Incorporating Component-Based Control Software In Manufacturing Engineering Instruction

Yu T. Morton, Douglas A. Troy, George A. Pizza, Osama Ettouney

Miami University, Oxford, OH.

Abstract

Efficient and timely development of control software is a challenge in deploying agile manufacturing systems. This is also a challenge in the academic environment because control software used in student projects must be developed quickly and efficiently. Instructors and students are required to have sufficient programming and software development background to change or modify a piece of control software. This introduces tedious programming tasks into a project and diverts students' focus on issues fundamental to manufacturing systems and processes. A component-based software development approach has been developed and utilized at Miami University that introduces flexible, adaptable, and user-friendly control software for manufacturing work cells. This paper presents the design framework and implementation of the software, as well as preliminary instructional results using the software.

Introduction

Modern manufacturing systems must be flexible, dynamic, and adaptive to meet the market demand[1]. Manufacturing engineering education must adopt new technology and new approaches to address the new challenges. A major problem facing manufacturing engineering education in addressing the problem is the inflexible and programming intensive nature of the control software. This paper addresses our approaches in an attempt to solve this problem by using software components to construct flexible and adaptive manufacturing control software.

Component-Based Software

Component-based software development is a recent approach in software engineering. In the component-based software paradigm, software systems are built with prefabricated software components[2]. These software components are well-defined precompiled building blocks with standardized interfaces and are well separated from their own development environment and from other components. A third party with no knowledge of a component's internal design and implementation can construct complex software systems by assembling software components through the use of visual design tools. Such characteristics make the component-based architecture an excellent choice for developing flexible control software for manufacturing systems[3,4,5]. Using the component-based approach, a set of generic software components can be created and stored in a component library. The desired system can be assembled using appropriately configured software components. Simulated components can be used in place of "real" components for testing and planning purposes. When changes occur in system

specifications, the components can be reconfigured, replaced, or "re-arranged", and interfaces can be reconnected to meet the new requirements. Reprogramming efforts are reduced in the process of adapting a system to a new configuration. The potential benefit of using the component-based software in manufacturing systems could be enormous.

There are a number of techniques and development tools available for creating and integrating software components. We used the Java programming language and Sun Microsystems' JavaBeans ™ component model to design and develop our software components. Java offers many advantages for manufacturing systems, such as being object-oriented, distributed, multi-threaded, having extensive class libraries, and platform independence[6]. Java's introspection mechanism allows a component interface to be exposed to application developers and component integration tools at run time and application design time. A JavaBean software component interacts with other components through Java event objects. An event object is a notification generated by a component whenever there is a change in the component state. A component can register to receive events generated by other components. When an event occurs, methods contained in a listener component can be invoked to execute code contained in the component. Java's delegation event model provides a standard mechanism for a *source* component to generate an *event* and send it to a set of *listener* components. We designed and implemented all of our components using the established mechanism provided by Java's event model and the JavaBeans component model.

Experimental Set Up: the Miami Flexible Manufacturing Work Cell

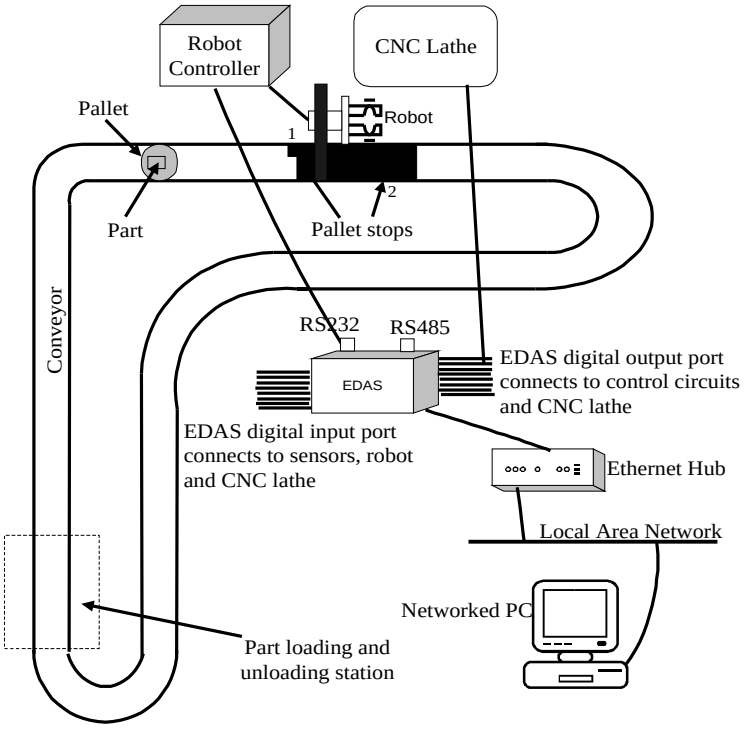

Figure 1. Miami University Flexible Manufacturing Work Cell Layout.

We applied our modeling and design approach to the Miami work cell (Figure 1). The Miami work cell consists of a conveyor, an automatic storage and retrieval system, and a milling station. The station is equipped with a RM-501 Mitsubishi robot, an Emco CNC lathe machine, sensors, control circuits, devices that stop and clamp pallets, and an Intelligent Instrumentation's communication device EDAS (Ethernet Data Acquisition System). The robot performs material handling functions by moving parts between the lathe and pallets. The sensors monitor the arrival of pallets, the presence of parts on pallets, the position of the robot and the state of the robot, and the state of the lathe. The control circuits are used to activate/deactivate pallet stops, clamp/unclamp pallets inside the station, move the robot between conveyor and the lathe, and start and stop the lathe machining cycle. The EDAS communication device provides the interface between the cell and a local area network. A PC connected to the network can be programmed to monitor sensor signals from the cell and set controls to operate the devices and machines at the cell via the EDAS.

State-based Modeling Approach

In order to have a set of generic software components that can be easily configured to adapt to an arbitrary work cell, we developed a state-based approach for modeling work cell control. The approach is inspired by the framework proposed by Adiga and Cogez's for object-oriented modeling of manufacturing system control software[7]. We modified Adiga and Cogez's model to contain three interacting components: (1) a State Table that combines the work cell state information, state transition rules, and physical system parameters, (2) a Decision component that encapsulates generic work cell control logic function and adapts to work with a specific physical system by retrieving and utilizing information stored in the State Table, and (3) a Communication component that provides all interface functions between the decision component and the work cell (Figure 2). Morton el al described the architectural and detailed designs of these software components[8].

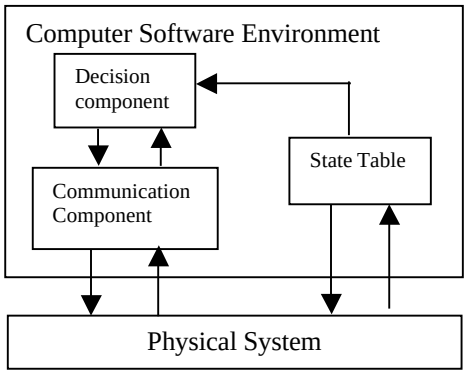

Figure 2. The General Framework for Manufacturing Control System.

Figure 3 depicts the composition of control software using software components and the interactions among these software components. In addition to the Decision, Communication, and State Table, we also implemented two simulated components to support off-line design and testing: SimulatedSensor and SimulatedControl. The simulated components can interact with the Decision and Communication components by providing simulated sensor signals and control

actions. They have been proven to be a useful tool in trouble shooting software and hardware problems in the system. The Start and Stop components are standard Java Swing Buttons. When a user presses the Start or Stop button, a Java ActionEvent will be generated to invoke public methods defined in the Communication component to initiate or terminate communications between the control software and work cell. The EDAS and the work cell are also shown in the figure to indicate the direct interactions between the control software and the manufacturing work cell hardware. The rectangles with rounded corners inside a software component represent public interface methods defined for the software component. Arrows represent events that facilitate interactions and message exchanges between components. The root of an arrow indicates the source of the event, while the head indicates the receiver of the event.

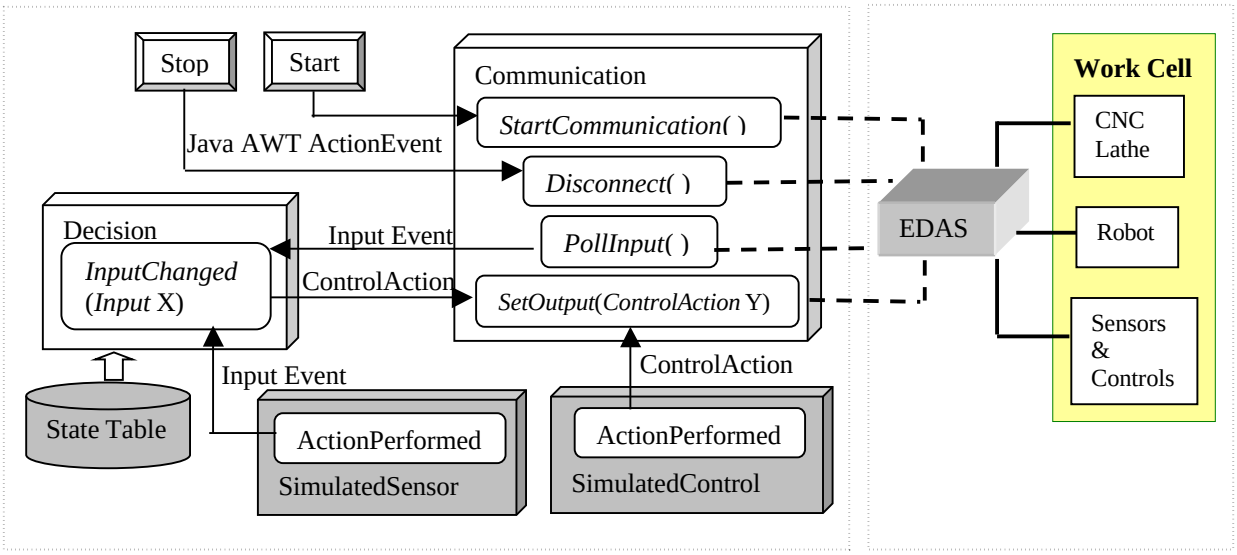

Figure 3. Component-Based Control Software for Miami FMS Work Cell.

Two classes of events are defined to encapsulate the interactions and message exchanges between our software components: Input and ControlAction. Both event class objects contain information that identifies the event object source and receiver. An Input event class object identifies a change in a sensor or a device state and the sensor that generated the change, while a ControlAction event class object carries commands or instructions issued by the control software.

The above software components have been successfully tested using the Miami work cell. The Decision, Communication, SimulatedSensor, and SimulatedControl components are created as pre-packaged software components. The State Table is the only component that contains parameters and data that pertains to specific work cell layout and product operation procedure. The key in applying our approach to manufacturing engineering instruction is a systematic approach to develop the State Table. This approach should emphasize manufacturing systems and processes, while avoiding intensive programming procedures. The following section describes the approach we developed to achieve this goal.

Systematic Approach to Develop the State Table

The State Table contains complete information about a work cell's state information, state transition rules, and physical system parameters. A systematic approach for establishing the State Table for a work cell can be achieved in three stages: (1) establishing a state transition diagram that describes the work cell operation in terms of its state and state transition rules, (2) specifying the relationship between the entities in the state model and the physical system parameters in the form of a look-up table, and (3) combining the state transition diagram and the look-up table to form a State Table that describes the work cell operation state and state transition rules in terms of the physical system parameters.

A typical work cell operation can be described by a sequence of states. At each state, the work cell waits for a specific event to occur. When the event occurs, the cell will perform one or more tasks and then enter the next state. An existing technique for analyzing a system that can be modeled in terms of states and state transitions is the state transition diagram, commonly used in the finite state machine analysis[9]. A state transition diagram uses nodes and arcs that connect the nodes to represent the states of a system and transitions from one state to another, respectively. The state transition rule can be defined by the general form:

Current state and *Input Event* and *Predicates* => *Next state* and *Action*

The event that triggers a transition from one state to another, the conditions of the systems under which the transition is allowed to happen, and the control actions associated with the transition are all presented in the state transition diagram. Students can analyze and determine operation procedures and create a state transition diagram for a given work cell and product. Figure 4 is an example milling procedure state transition diagram at the Miami work cell. The states are denoted as ovals and a number is assigned to each state for quick reference. State transition triggering events, conditions, and actions are marked on the arcs connecting the states in the form of *input/predicates/action*. A lack of a predicate for a state transition is shown in the form of *input*/ /*action* in the diagram.

The work cell operation as outlined by the state transition diagram in Figure 4 has twelve discrete states identified by numbers 0 through 11. In the initial state (state 0), the station is empty. When a sensor signals the arrival of a pallet, a control action deactivates the entrance stop to allow the pallet to move inside the station (state 1). When a second sensor signal verifies that the pallet has stopped inside the station, a control signal activates a clamp to lock the pallet in position (state 2). When a sensor signal indicating that the pallet has been secured, the system examines whether or not a part is on the pallet. If the part is present, the control action will send a command to the robot to pick up the part from the pallet (state 3). If a part is not detected on the pallet, the control action will deactivate pallet stop 2 to let the pallet out of the station (return to state 0). At state 3, the system waits for a signal from the robot to indicate that it has completed its task of picking up the part from the pallet. Once the signal is received, the robot will be moved to the lathe (state 4). The rest of the operation is self-explanatory from the transition diagram and will not be discussed in detail here.

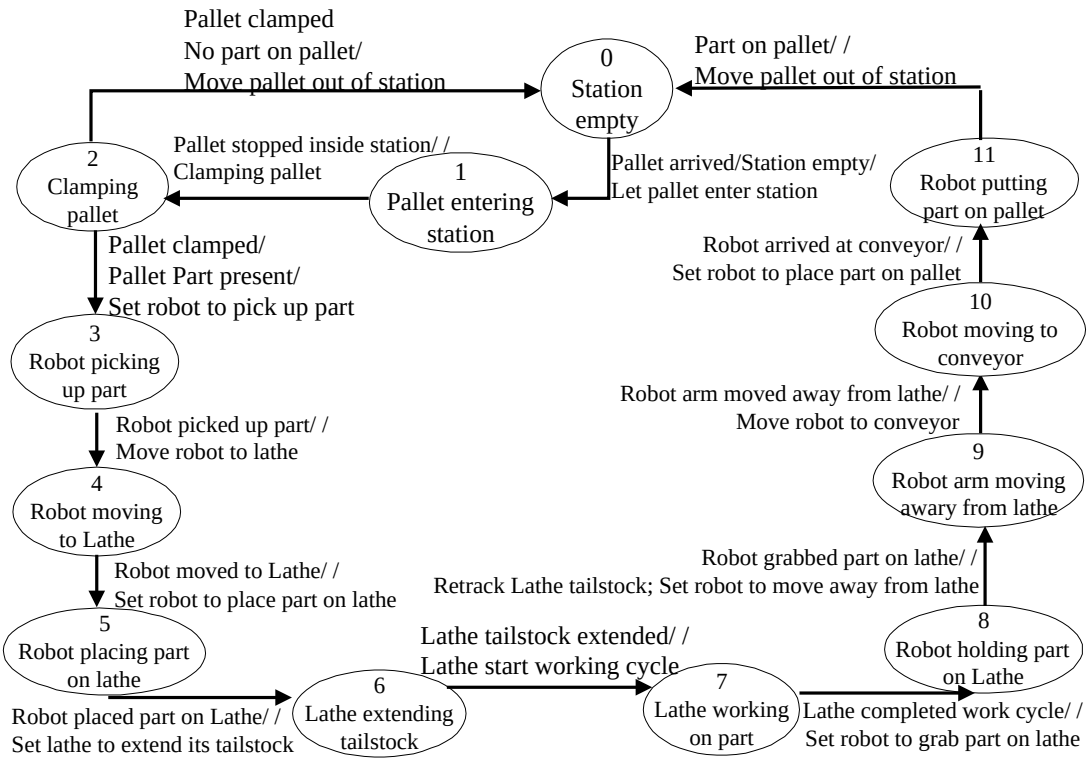

Figure 4. State Transition Diagram for MU FMS Work Cell Operation.

The above work cell operation and state diagram are specific to the Miami work cell. Many of the underlying mechanisms, however, are fundamental to all work cells. For example, the sensors that monitor the work cell translate the physical changes in the system into digital signals for the control computer via the communication device. Control actions are sent from the control computer as instructions to devices and equipment in the work cell. From the perspective of the control computer, the input sensor signals and output control actions are generic digital signals that are common to all systems. What makes one system different from another is the interface connecting the sensor and control circuits to the ports and channels of the communication device, the system and signal protocols, and the control logic of the work cell operation. For example, in the Miami work cell, a sensor located at pallet stop 1 is connected to the bit 4 of an input port in the EDAS. If a pallet arrives at pallet stop1, the bit 4 of the EDAS input port receives a digital high signal. This signal corresponds to the *pallet arrived* input event in the state transition diagram in Figure 4. Our approach in mapping the events, conditions, and actions in the state model to physical parameters is to create a look-up table such as Table 1.

The State Table is obtained by combining the information provided by the state transition diagram and the look-up table. Table 2 is the State Table developed for the milling procedure at the Miami work cell using this approach. The State Table provides all the necessary system specific information needed by the decision component. Currently, Table 2 is created using a manual procedure from the state transition diagram and the look-up table. It is feasible for instructional purposes since most laboratory work cell contains limited number of sensor, and control functions. For industrial applications, it would be necessary to develop computer-aided tools to perform this procedure.

Work cell parameter	State model indicators
Input port 0, bit 0 signal high	Robot arrived at the lathe
Input port 0, bit 1 signal high	Robot arrived at conveyor
Input port 0, bit 2 signal high	Station open
Input port 0, bit 3 signal high	Lathe status intermediate stop
Input port 0, bit 4 signal high	Pallet arrived at station
Input port 0, bit 5 signal high	Pallet inside station
Input port 0, bit 6 signal high	Lathe tailstock extended
Input port 0, bit 7 signal high	Part present on pallet
Input port 2, bit 0 signal high	Robot completed a task
Output port 1, bit 0 set to high	Move robot to conveyor
Output port 1, bit 1 set to high	Move pallet out of station
Output port 1, bit 2 set to high	Move robot to the lathe
Output port 1, bit 3 set to high	Close pallet station
Output port 1, bit 4 set to high (low)	Deactivate (activate) pallet stop 2
Output port 1, bit 5 set to high (low)	Deactivate (activate) pallet stop 1
Output port 1, bit 6 set to high	Start the lathe cycle
Robot1.txt	Robot picks up part from pallet
Robot2.txt	Robot places part on the lathe
Robot3.txt	Robot grabs part on the lathe
Robot4.txt	Robot arm moves away from the lathe
Robot5.txt	Robot places part on pallet

Table 1. Miami Work Cell Milling Procedure State Model Attributes Look-up Table.
(The port and bit numbers in the table refer to the EDAS I/O ports and channels.)

Current State	Next State	Input Event	Predicates	Control Actions
0	1	Port 0 bit 4 high	Port 0 bit 5 low	Port 1 bit 5 high, bit 0,1,2,3,5,6 low
1	2	Port 0 bit 5 high	None	Port 1 bit 0,3,4,5 high, bit 1,2,6 low
2	0	Port 0 bit 2 low	Port 0 bit 7 low	Port 1 bit 0,1 high, bit 2,3,4,5,6 low
2	3	Port 0 bit 2 low	Port 0 bit 7 high	Port 1 bit 0,3 high, bit 1,2,4,5, 6 low Send Robot1.txt through serial port to robot controller
3	4	Port 2 bit 0 high	None	Port 1 bit 2,3 high, bit 0,1,4,5,6 low
4	5	Port 0 bit 0 high	None	Port 1 bit 3 high, bit 0, 1,2,4,5,6 low Send Robot2.txt through serial port to robot controller
5	6	Port 2 bit 0 high	None	Port 1 bit 3, 6 high, bit 0,1,2,4,5 low
6	7	Port 1 bit 6 high	None	Port 1 bit 3, 6 high, bit 0,1,2,4,5 low
7	8	Port 0 bit 3 high	None	Port 1 bit 3 high, bit 0, 1,2,4,5,6 low Send Robot3.txt through serial port to robot controller
8	9	Port 2 bit 0 high	None	Port 1 bit 3, 6 high, bit 0,1,2,4,5 low Send Robot4.txt through serial port to robot controller
9	10	Port 2 bit 0 high	None	Port 1 bit 0, 3 high, bit 1,2,4,5,6 low
10	11	Port 0 bit 1 high	None	Port 1 bit 3 high, bit 0,1,2,4,5,6 low Send Robot5.txt through serial port to robot controller
11	0	Port 0 bit 7 high	None	Port 1 bit 1 high, bit 0,2,3,4,5,6 low

Table 2. The State Table for Miami Work Cell Milling Procedure Operation.

Proceedings of the 2002 American Society for Engineering Education Annual Conference & Exposition

Conclusions

We adopted and modified Agida and Cogez's framework to model component-based work cell control software in terms of three basic components: a State Table containing the work cell operation state transition rules and system parameters, a Decision component responsible for the control logic of the work cell operation, and a Communication component representing the interface between the decision component and the physical system. Separating the control logic from interface functions made it possible to design and implement the decision component as a generic software component. This generic component retrieves system specific information from the State Table stored in a database on the control computer and applies the information to its control logic. The use of the State Table provides a flexible way to upgrade and change the control software. Instead of reprogramming the software components, changes in work cell operation procedure and equipment updates can be absorbed by modifications of the State Table data, allowing the control software to be adaptive in a dynamic environment. The Communication component is hardware dependent. Our implementation of the communication component for the EDAS provides a prototype for other similar devices.

The only component that is dependent on specific work cell layout and operation procedure is the State Table. We developed a systematic approach to generate the State Table. In this approach, students analyze the work cell layout and required operation procedure. Based on the analysis, students can develop a state transition diagram for the operation procedure and a look-up table that matches the work cell physical parameters with various indicators such as the conditions, predicates and controls appearing in the state transition diagrams. From the state transition diagram and the look-up table, the State Table can be derived manually for instructional work cell. This procedure requires students to develop a thorough understanding of manufacturing systems, methods, and processes, while avoids unnecessary programming effort.

This method was developed and tested with an undergraduate student via a summer independent study. The student had no prior knowledge of manufacturing systems. Upon completion of the project, the student developed excellent understanding of the work cell operation, sensors and control hardware functions, and the manufacturing methods and processes involved in the project. We plan to introduce this method to a senior level undergraduate course on Computer-Integrated Manufacturing at Miami in the fall of 2003.

Bibliography

1. NGMS-IMS (Next Generation Manufacturing Systems-Intelligent Manufacturing System) Research Reports, "Scalable flexible manufacturing." *Advanced Manufacturing*, http://www.advancedmanufacturing.com/March00/research.htm, March, 2000.
2. Szyperski, C., Component Software, Addison-Wesley, 1998.
3. Buschmann, F., A. Geisler, T. Heimke, and C. Schuderer C., "Framework-based software architectures for process automation systems." *Annual Reviews in Control*, 24, P163, 2000.
4. Edwards, J., P. Clements, J. Gascoigne, and I. Coutts, "Component-Based Systems: the basis of future manufacturing systems." *Component-based Software Engineering*, Edited by Thomas Jell, P105, 1998.

5. Naylor, A.W. and Volz, R.A., 1987, "Design of integrated manufacturing system control software." *IEEE Trans. on Systems, Man and Cybernetics*, 17, pp.881, 1987.
6. O'Neil J., *JavaBeans Programming from the Ground Up*, Osborne McGraw-Hill, 1998.
7. Adiga, S., and Cogez, P., "Towards an object-oriented architecture for CIM systems." in S. Adiga (ed) *Object-oriented Software for Manufacturing Systems*, pp. 44-64, 1993
8. Morton, Y. T., Troy, D. A., and Pizza, G. A., "An Approach in Developing Component-based Control Software for Flexible Manufacturing Systems." *Proceedings of American Control Conferences*, Anchorage, Alaska, May, 2002.
9. Schach, S. R., Classical and Object-oriented Software Engineering, 4th edition, McGrill Hill, 1999.

Dr. YU MORTON is an Assistant Professor at the Manufacturing Engineering Department at Miami University. Her research areas are software radio techniques, radar and satellite remote sensing, and component-based control software for manufacturing systems. She holds a BS degree from Nanjing University, MS degree from Case Western Reserve University, MS degree from Miami University, and Ph.D. from the Pennsylvania State University.

Dr. DOUGLAS TROY is a Professor and Chair of the Computer Science and Systems Analysis Department at Miami University. His research interest is in programming for automated manufacturing and software development environments. He holds a BS degree from Miami University, MS from Ohio State University, and PhD from The University of Waikato.

GEORGE PIZZA is a junior Computer Science and Systems Analysis major at Miami University. His interests are in artificial intelligence and software design and development. Tony will graduate in May of 2003 after which he plans to work in the IT field and possibly attend graduate school.

Dr. OSAMA ETTOUNEY is a Professor and Chair of the Manufacturing Engineering Department at Miami University. His research areas are control systems, engineering design, and computer-integrated manufacturing. He received a BS degree from University of Cairo, MS degree from the MIT, and Ph.D. from the University of Minnesota.