITY: 1
CONFIGURATIONS: 1
PORTS: P1
MODEL: MITSUBSH79
EQUIPMENTNAME: MILL_END
CURRENT_CONFIG: 1
CONFIGURATION: 1 {PROGRAMS {PID: 1
FILENAME: gcode.dat }}}
```

Configuration file for the Conveyor

```
RESOURCE CON1 TRANSPORT {OWNER: work cell1
SERVERKIND: WorkStation3
CAPACITY: 1
PORTS: P1: P2
MODEL: KOBE_76
EQUIPMENTNAME: T_AGV
CONFIGURATIONS: 1
CURRENT_CONFIG: 1
CONFIGURATION: 1
{PROGRAMS {PID: 1 FILENAME: port1.pgm PID: 2
FILENAME: port2.pgm PID: 3 FILENAME: port3.pgm
PID: 4 FILENAME: port4.pgm }}}
```

3. CONCLUSIONS

The use of automatic synthesis of supervisory controllers allows a high degree of flexibility in the system. Whenever there has been a significant change in the system configuration (when new job routes have been defined or when resources have been added/removed), the control-laws are recalculated and re-synthesized. We have suggested hierarchical synthesis as a strategy for rapidly configuring large systems. In this paper, a methodology for formally modeling hierarchical resource allocation systems is developed. A distributed hierarchical control policy for ensuring deadlock free behavior in such a system has been proposed. In paper, we apply this methodology to model a FMS setup under the framework of our architecture.

The software module we have implemented based on this architecture is highly configurable to suit the needs of a variety of manufacturing environments. A CORBA based framework has been used to develop the various object modules. This gives us the added benefit of being able to run the application across multi-platforms (operating systems).

Furthermore, the use of distributed object technology to implement the system enables us to run each resource module as a distributed object/server on a computer node. It is possible to access the control panels associated with each resource from a separate computer and this allows the operator to access the system at different control levels (the resource or the work cell).

Currently, the software implementation of this architecture has been restricted only to the resource and the work cell level. The higher-level modules and the distributed, hierarchical controller have not been implemented yet. Future work includes the development of the higher-level modules and the distributed controller that ensures deadlock free behavior of the entire system. This would help in realizing a scalable, 'unifying' operating system for manufacturing systems

REFERENCES

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

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

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

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

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

[6] Wyns, J., Brusse l, H., Valckenaers, L. (1996). *WorkStation Architecture in Holonic Manufacturing Systems*, Cirp Journal on Manufacturing Systems, Vol. 26, 220-231.

Authors:

PhD Eng, Ilie Octavian POPP, Assoc. Prof., "Lucian Blaga" University of Sibiu, Faculty of Engineering,
E-mail: ilie.popp@ulbsibiu.ro
PhD Eng, Ioan BARSAN, Professor, "Lucian Blaga" University of Sibiu, Faculty of Engineering,
E-mail: ioan.barsan@ulbsibiu.ro,
PhD Eng, Dorin TELEA, Professor, "Lucian Blaga" University of Sibiu, Faculty of Engineering,
E-mail: telea.dorin@email.ro