IIE Transactions (1997) **29**, 901–909

An agile manufacturing workcell design

ROGER D. QUINN[1], GREG C. CAUSEY[1], FRANK L. MERAT[2], DAVID M. SARGENT[2], NICK A. BARENDT[2], WYATT S. NEWMAN[2], VIRGILIO B. VELASCO JR[2], ANDY PODGURSKI[3], JU-YEON JO[3], LEON S. STERLING[3] and YOOHWAN KIM[3]

[1]*Department of Mechanical and Aerospace Engineering, Case Western Reserve University, Cleveland, OH 44106, USA*
[2]*Department of Electrical Engineering and Applied Physics, Case Western Reserve University, Cleveland, OH 44106, USA*
[3]*Department of Computer Engineering and Science, Case Western Reserve University, Cleveland, OH 44106, USA*

Received October 1995 and accepted August 1996

This paper introduces a design for agile manufacturing workcells intended for light mechanical assembly of products made from similar components (i.e., parts families). We define agile manufacturing as the ability to accomplish rapid changeover from the assembly of one product to the assembly of a different product. Rapid hardware changeover is made possible through the use of robots, flexible part feeders, modular grippers, and modular assembly hardware. The division of assembly, feeding, and unloading tasks between multiple robots is examined with prioritization based upon assembly time. Rapid software changeover will be facilitated by the use of a real-time, object-oriented software environment utilizing graphical simulations for off-line software development. An innovative dual VMEbus controller architecture permits an open software environment while accommodating the closed nature of most commercial robot controllers. These agile features permit new products to be introduced with minimal downtime and system reconfiguration.

1. Introduction

1.1. *Definition of agile manufacturing*

Agile manufacturing is a term that has seen increased use in industry over the past several years. The definition of ‘agile’, however, is not clear, nor is it consistent: “Agility: The measure of a manufacturer’s ability to react to sudden, unpredictable change in customer demand for its products and services and make a profit” [1]; “Today factories are coming on line that are agile at tailoring goods to a customers requirements, without halting production . . .” [2]; “Agile manufacturing assimilates the full range of flexible production technologies, along with the lessons learned from total quality management, ‘just-in-time’ production and ‘lean’ production” [3]. The only common thread among the various definitions is the ability to manufacture a variety of similar products based on what may be rapidly changing customer needs.

A definition of ‘agile’ manufacturing has been adopted that applies to light mechanical assembly of products made from components in parts families: *Agile manufacturing is the ability to accomplish rapid changeover between the manufacture of different assemblies.* Rapid changeover, further, is defined as the ability to move from the assembly of one product to the assembly of a similar product with a minimum of change in tooling and software. A corollary of this definition of agility is that agile manufacturing should also allow for the rapid introduction of new parts. Agility in manufacturing opposes the prevailing mass-production, Fordist [4], paradigm, characterized by the methods championed by Henry Ford: high-volume production of low-cost, standardized products. This contemporary paradigm shift is motivated by the ever increasing competition seen in all industries and by a more demanding customer base.

Many researchers are focusing on the management and organizational aspects of agile manufacturing. Before agile manufacturing can have an appreciable effect, however, the capital equipment must be up to the challenge: “How agile a manufacturing enterprise can be is in direct proportion to the dexterity of its machines. In general, a company’s people are more adaptable to change than its manufacturing machine tools” [5]. Rapid changeover enables assembly hardware to be flexible as well as to make the production of small lot sizes economic. This is accomplished through the use of robust, reusable software; quick-change grippers for the robotic manipulators; modular worktables; and parts feeders that are flexible enough to handle several types of part without needing mechanical adjustment. These feeders use vision, in place of hard fixturing, to determine the position and orientation of parts. Generic, reusable pose estimation vision routines permit new parts to be added to the system with a minimum of effort.

A testbed implementation of an agile manufacturing workcell has been developed (Fig. 1) that includes me-

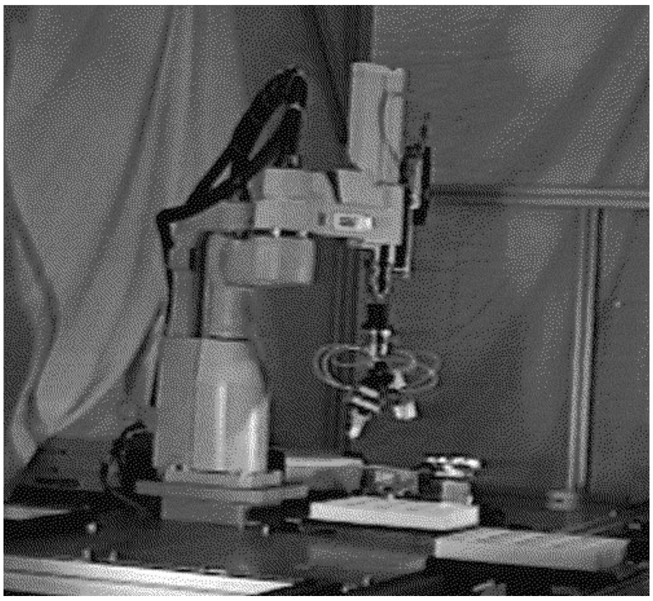
Fig. 1. Agile workcell.

chanical manipulators, flexible part feeders, a vision system (cameras, frame grabber, and a library of image processing routines), as well as a limited number of dedicated sensors and actuators. A workcell controller integrates and synchronizes the operation of the individual components.

1.2. *Scope and importance of work at Case Western*

Several companies have implemented what may be considered 'agile' manufacturing. For example, Motorola has developed an automated factory with the ability to produce physically different pagers on the same production line [6]. At Panasonic, a combination of flexible manufacturing and just-in-time processing is being used to manufacture bicycles from combinations of a group of core parts [7]. Against the backdrop of such work, the Case Western Reserve University (CWRU) workcell is innovative in several ways. The use of vision-guided, flexible-parts feeders is one example. Another is the object-oriented design of the software. The over-arching design philosophy of quick changeover, however, is what makes this workcell particularly novel. The CWRU workcell has been designed to be a versatile production facility, amenable to a wide range of light manufacturing applications.

2. Workcell hardware

The agile workcell developed at CWRU consists of a Bosch flexible automation system, multiple robots, as many as four flexible part feeders per robot, and an Adept MV controller with an AdeptVision System. Small Adept robots were chosen for their cost effectiveness. The robots are mounted on pedestals near the conveyor system. Pallets with specialized parts fixtures carry assemblies throughout the system. Finished assemblies are removed from the pallets by an unloading robot. A safety cage encloses the entire workcell, serving to protect the operator as well as providing a structure for mounting overhead cameras.

2.1. *Conveyor system*

The conveyor system used in the CWRU workcell is a Bosch model T2. Pallets are circulated on two main conveyor sections. These sections are straight and parallel to each other, operating in opposite directions. Pallets are transferred between these two sections by means of lift transfer units (LTUs). These allow for the circulation of pallets around the conveyor system and the capability to 'shuffle' or re-order the pallets.

Each of the pallets in the system has a unique identification number, allowing the system to track and direct their progress. Stops are mounted at critical points on the conveyor to control the flow of the pallets.

An innovative design feature is the short 'spur line'. A spur line (see Fig. 3) is simply an extension of the conveyor, perpendicular to the main line (analogous to a railroad spur), which is used to remove pallets needed at an assembly station from the main conveyor. This allows the flow of the main conveyor line to be maintained while a robot performs an assembly at the spur. Owing to space constraints, optical proximity sensors are used to detect the presence of a pallet on a transfer station instead of the standard rocker and inductive proximity sensors.

2.2. *Assembly stations*

Each assembly robot is surrounded by two modular, removable work tables and two fixed feeding tables (Fig. 2). The modular tables are easily exchangeable, allowing for specialized assembly hardware to be placed within the robot's work envelope (Fig. 3), and contain pneumatic actuators and electrical sensors with quick connectors allowing for the rapid change of any specialized tooling required for a given assembly. As part of the rapid changeover procedure, the modular work tables are registered in the robot's world coordinate system by an arm-mounted camera. The feeding tables are fixed, and the horizontal parts-feeding conveyors are mounted to them.

One drawback of the conveyor/spur system is the time required to exchange a full pallet for an empty one (approximately 15 seconds). During this time the robot could conceivably be inactive. An elegant solution to this problem is a mini-warehouse: a fixture is located on the modular portion of the work table to hold a few completed assemblies. During a pallet swap, the robot can continue the assembly operation, placing the completed assemblies in the mini-warehouse, while the incoming

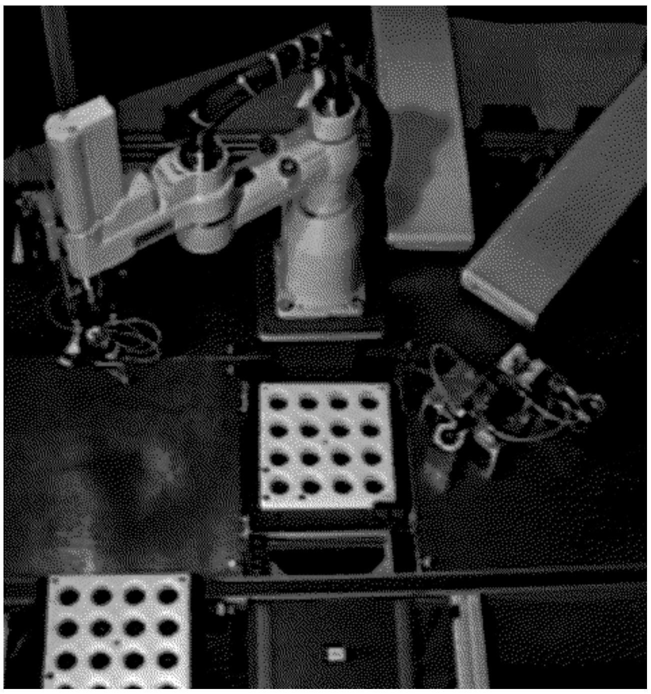

Fig. 2. Overhead of a workstation.

pallet arrives. After the incoming pallet is transferred to the spur, the vision system registers the pallet in the same manner as the modular work tables (i.e., an arm-mounted camera). The robot then places the current assembly (still in its gripper) on the pallet and then proceeds to move the completed assemblies from the mini-warehouse to the pallet.

Several workcell layouts were examined, varying in their placement of the robot relative to the spur (and thereby the pallet). The first layout examined, shown in Fig. 3, places the robot facing the spur with the pallet centered in its work envelope. The parts feeders enter the work envelope of the robot from the rear on both sides. The second layout examined (Fig. 4) placed the robot next to the spur, with the robot facing away from the main line of the conveyor. The pallet was located to the right side of the robot's work envelope with the feeders located to the front and left side of the robot. The final layout examined (Fig. 5) placed the robot in front of the spur (as in the first layout) but rotated by 90°. The pallet would be located on the right side of the work envelope, the feeders would be placed to the left side of the work envelope, and the assembly area would be directly in front of the robot.

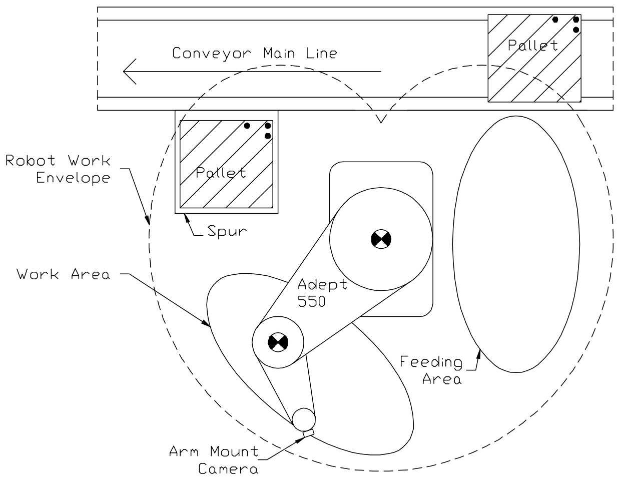

Fig. 4. Layout concept 2.

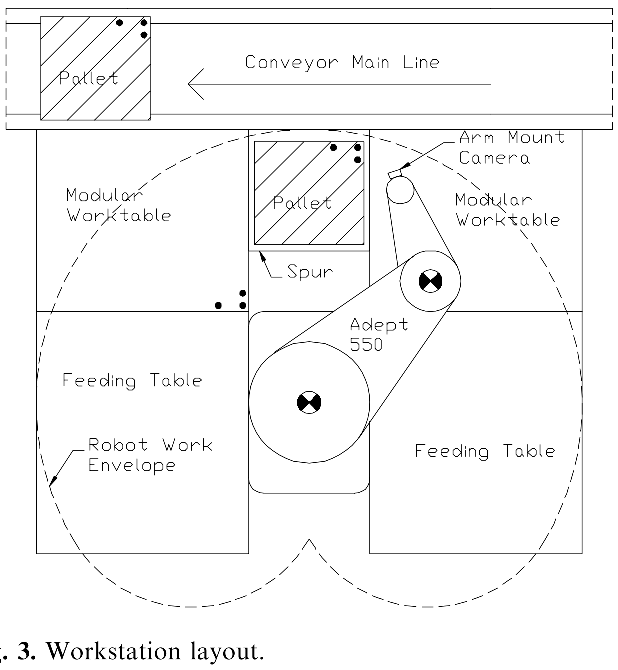

Fig. 3. Workstation layout.

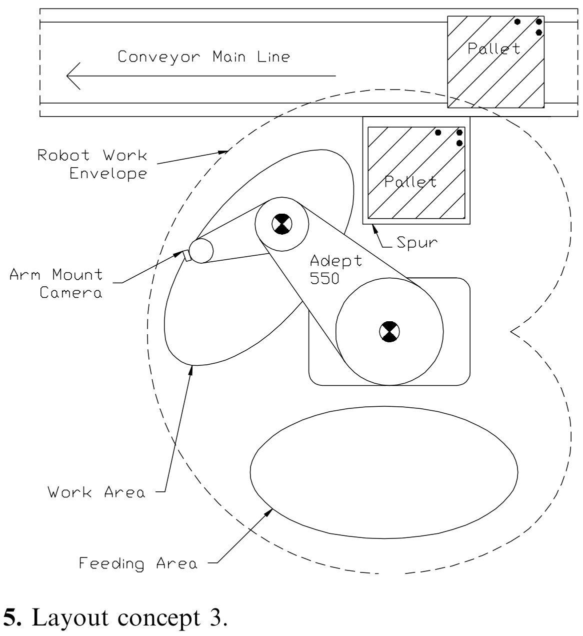

Fig. 5. Layout concept 3.

After evaluating several features of each option, including the placement of the robots relative to the conveyor, the orientation of the robots, the impact of feeder placement relative to the robot work envelope, and the robot motions necessary for a generic assembly given a particular envelope layout, it was determined that the first layout would best suit our needs. This layout yielded the best use of the robot's work envelope while also reducing the amount of motion for a generic assembly. We define a *generic assembly* as a series of movements between various parts feeder locations, assembly locations, and pallet locations that would typify an assembly task.

2.3. *Assembly procedure*

Currently, we are testing the system by using a small assembly consisting of four plastic components (Fig. 6). In our case, the first component, part A, is used as the base to which the other three components are attached. Part B is snapped onto the exterior of the base component. The A/B subassembly must then be inverted. The last two components, part C and part D, are inserted into the bottom of this sub-assembly, with a special guide being used to insert the last component. This process typifies the type of 'light' assembly task for which our workcell was developed.

Several concepts were generated for the assembly procedure. For this assembly, consisting of only four parts, we assumed that no more than two robots would be used. One factor examined was the division of labor. For example, in case 1, each robot could perform an entire assembly task or, in case 2, the robots could each perform part of the task. We also examined a third case in which one robot would be dedicated to parts feeding and another robot would be dedicated to assembly.

Examining the example assembly, a natural division of labor would be to split the job in halves: one robot could attach part B to part A, then a second robot would take this sub-assembly and insert parts C and D. To test this concept we programmed an Adept SCARA robot to emulate the motions necessary for an assembly. For case 1, the time required for each assembly move was recorded, and the time required for gripper changes was estimated. Because four parts were to be manipulated in case 1, a gripper change was necessitated. For case 2, the assembly was simulated as two separate subassembly tasks and the larger of the two sub-assembly times was used as the assembly cycle time (i.e., the time between assembly outputs). In this case, no tool change was required as each robot only handled two parts. Two grippers on a single rotary wrist (Fig. 7) let the robot handle the two parts without a gripper change. We found that two robots working in tandem could produce lower overall cycle times than two robots working independently. This resulted mainly from the time required to perform a tool change when the robots operated independently as well as the added motion required to grasp parts from both the left and right side feeders. Thus, lower cycle times can be achieved by using two robots working in tandem.

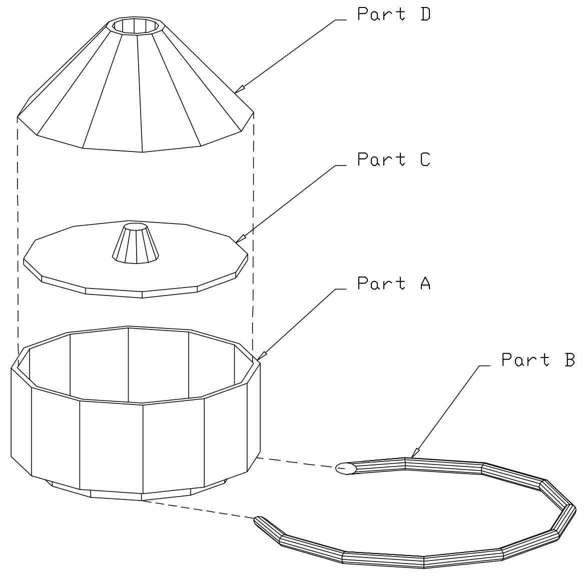

Fig. 6. Example assembly.

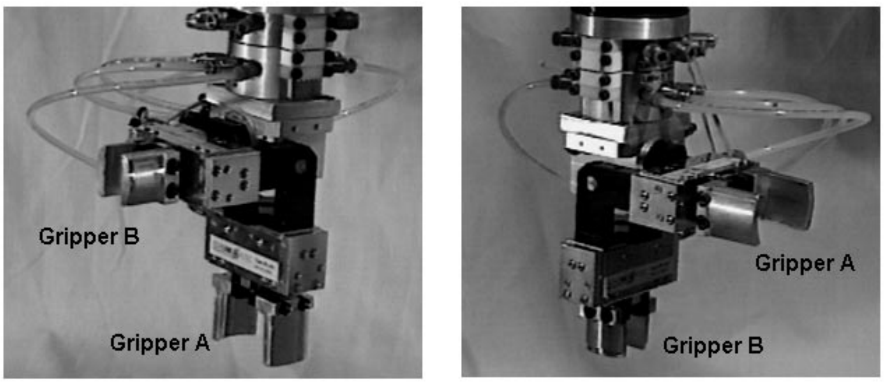

Fig. 7. Multiple grippers on a rotary wrist.

A third case, wherein one robot would be used for parts feeding and another would be dedicated to assembly, was also examined. However, this approach was rejected because space constraints on the pallet fixtures would limit throughput and increase cycle times.

2.4. *Flexible parts feeders*

In keeping with the goal of rapid changeover, the parts feeders need to present a wide variety of parts to the workcell with a minimum of mechanical alteration. Currently, most feeding for automated assembly is performed by using vibratory bowl feeders. Although this is effective, it lacks the flexibility needed for agile manufacturing. Reliability is also an important issue: "Studies show that custom vibratory feeders are responsible for as many as half of all failures in automation systems" [8]. In an attempt to overcome these problems, we have designed a novel parts feeder.

The flexible feeder design consists of three conveyors (Fig. 8). The first conveyor is inclined and lifts parts from a bulk hopper. The second conveyor is horizontal and transports the parts to the robot. An underlit translucent conveyor belt presents part silhouettes to the robot's vision system, which then selects parts that are suitably oriented for pickup. An array of compact fluorescent lights is installed within each of the horizontal conveyors to provide a lit background on which the parts produce a clean binary image. The third conveyor returns unused or unfavorably oriented parts to the bulk hopper. Proper functioning of the feeders depends on the parts' being lifted from the bulk hopper in a quasi-singulated manner. Many factors influence the effectiveness of the inclined conveyor; i.e., the angle of the conveyor with respect to the horizontal, the belt properties (e.g., coefficient of friction), the type of belt (cleated, magnetic, vacuum), and the linear speed of the belt.

When the feeder is to be used for a different part (i.e., there is a changeover) the bulk hopper is emptied and filled with the new part. If the parts are of similar geometries, no changes to the feeding system are typically needed. Some parts, such as circular or cylindrical ones (i.e., parts that would roll back down the incline) may require a different belt surface (e.g., one with cleats) or a different angle of inclination for the inclined conveyor.

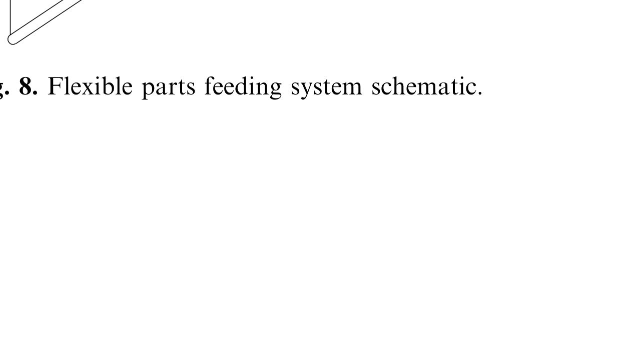

Fig. 8. Flexible parts feeding system schematic.

2.5. *Vision system*

One essential function of the vision system is to determine the position and orientation (pose) of parts in the flexible parts feeders, eliminating the need for conventional mechanical feeders (e.g., bowl feeders). Pose estimation is performed by using built-in functions of the AdeptVision software, and must be fast enough not to degrade the assembly cycle-time. Parts on the feeder belts are examined with binary vision tools. First, the vision system determines if a part is graspable (i.e., the part is in a recognized, stable pose and enough clearance exists between the part and its neighbors to grasp it). Secondly, the pose of the part in the robot's world coordinates is determined. This pose, and the motions associated with acquiring the part, are checked to make sure that they are entirely within the work envelope of the robot. A secondary function of the vision system is to register pallets and modular work tables to a robot's world coordinate system, avoiding the need for alignment hardware and facilitating rapid changeover. Although not a part of our current work, we plan to use vision for error recovery, whereby the cameras can be used to inspect critical points in the system, or assemblies in process.

The vision processing is currently performed on an AdeptVision processor. The vision system uses eight standard CCD cameras, mounted above the flexible parts feeders and on the robot arms. It would be possible to use one arm-mounted camera at each assembly station for all parts feeding and pallet and modular table registration. However, this would have a detrimental effect on workcell throughput and feeder efficiency. Also, varying focal lengths and *f*-stops would require the use of a servo-controlled zoom lens if only arm-mounted cameras were used. Because this would incur substantial penalties in both cost and robot payload limitations, the decision to use multiple cameras was made. Because the number of camera inputs to the AdeptVision system is limited to four, a low-cost custom video multiplexer was developed, utilizing a monolithic MAXIM441 video-switcher that allows up to four cameras to be attached to each video system input.

The AdeptVision System can process only one image at a time. Any conflict between multiple vision operations causes image corruption, so a vision scheduler that is responsible for all vision operations was developed. To best utilize the AdeptVision resources, requests for vision processing are queued by the vision scheduler by using a multi-scan queuing mechanism. The scheduler inspects the request queue whenever the vision processor becomes available. If multiple requests are queued, the vision scheduler selects the most critical one for service. For example,

let Q denote the request queue and let the 'critical robot' be that which has the longest estimated time to completion for its current assembly. The vision scheduler examines the request queue in the following manner:

If Q contains only one entry, service it immediately.
Q is inspected in the following order, with requests from the 'critical robot' receiving priority:
(1) pallet/worktable registration requests;
(2) locate next part for assembly requests;
(3) other requests.

If a vision operation fails, it is retried. After the maximum number of retries has been reached, the request is requeued as a request from a 'non-critical robot', lowering its priority, and allowing other requests to be serviced.

In keeping with the quick-changeover philosophy, the vision routines are designed to be reusable; i.e., a given routine may be used to locate several different but similar parts (e.g., similar symmetries/asymmetries, topology). This approach has many advantages, including minimizing the number of software routines. In addition, this reusability allows for software modularity and 'agility' [9]. For example, by parametrizing the features that a routine searches for, a software vision recognition routine can be applied to parts that have a similar profile but are of different sizes. This means that parts with geometries similar to those already in the software library can be added to the system by simply modifying the inspection procedures that call these lower-level, reusable routines. This approach readily lends itself to object-oriented programming techniques wherein a general recognition routine may be defined as an object class.

2.6. *Introduction of new parts*

New assembly operations involving previously used parts require only software modifications of existing assembly routines. However, modifying an assembly procedure to incorporate a new part involves a few well-defined tasks. First, a vision routine that determines the pose of the part must be developed, utilizing the library of reusable vision routines. If the new part has characteristics that appear nowhere else in the parts library, new reusable routines may need to be added to the software library. Secondly, if the part has not been designed for use on the generic parts feeders (e.g., it has few or no stable poses, as in the case of a cylinder), the feeders may require a different belt or a change in the angle of inclination for parts with multiple stable poses. Thirdly, a new gripper design may be necessary to manipulate the new part. To minimize specialized hardware and avoid tool changes during assembly, this last step should be performed concurrently with the gripper designs for all other parts to be assembled at a given robot. For instance, if a given operation requires both part A and part B to be assembled at the same robot, the gripper designer should take this into account. The grippers for a given assembly would ideally also be designed concurrently with the components, allowing both designs to be iteratively refined to optimize performance and reliability. Concurrent engineering as well as other recent technologies, such as rapid prototyping and solid modeling CAD systems, facilitate agile manufacturing from the design stage all the way through production and quality control [10,11].

2.7. *Computer hardware/controller design*

The current software has been developed entirely in the proprietary V+ programming language and operating system, on Adept's MV controller. For most industrial applications, this programming environment would be sufficient; however, it lacks the power and flexibility needed to support rapid software development and changeover, i.e., agility. This is largely because V+ lacks features that are standard in other languages and operating systems, such as user-definable functions in the manner of C/C++, standard data structures, and a well-developed shell script language.

To circumvent these limitations, a more extensive controller interface design is under development that will allow the system to support C and C++, and provide a friendlier and more flexible user interface. In addition, it will allow the use of a commercial real-time operating system, thus simplifying software development and further increasing agility.

This new controller interface will use a second VMEbus in addition to the standard MV robot controller VMEbus (Fig. 9). This second VMEbus houses I/O boards and dedicated single-board computers (SBCs), running under a commercial real-time operating system. C and C++ programs running on the SBCs are respon-

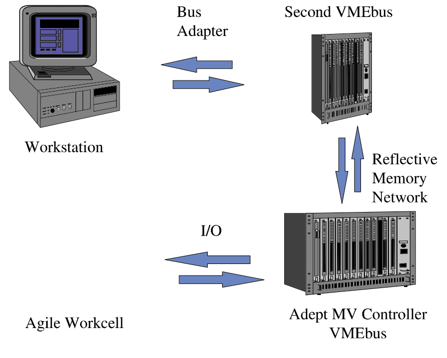

Fig. 9. System architecture.

sible for all high-level control and assembly sequencing (e.g., conveyor control, pneumatic operations, specifying robot destinations), whereas the MV controller is used exclusively for low-level robot motions (e.g., servo control and trajectory generation) and some machine vision routines. The combination of VME, a commercial real-time operating system (RTOS), and C++, being *de facto* standards, helps to provide the workcell controller with an open architecture [12]. A second vision-processing board can also be used on the second VMEbus, augmenting the AdeptVision system.

The two buses are connected by a reflective memory network consisting of two memory cards, one on each bus, which can be connected by either a cable or a fiber-optic link. Changes made to memory on one board are automatically reflected on the other, thus allowing commands and data to be transmitted between the two buses [13]. The SBCs can place robot and vision commands on the reflective memory network to be read by a set of command servers running on the MV controller (i.e., utilizing shared memory). The servers execute the commands and, where applicable, return the results via the same network. A testbed version of this architecture has successfully controlled components of the workcell. A full implementation is expected to be completed in the near future.

3. Workcell software

The flexibility of an agile manufacturing system is provided largely by its software. However, this flexibility does not come without careful design. Although software is inherently easier to change than hardware, the structure of a software system can degrade after repeated modification, leading to poor reliability and increased maintenance costs. In designing the workcell control software we have employed software engineering methods and tools that support the principle of design for change. In particular, our software is object-oriented, that is, it is based on identifying the objects of the system, which are those entities having a state and a behavior. Physical devices, abstract data structures, and entire subsystems are modeled as objects that provide a well-defined set of services whose implementation is encapsulated and hidden.

3.1. *Operating system*

The initial versions of the workcell control software were implemented with the V+ operating system and programming language provided with the Adept MV controller. Although V+ provides adequate facilities for many robotic applications, a more advanced operating system and programming language was necessary to support our software design philosophy and the goals of agile manufacturing. In general, workcell control involves the management of a number of concurrent tasks with real-time constraints. An RTOS with facilities for task scheduling, communication, and synchronization is necessary.

3.2. *Software architecture*

The workcell control software is designed as a hierarchy of servers. At the highest level, the workcell controller services requests from the human operator for crates of finished assemblies. In performing the task it communicates with subordinate servers. In general, servers are designed with as few assumptions about the overall workcell structure as possible, so that they are not sensitive to changes in that structure. Where appropriate, servers operate concurrently.

Error handling is also hierarchical. If a server encounters an error condition, it first tries to resolve it locally, e.g., by making additional requests to subordinate servers. If this fails, the server indicates to its client that it was unable to provide the requested service. The client then tries to resolve this error condition. The unresolved error continues to rise up the hierarchy to servers with increasing spheres of influence. In the absence of redundant servers for the unresolved error in question, the controller will inform the operator of a problem requiring human intervention.

3.3. *Workcell simulation*

As the software development progressed concurrently with the construction of the hardware system, it became evident that an emulation of the expected hardware system would be extremely useful. We began development of a comprehensive simulation that would permit the workcell control code to be developed and tested without using the actual hardware and debugged without halting the production of a functioning workcell.

Graphical simulation was also used (Fig. 10) to facilitate visualization of the hardware operation. This was especially useful for investigating various workcell layouts.

The conveyor system has been successfully simulated, and detailed simulations of the robots and vision system are under development. The simulation code mimics the inputs and outputs of the workcell, allowing for transparent use of the simulation. In other words, the code used to control the simulated workcell is the same code that is used to control the actual workcell. This is a powerful tool for software design, because there are no inconsistencies between the simulation control code and the actual control code, eliminating possible porting problems in moving from the simulation to the actual control platform.

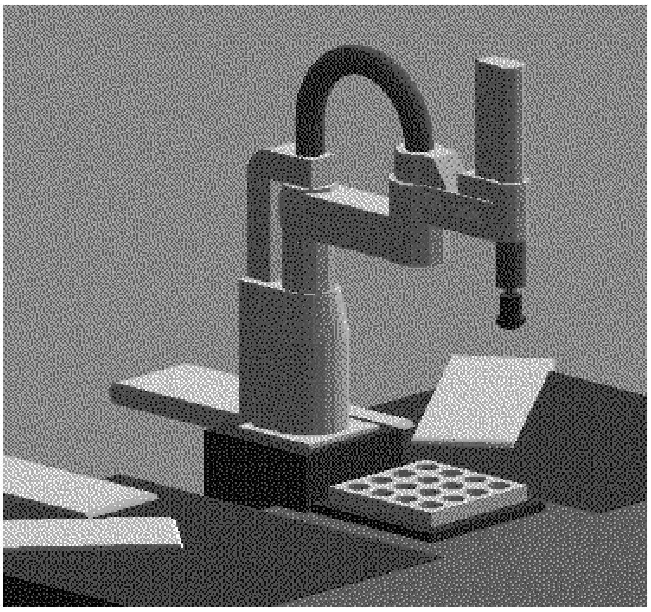

Fig. 10. Workcell simulation.

4. Conclusions

This research successfully validates the critical issues for the design of an agile manufacturing system. Flexible parts feeders, machine vision, modular hardware, a sophisticated controller interface, online error correction, graphical simulations and modular software are all essential elements of an extensive implementation. The division of tasks between workcell robots is shown to have a significant effect on assembly times, and using multiple robots in tandem to perform subassemblies is shown to be advantageous in a typical assembly task. Results show that the test assembly can be performed in approximately 20 seconds. Several thousand parts have been assembled in the workcell and assembly errors have been recorded and analyzed. Concepts for Design for Manufacture and Assembly (DFMA) are being developed as guidelines for future products to facilitate automated assembly.

In continuing work, our system is being expanded to include increased use of modular vision routines, the use of a real-time operating system and object-oriented programming, and extensive error detection and recovery. Product design for manufacturing and assembly will also play a key role in facilitating feeding, assembly, and pose estimation.

Acknowledgements

This work was supported by the Cleveland Advanced Manufacturing Program (CAMP) through the Center of Automation and Intelligent Systems Research (CAISR) and the Case School of Engineering.

References

[1] Noaker, P.M. (1994) The search for agile manufacturing. *Manufacturing Engineering*, **113**, 40–43.

[2] Comerford, R. (1993) The flexible factory: case studies. *IEEE Spectrum*, **30**, 28–29.

[3] Goldman, S. and Nagel, R. (1993) Management, technology and agility: the emergence of a new era in manufacturing. *International Journal of Technology Management*, **8**(1/2), 18–38.

[4] Burgess, T. (1993) Making the leap to agility: defining and achieving agile manufacturing through business process redesign and business network redesign. *International Journal of Operations and Production Management*, **14**(11), 23–34.

[5] Koepfer, G. C. (1995) Agile: it's about machines too. *Modern Machine Shop*, **67**, 10.

[6] Strobel, R. and Johnson, A. (1993) The flexible factory: case studies. *IEEE Spectrum*, **30**, 29–32.

[7] Bell, T.E. (1993) The flexible factory: case studies. *IEEE Spectrum*, **30**, 32–35.

[8] Boehlke, D. (1994) Smart design for flexible feeding. *Machine Design*, **66**, 132–134.

[9] Sargent, D.M. (1996) A framework for computer vision in agile manufacturing. Master's thesis, Case Western Reserve University.

[10] Japikse, D. and Olsofka, F.A. (1993) Agile engineering accelerates design. *Mechanical Engineering*, **115**, 60–62.

[11] Tracy, M.J., Murphy, J. and Denner, R. (1994) Achieving agile manufacturing in the automotive industry. *Automotive Engineering*, **102**, 19–24.

[12] Wright, P.K. (1995) Principles of open-architecture manufacturing. *Journal of Manufacturing Systems*, **14**, 182–202.

[13] May, S. (1992) Using reflective memory to build highly interactive real-time multiprocessing systems. *VMEbus Systems*, **9**(3), 13–30.

Biographies

Roger D. Quinn is an associate professor in the Mechanical and Aerospace Engineering Department at CWRU. His research specialties include agile manufacturing, dynamics and control, and biologically inspired robotics. He received his Ph.D. degree in Engineering Mechanics from VPI&SU.

Frank L. Merat is an associate professor in the Electrical Engineering and Applied Physics Department at CWRU. He specializes in computer vision, neural network signal processing, computer-aided inspection, simultaneous engineering, and manufacturing. He received his Ph.D. degree in Electrical Engineering from CWRU.

Wyatt S. Newman is an associate professor in the Electrical Engineering and Applied Physics Department at CWRU. He served as director of the Center for Automation and Intelligent Systems Research (CAISR) for five years. He specializes in robotics, agile manufacturing and rapid prototyping. He received his Ph.D. degree from MIT.

Andy Podgurski is an associate professor in the Computer Engineering and Science Department at CWRU. He specializes in program analysis, software reliability and software for agile manufacturing. He received his Ph.D. degree from UMass.

Leon Sterling joined the Computer Engineering and Science Department at CWRU in 1985. He served as director of the Center for Automation and Intelligent Systems Research (CAISR) for two years. His research includes expert systems and meta-programming for intelligent systems. He is currently Professor of Computer Science at the University of Melbourne.

Gregory C. Causey is a Ph.D. candidate in the Mechanical and Aerospace Engineering Department at CWRU. He received an M.S.M.E. degree from CWRU. His research specialties include agile manufacturing assembly and hardware, flexible parts feeding and robotics.

David M. Sargent received an M.S. degree in Electrical Engineering from CWRU. His research specialties include machine vision and generalized vision routines. He currently works in industry.

Nicholas A. Barendt is pursuing an M.S. degree in the Electrical Engineering and Applied Physics Department at CWRU. His research includes machine vision and robotics.

Virgilio (Dean) B. Velasco, is a Ph.D. candidate in the Electrical Engineering and Applied Physics Department at CWRU, from which he also obtained an M.S. degree. His research activities include robotics, controller hardware and intelligent grasping techniques.

Ju-Yeon Jo is a Ph.D. candidate in the Computer Engineering and Science Department at CWRU. Her research activities includes developing a simulation of the agile manufacturing workcell and control software.

Yoohwan Kim is a Ph.D. candidate in the Computer Engineering and Science Department at CWRU. His research includes developing software for the agile manufacturing workcell.