A Distributed Network Architecture For Managing Cells In An Integrated Automated Manufacturing Facility

J. Clark and J. Cooke
Capricornia Institute, Australia

G. S. Clark
Anvil Design, Australia

Abstract

The individual control of machines or processors is subordinate to the management of the entire manufacturing production function. This distinction is necessary in order to provide the appropriate separation of and detailed focus on process activities while still providing acceptable interfaces for the upload of data and the download of instructions, recipes, or commands.

Introduction

On the management side, the hierarchical collection, input processing, and data assimilation provides the information which triggers the management decision process. These decisions ultimately result in downloaded instructions or commands which control the individual process. Successful integration of the management function with control activities is not easy due to the diversity of manufacturing applications. Numerous machine tool vendor solutions require the generation of custom software in order to integrate equipment, each with its own controller, on the shop floor. This paper presents a software architecture that is intended to reduce the problems faced by systems integrators when implementing manufacturing cell management controllers. This architecture is based upon modelling manufacturing cells as finite state machines and utilizing the results of the model to simplify cell control to event-action processing.

It is important to understand that there is a major difference between managing the manufacturing production function of a plant or facility and the control of the individual process components of the automated manufacturing (or re-manufacturing, i.e. repair) function.

One successful approach toward automating the manufacturing production function is to provide multilevels of management hierarchy and control. Commonly referred to as architecture, it is easy to see, once it is pointed out, that there can be software as well as hardware architectures.

In a three - level architecture the management function resides in what can be referred to as the area host. Area in this sense is defined as the collection of all manufacturing cells, each controlling at least one process or a related set of processes which result in the manufacture or repair of a single identifiable unit. A unit may be a single discreet part, such as an engine turbine blade, or, the entire engine composed of numerous sub-assemblies, each of which is composed of many discreet parts. The complexity of the production unit greatly affects the complexity of the structure and number of levels in the architecture. However, it is generally agreed that a manufacturing plant should not have an architecture which exceeds five levels.

Management of the entire production function may be summarised by presenting the analogy of an orchestra, where the goal is to reproduce the score as written by the composer. Just the right blend of the individual instruments (manufacturing work stations) within the section (manufacturing cell) properly directed (integrated) together in space and time will achieve the desired result. Whether or not the musical result (product) is acceptable to the audience (customers) is not at issue as long as the score (manufactured product) truly represents the intent of the composer (product design engineer).

The intent of this paper is to focus on control aspects of the level two cell management controller. Figure 1 illustrates a typical three - level architecture for a manufacturing plant.

At the highest level is a Plant or Area Host (referred to here as an Area Host). This computer system is typically a minicomputer or some higher end machine. Its major role in the architecture is to facilitate data manipulation for Manufacturing Resource Planning (MRP) functions. MRP management functions include, but are not limited to, those activities which when properly adjusted maximise production while simultaneously managing inventory, production capacity, and quality.

Below the Area Host, the facilities is decomposed into areas of automation called 'cells'. Figure 1 describes two cells, a welding cell, and a inspection cell. Each cell is comprised of a group of equipment controllers and a cell controller. Each equipment controller's role is to monitor and respond to the continuous and discrete events inherent in and unique to operating the automated machinery directly under its control.

A discussion of event types will help clarify the difference between event processing in the automated equipment and event processing in the cell controllers. Continuous events are analogue phenomena such as welder voltages and images resulting from vision inspection cameras. In contrast, discrete events are representative of some unique mode or condition present in the system.

The cell controller's role is to respond to the discrete events which are transmitted from the equipment controllers. The events are discrete in that they are representative of some mode or condition present in the equipment controller. Examples of this type of event may be requests for some instruction as to what to do next or a message providing data relevant to the operation of other equipment in the cell. By responding to these events the cell controller is <u>managing</u> the state transitions which the equipment controllers can perform.

The cell controller must also function as a data acquisition point for the Area Host, providing such information as part tracking and lot verification data. The Area Host may also download data relevant to the next lot of components to be processed through the cell. The cell controller may need to initialize the cell based on this data.

What follows is a discussion of a software architecture for the cell management controller. The approach presented here involves two key ideas:

. Equipment controllers are modelled as finite state machines; and

. Communications with the controllers are handled asynchronously with respect to the cell control software.

Finite State Analysis

There are two categories of finite state machines; combinational and sequential state machines[1]. The difference between these categories lies in the notion that continuous state machine outputs are determined solely by its current inputs. An example of this is an analogue to digital converter. Components of equipment controllers could be modelled as continuous machines.

The output of a sequential state machine is determined by both its current and past inputs. This can be simplified by saying that sequential machines have a degree of memory (i.e. the past inputs), and combinational machines do not. This memory is represented in the form of states. A sequential machine is always in a specifiable state. Further, in any given state, certain events can cause a sequential machine to change state, produce outputs, or both,[1 pg 78). Basically, a sequential state machine models the possible events and necessary actions for state transitions inherent within a cell control system.

This event and action modelling attribute of the sequential state machine model make it ideal for specifying the management behaviour of a cell controller. Figure 2 presents a Cell State Table and simplified State Transition Table for the weld controller within the weld cell of Figure 1. Similar transition tables can be defined for the other equipment controllers. The number of possible states in the state transition table is simplified for the clarity of this discussion. Analysis of a cell similar to this welding cell, would reveal more states such as error, loaded and unloaded states.

This sequential state machine analysis has an important role in the software architecture. This being that the requirements identified during the cell requirements analysis phase map directly into the physical implementation. This will be made clear as the discussion progresses.

The Cell Management Controller Software Architecture

Figure 3 presents a graphic of the cell management controller architecture.

Figure 4 presents this architecture as it would be applied to the welding cell management controller of Figure 1.

As shown in Figure 3, the system architecture consists of the following modules:

. Asynchronous Communication management;

. A Event Queue and Log;

. A Event Dispatcher;

. A Cell Coordination Manager;

. A few Equipment Controller Managers; and as shown in figure 3

. State Transition Tables and a Cell State Table.

A discussion of each follows.

Asynchronous Communications Management

In order to modularize the complexity of communication subsystem management, highly reusable communications managers are needed. Each protocol requires a communications manager to transmit data to the equipment within the cell. These are shown in Figure 4 as the Alan Bradley DataHighway TM, and the ARCNET asynchronous communications managers.

The interface between the communication managers and the cell control software is as follows:

- Incoming events from equipment controllers are placed on the event queue. This can be done by some type of asynchronous event triggering system call such as an Asynchronous Traps (AST) under VMS, a software interrupt on the Intel 86 family based micro, or a UNIX signal between processes.

- Outgoing communications, resulting from action specified in the state transition tables, are requested by non-blocking interfaces with the communication managers.

Many of the communication management requirements inherent in cell management can be solved by purchasing a commercial software product. Most common factory communications protocols have been implemented on a variety of platforms and are available in the marketplace.

The Event Queue and Event Log

The queueing system is an implementation of the single queue, single server model. This model is deterministic. With some experienced overhead and throughput estimates, boundaries for event processing time can be calculated.

All events for the cell are funnelled through the queue. If debugging is necessary all events are recorded in the event log. The chain of events leading up to a system or production line crash can thus be recreated. The authors have found this feature to be invaluable during early testing and trouble-shooting in automated production facilities. This is especially true when integrating a large number of customized equipment controllers.

This integration task is one which must be done at the cell controller level. Often it is not possible to pre-test vendor supplied equipment prior to installation in a production line, therefore recreation of a fatal sequence of events is invaluable.

The Event Dispatcher

The event dispatcher is the queue server. Its function is twofold:

- To dispatch events to the appropriate equipment management module; and

- To manage any critical section problems inherent with multiple threads manipulating the same data structures.

The critical section problem is managed by disabling asynchronous functions while removing events from the queue.

The Cell Control Manager

The cell control manager is responsible for performing such tasks as MRP data transactions with the Area Host, placing cell initialization events on the queue, initializing the state transition tables and the cell state table, and any other initialization actions which need to be performed.

Equipment Controller Manager

The equipment controller managers are modules that update the cell state table and perform searches on their specific state transition tables. These modules are parameter driven and should be identical. With some modifications to the architecture they could be reduced to a single module that performs table look ups. A drawback is that this simplification may cause greater problems in the areas of debugging, maintenance and adaptability to unique requirements.

This software architecture is parameter driven because of the memory aspect within the sequential state machine model. This is incorporated into the architecture through the cell state table and the state transition tables. This allows the majority of processing inherent in cell management to be reduced to state table and state transition table look ups. Without the finite sequential state machine data model, the software would require complex control flows to implement the event - action pairs required from a cell controller.

It should also be noted that the sequential state machine model reduces the difficulty of mapping the requirements model through the design stages, to the resulting physical implementation of the cell controller.

Summary

The analysis of cell management control requirements is simplified by modelling the cell as a sequential state machine. This state machine model documents the control requirements that must be met by the cell management controller software. The architecture presented here allows the requirements identified by the state machine analysis to be integrated directly into the physical components of the control system. These physical components are the cell state table and each of the state transition tables. This is a significant step towards verifying that the physical implementation will satisfy the requirements that are identified during the initial analysis phase.

The event log in the cell control architecture is invaluable. The featu:e is created by the audit trail left by the single server queuing model. The ability to recreate crashes and identify bugs reduces trouble-shooting efforts.

It is also possible for this architecture to be implemented for maximum code reuse. This would be accomplished by designing a generic equipment controller manager module. This module would be parameter driven by the data in the cell state table and state transition tables.

This paper did not attempt to discuss some additional important aspects of cell management, such as, user interface requirements. In many cases cell management requirements may specify tasks of this nature. However, the architecture presented is quite open and there is room for integration between this and other architectures. The only constraint is that the cell management control software as described here cannot block or perform any time intensive functions while other events are occurring within the cell. As long as this requirement is met, lower priority task can be interfaced with this architecture.

Bibliography

1. Hatley, D.J. & Pirbhai, I.A. *"Strategies for Real-time Systems Specifications"*. Dorset House, 1987.

2. Booth, T.L., *"Sequential Machines and Automata Theory"*, John-Wiley & Son, NY, 1967.

3. Swartout, W. and Blazer, R., *"On the Inevitable Intertwining of Specifications and Implementation*, CACM Vol/25, No. 7, fully 1982.

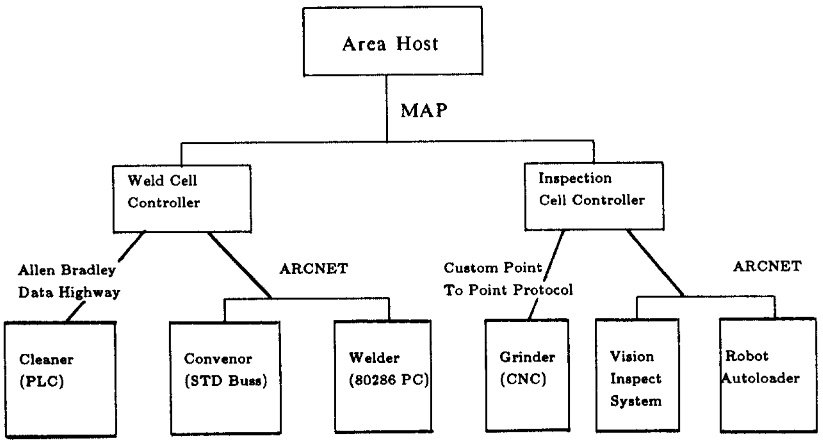

Figure 1.
A Typical Advanced Manufacturing Facility Architecture

Cell State Table

Equip Controller	Current State
Welder	Welding
Cleaner	Cleaning
Robot	Idle
Conveyor	Ready for load

Welder State Transition Table

Current State	Event	Action	Next State
Idle	"load welder"	"initiate robot load cycle"	Loading
Loading	"load done"	"initiate weld cycle"	Welding
Welding	"weld done"	"initiate robot unload cycle"	Unloading
Unloading	"unload done"	N/A	Idle

Figure 2.
Cell State Table and State Transition Tables for the Welder

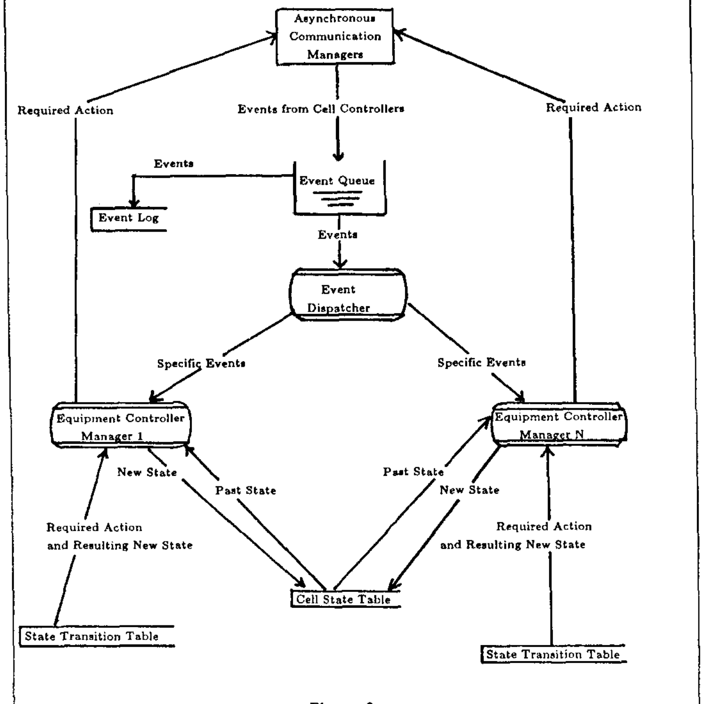

Figure 3.
Graphic of Software Architecture for Cell Controller

Figure 4 presents this architecture as it would be applied to the welding cell controller of Figure 1.

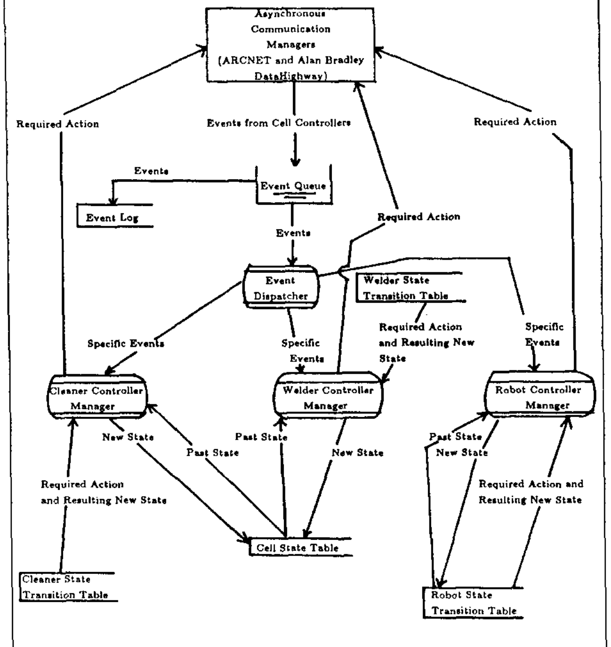
Asynchronous Communication Managers (ARCNET and Alan Bradley DataHighway)
Required Action
Events from Cell Controllers
Required Action
Events
Event Queue
Event Log
Events
Required Action
Event Dispatcher
Welder State Transition Table
Specific Events
Specific Events
Required Action and Resulting New State
Specific Events
Cleaner Controller Manager
Welder Controller Manager
Robot Controller Manager
New State
Past State
Past State
New State
Past State
New State
Required Action and Resulting New State
Required Action and Resulting New State
Cell State Table
Cleaner State Transition Table
Robot State Transition Table

Figure 4.
Graphic of Software Architecture for Welding Cell Controller
From Figure 1.