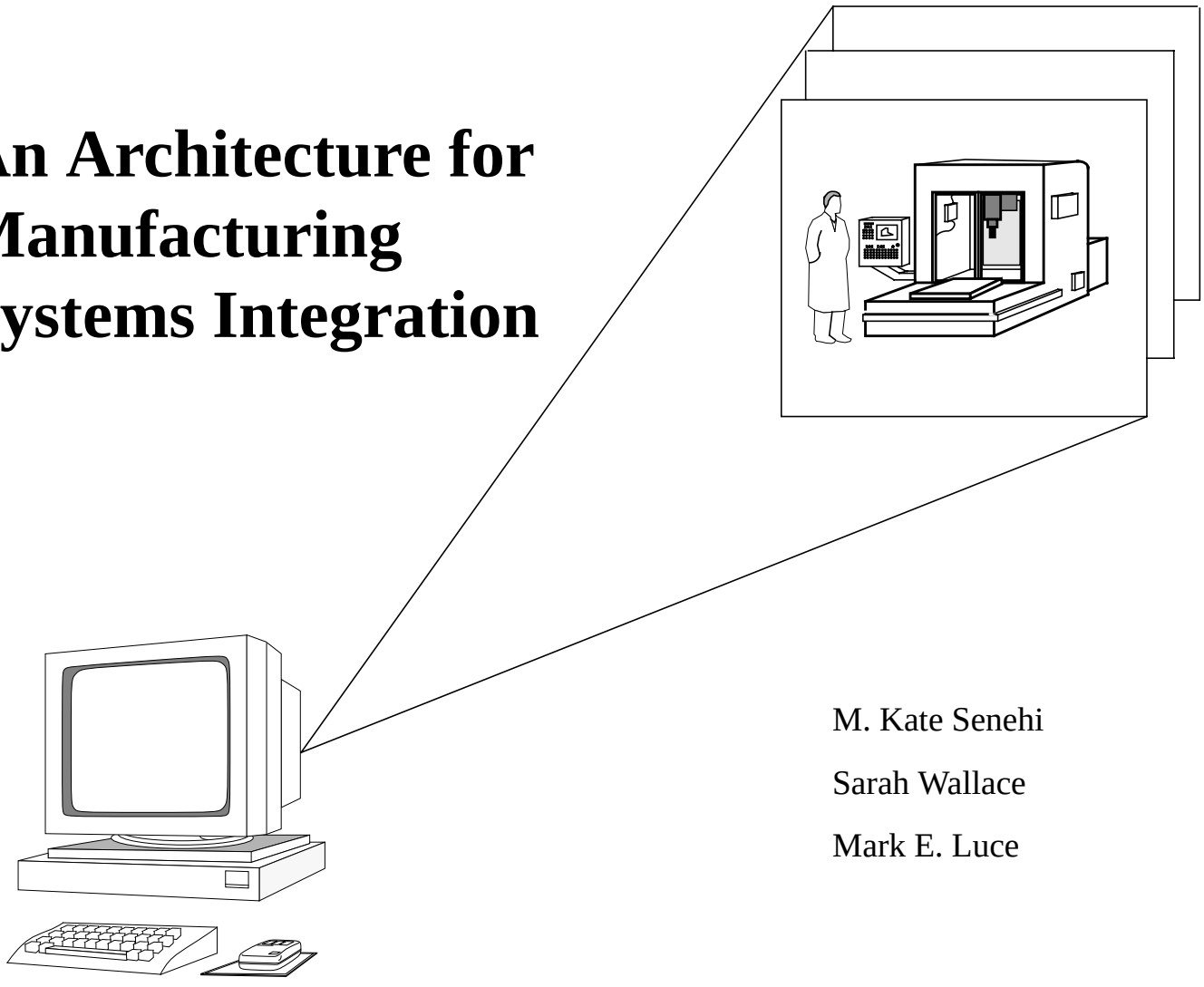
Manufacturing
Systems
Integration

An Architecture for Manufacturing Systems Integration



M. Kate Senehi

Sarah Wallace

Mark E. Luce

United States Department of Commerce
National Institute of Standards and Technology
Manufacturing Engineering Laboratory
Gaithersburg, MD 20899

Manufacturing
Systems
Integration

An Architecture For Manufacturing Systems Integration

M. Kate Senehi

Sarah Wallace

Mark E. Luce

Reprint of:

Senehi, M.K., Wallace, S., and Luce, M.E., "An Architecture for Manufacturing Systems Integration," *Proceedings of the Manufacturing International Conference*, Dallas, TX, 1992.

United States Department of Commerce

Robert A. Mosbacher,
Secretary of Commerce

National Institute of Standards and Technology

John W. Lyons, Director

An Architecture for Manufacturing Systems Integration

M. Kate Senehi, Sarah Wallace, Mark E. Luce

Manufacturing Engineering Laboratory

National Institute of Standards and Technology

Abstract

The Manufacturing Systems Integration (MSI) project at the National Institute of Standards and Technology (NIST) is developing a system architecture that incorporates an integrated production planning and control environment. The architecture is composed of an integrating infrastructure and a set of modules, one for each manufacturing function. Manufacturing functions which are specified include process planning, production planning (resource allocation and scheduling), order entry, and control. The MSI architecture identifies data which need to be either available to, or exchanged between manufacturing systems, and specifies information models for this data. These information models will be provided as input to appropriate standardization efforts.

This paper describes the MSI project and provides a summary of the MSI architecture. It explains the control and production planning paradigms used and describes provisions for error recovery. It enumerates the information models used by the modules and identifies the interfaces between those modules.

1.0 The MSI Project

Despite years of development of excellent commercial products, several major technological problems are preventing American manufacturers from easily building integrated manufacturing systems. Chief among these difficulties is the lack of standards which govern the type and form of information which manufacturing systems must exchange. Secondly, control architectures capable of coordinating the integration of the factory floor and incorporating scheduling systems have not been developed. The MSI project is developing an architecture which addresses these problems for discrete machined parts manufacturing. It is expected that the architecture will be applicable to other discrete manufacturing industries.

The MSI project, supported by NIST's funding for Scientific and Technical Research and Services and the Navy Manufacturing Technology program, is part of the Automated Manufacturing Research Facility (AMRF) effort [1]. The AMRF was established in 1981 to provide a testbed for experimental verification of new concepts for automated manufacturing and to assist American industry in improving manufacturing capabilities through the advancement and application of appropriate technology. Early research focused on the development and testing of hierarchical control architectures [2], the creation of automated workstations (e.g. [3], [4], [5], [6], [7], and [8]), and plug-compatible interfaces for manufacturing systems [9]. Later research focused on the development of formalized representations of manufacturing data [10]. These efforts form part of the technology base which supports current research in the MSI project.

The MSI project is developing an architecture for the integration of production engineering and control within a shop[1]. The emphasis of the MSI project is on the integration of manufacturing

1. A shop is a unit in a factory which performs a specific set of related functions (e.g. paint shop, machine shop, etc.). A factory is typically composed of several shops.

systems, rather than on their development. The architecture will be implemented, tested and refined iteratively, until a complete, detailed, implemented, and thereby proven architecture is developed. The implementation of the MSI architecture will be validated by the successful production of selected parts, using either actual or emulated shop floor equipment or a combination of both.

A number of results are expected from this project. First, information models defining the information which needs to be exchanged between manufacturing systems will be produced. These information models will enable manufacturing systems of the same type to be inter-operable and will aid in the identification of functional and interfacing requirements which manufacturing systems need in order to take advantage of an integrated manufacturing environment. Secondly, the formulation of the architecture will result in an improved understanding of the relationship between control and production planning systems in the error-prone factory environment. Finally, the project will produce a highly configurable testbed which is useful for testing proposed standards and will permit researchers to explore how ideas about integration apply to a large factory.

2.0 The MSI Vision of CIM

In the manufacturing community, there are many notions of what Computer Integrated Manufacturing (CIM) should be, many of which are different. The MSI architecture presumes a specific vision of CIM. To aid the reader in understanding the architecture, the MSI view of CIM is presented here.

The goal of research and development in Computer Integrated Manufacturing is to provide technologies for the creation of automated or semi-automated factories which function efficiently and cost-effectively. At present, many factories have automated systems which contribute significantly to the quality of the operations. Typically these automated systems are integrated into the factory by interacting with humans, rather than with other automated systems. While automation, whether partial or complete, offers substantial improvements for the factory by overcoming human fallibility, inefficiency in performing repetitive tasks, and cognitive bias, the full benefits of automation cannot be realized unless automated systems can interact with each other with only limited supervision by humans. To remove humans from the loop, it is necessary to automate the integration of the factory.

Factory integration is the process of making all the systems in a factory function as a cohesive unit. A first step in integration is to enable automated systems to meaningfully exchange information with each other. Unlike humans, computers require formal interfaces to accomplish this. To create a formal interface, communication links must be established, the content and format of exchanged information must be defined and the method of information exchange must be determined. For information which is persistent, a place and mode of storage must be determined and access to that information provided. Additionally, the integrity of all stored information must be maintained.

A second step in factory integration is to provide mechanisms to coordinate the activities of controllers within the factory. This is accomplished by mandating a control architecture and a formal informational interface between controllers. A control architecture specifies the relationships (or lack thereof) between controllers. A control architecture should provide a means of activating and de-activating controllers, monitoring their health, and commanding them to perform tasks.

Once formal interfaces for the systems in the factory have been established, data access and man-

agement are instituted, and a control strategy is implemented, an initial level of integration in the factory has been accomplished. This partial integration can provide substantial benefits. However, to use the full capabilities of an automated system, it is necessary to know and rely upon the *functionality* of the system. This can be accomplished by formalizing functional specifications for factory systems. A functional specification for a factory system specifies the inputs the system expects, the outputs the system produces, and what operations the system performs. A functional specification should not dictate how a system performs its operations.

A CIM architecture that specifies 1) informational and functional interfaces between systems, 2) a control structure, 3) communication services, and 4) a mechanism for providing data access and data management is sufficient to serve as a template for the integration of factories which are designed to be fully automated from inception. However, in many cases, existing factories will become increasingly automated through the inclusion of new automated systems and the replacement of humans by automated systems. Unfortunately, in this situation, there is a tendency to incorporate new systems in a manner which suffices to integrate the system at hand, but which places limitations on systems which may be integrated in the future by failing to recognize the requirements of these systems. In order to provide a smooth transition through these phases of partial automation, it is necessary to have a target architecture which provides a systematic method for the inclusion of the human element and pre-existing factory systems. The authors hypothesize that it is possible to define such an architecture by starting with an architecture for a fully automated factory which permits the incorporation of systems not originally designed to work with the architecture and adding provisions for human intervention.

A successful CIM system should provide all the functionality available in current factories. These functionalities include support for controllers, production management and error-handling. The need for a control architecture has already been discussed. Production management consists of the shop floor activities necessary to produce a part. This includes the determination of part mix, batching of orders, resource allocation and scheduling.

As long as the factory operates without errors, providing support for controllers and production management systems is straightforward. In order to perform error detection and error recovery, more sophisticated control interfaces and production management systems are required. Consider, for example, the error which occurs when a robot drops a part. In order to recover from this error, the current activity of the robot must be interrupted, and the part retrieved. The situation must then be assessed. If the part is undamaged, it can be placed at its destination and the operation can be continued. If the part is damaged, an evaluation must be made as to whether to rework the part, or scrap it and make another. In an automated system, robot programs and production plans must be made to execute either of these options. Since production management systems with the necessary capabilities do not exist, it is anticipated that error recovery will involve humans far into the foreseeable future.

3.0 The MSI Architecture

The MSI project is producing an architecture which specifies informational interfaces between systems, functional specifications of systems, a control structure, a mechanism for providing human intervention, and services for communications, data access, and data management [11],[12]. Additionally, this architecture will permit the inclusion of pre-existing systems and provide support for error-handling. As implied in the previous section, this architecture will provide

an approach for achieving automation and integration incrementally in a factory.

Presently, the MSI architecture provides for information exchange among systems in the factory. It mandates a specific control architecture and provides for informational and functional integration of control in error-free situations. The MSI architecture is currently being extended to provide functional specifications for systems. In particular, the problem of informational and functional integration for production management and support for error recovery is being addressed.

The following sections present portions of the most current version of the MSI architecture.[2] The prototype implementation of the MSI architecture, which is being developed concurrently, is not discussed at length.

3.1 Information Exchange

Information exchange is a major part of the MSI architecture. The elements necessary for automated information exchange are communication services, data access, and data management services.

The MSI architecture requires a communication service that adheres to the messaging paradigm. The significant features of the messaging paradigm are guaranteed message delivery and point-to-point communication between any two systems that need to share information directly. The Manufacturing Message Specification (MMS) [13] is an international standard which supports the messaging paradigm. The current implementation of the MSI architecture prototype is implementing the communication services using MMS.

The MSI architecture provides for the storage and access of persistent information. Consistent with the goal of providing a transition path for existing shops toward automation, the MSI architecture allows both centralized and distributed storage of data. It assumes that information to be shared between multiple systems is globally accessible and is not embedded in the systems that generate and use it. This does not preclude the use of local data management by a system to achieve satisfactory performance. However, when shared data is locally managed by a system, *that* system must maintain consistency between the globally accessible data and any local copies of the data. In all cases, the architecture requires a data access system which can combine information from multiple data repositories and does not assume that systems know the location of shared data [14]. This global data access and management capability allows pre-existing data systems (e.g. data already stored in databases, files, etc.) to be incorporated.

Because data is stored and retrieved from this logically central repository rather than being passed in messages between systems, there is one data interface for each system, rather than one for each possible pair of systems which may need to share information [2].[3]

3.2 Information Models for Information Exchange

For manufacturing information which is shared among systems, it is necessary to include all the characteristics that any of the systems using it may need. In order to achieve this, an *information model* which identifies the real world objects, their interrelationships, and useful features is required. Within the current scope of the architecture, the following information models are rec-

2. A comprehensive treatment of the previous version of the architecture is found in [11].
3. Note that the systems may well have other interfaces which are control interfaces.

ognized as being necessary in an automated shop.

The Facility Model contains both a physical and functional description of resources available in the shop. It contains templates for all such resources (e.g. machine tools, robots, etc.), information on shop floor configuration, and status information on shop systems. Note that, in an automated shop, the computer systems and software processes are also shop resources and must be included in this model.

The Process and Production Plan Models play a central role in the integration process. Process and production plans are key vehicles by which information is shared among systems in the MSI architecture. A process plan designates the steps necessary to make a part, specifying the sequence(s) of operations by which a part is made and the relative timing of these operations. A process plan may express both alternative and parallel paths of part production. This definition differs from the traditional use of the term "process plan." A production plan is constructed from a process plan (see section 3.5.2 on page 8) and specifies the resource allocation and scheduling information to make a specific number of parts. Since a production plan is constructed with reference to a process plan, it is obvious that their representations are logically, if not physically, linked. Process and production plans are used by controllers to manufacture parts.

Since there is currently no standard for representing process and production plans, the current implementation of the MSI architecture prototype uses an internally developed process and production plan specification. Detailed discussions of this specification can be found in [15] and [16].

The following table summarizes additional models for information which must be specified in an automated shop.

Product Model	Specifies information needed to describe the parts being manufactured: includes information needed to create a solid model of the part, to describe manufacturing features, and to specify detailed tolerancing information.
Orders Model	Describes information about orders: includes the type of the part to be manufactured, the quantity to be made and identifies information needed to record the engineering status and production status of the order.
Inventory Model	Specifies information needed about stock (e.g. part blanks), consumable machining supplies, free carriers and completed parts no longer in-process. Such information includes type, quantity, location, etc.
Tooling Model	Describes tool characteristics (e.g. type, material, etc.) and dynamic information (e.g. tool life, location) about tools in the shop.
Work In-Process Model	Describes the state and location of workpieces (blanks, partially completed pieces and finished parts) **assigned** to **active** jobs, lots, carriers, etc.
Materials Model	Describes the characteristics of raw materials, stock, and consumable machining supplies.

Table 1. Information Models in the MSI Architecture

3.3 Systems in the MSI Architecture

As previously stated, the completion of the architecture requires functional specifications for systems which are to be integrated using the architecture. Systems recognized as necessary for the operation of a shop by the MSI architecture are:

- Part Design Systems — which create the designs for parts, associated fixtures and jigs,
- Process Planning Systems — which create step-by-step plans for manufacturing the part, associated fixtures and jigs according to the design,
- Production Planning Systems — which select batch sizes, specific machines, and scheduled times to perform the tasks specified by the process plan,
- Controllers — which perform manufacturing tasks,
- Order Entry Systems — which permit entry of orders which direct the shop as to what to make and when to make it,
- Configuration Management Systems — which identify and control shop resources and capabilities, and
- Material Handling Systems — which route and deliver material throughout the shop.

Currently, the MSI architecture provides a complete functional specification for controllers, and partial functional specifications for process and production planning systems.

3.4 Inclusion of Systems into an Integrated Shop

One of the goals of the MSI architecture is to provide a smooth transition from a presently existing non-automated or partially automated shop to a totally automated shop. Given an existing shop with both automated and manual systems, it is possible to move towards integration by first providing services for information exchange and creating data repositories structured according to the information models, and then integrating existing automated systems. As additional automated systems are brought into the shop to replace humans, these may also be integrated. Using this technique, integration of the shop can proceed incrementally, permitting the use of previous integration efforts and facilitating the coordination of systems.

Automated systems which are not compatible with the MSI architecture are called *foreign* systems. The MSI architecture provides a method for determining the compatibility of foreign systems with the MSI architecture and a methodology for integrating them into an environment which is informationally integrated.

The method for determining a system's compatibility with the MSI architecture is to compare the functional specification for the same type system provided by the architecture to that of the candidate system. The following procedures are used to integrate the foreign system.

In order to incorporate a foreign system that has the complete functionality of one or more architecturally specified systems, it must be described in the MSI data repositories as specified by the information models, and interfaced to fit into the data and communication environment. Such interfacing consists of converting commands and information into forms expected by the system and converting outputs of the system into forms expected by other MSI systems. If a system does not exhibit the functionality required by the formal functional specification of the architecture for that system, the usefulness of the system will be limited. If the system has additional functionality

which requires externally produced information not specified by the architecture, its operation must be circumscribed, reducing its functionality to a subset consistent with the architecture. If the system does not exhibit the complete functionality of *any* architecturally specified system, it is necessary to provide the missing functionality. If this is not possible, the system cannot be fully integrated into the shop. However, it may still be possible to perform some degree of informational integration and the system may be usable. In this way, the architecture allows for the inclusion of systems whose internal structure is unknown, but whose external interfaces conform to the architectural specification.

3.5 Control Architecture

The MSI project addresses production management issues within a single shop. It is assumed that a shop receives externally generated orders and is self-contained with respect to manufacturing production concerns. It is recognized that the shop receives other external inputs such as part blanks, supplies, and other raw materials. Since acquiring these is independent of the manufacturing process, the MSI architecture assumes they are available in unlimited supply. Thus the MSI project limits its scope to manufacturing production concerns and does not address other concerns such as purchasing, marketing, etc.

The MSI project approach to manufacturing production is top-down from the higher level goal of maximizing productivity rather than bottom-up from observation of performance on the shop floor. Because a top-down approach is being used, all systems in a shop can be coordinated in a manner which uses each system in a way that advances the *goal of the shop*, rather than that of any individual system.[4]

The goal of a shop is to fill orders in a cost-effective manner; each order corresponds to a high level task for the shop, which requests the manufacturing of a specific number of a certain product. From this task, tasks for machine tools and material handling systems must be generated. This can be accomplished by iteratively decomposing each task into smaller subtasks until the subtasks are appropriate for shop equipment to perform. This technique is known as hierarchical task decomposition [1]. Subtasks of a given task may be either dependent or independent of each other. Clearly, decomposition is most effective in reducing the complexity of the task when its subtasks are independent. In this case, coordination of the execution of the subtasks is not necessary. In the case where subtasks are dependent on one another, the decomposition is still useful, but the subtasks must be coordinated.

3.6 Hierarchical Control

The decomposition of a high level task for filling an order generates a hierarchical task decomposition in which equipment tasks are frequently dependent. Consequently these equipment tasks must be coordinated. For example, the robot's delivery of a part to the machine tool must be coordinated with the state of the machine tool's vise.[5] The MSI architecture achieves this coordination by the imposition of a hierarchical control structure. A hierarchical control structure has a single highest-level controller. Every other controller has exactly one supervisory controller from which it receives commands, and zero or more subordinate controllers to which it may issue commands.

4. Note that this differs from the traditional approach to increase shop productivity, which is to maximize machine utilization on the shop floor.

5. In general, material handling must be coordinated with machining.

In the MSI architecture, this highest level controller is the *shop controller.* The shop controller has general responsibility for all production processes involved in filling orders. The coordination of all orders for parts, the determination of global scheduling constraints and the creation of routing slips for part delivery are done by the shop controller. However, many of the details required to fill these orders are the responsibility of subordinate controllers and are not visible to the shop controller.

Equipment is controlled by an *equipment controller.* By definition, an MSI equipment controller can execute only one task from its supervisor at a time.[6] Equipment controller tasks are items such as loading a part, opening a vise, or manipulating the spindle of a machine tool, depending on the particular equipment.

Between the shop and equipment controllers there may be any number of controllers, called *workcell controllers,* which coordinate the activities of two or more subordinate controllers, each of which is either an equipment or a workcell controller. The number of controllers between a given equipment controller and the shop controller remains unchanged regardless of the tasks being executed. This number specifies the level of control for that controller, and may be different for different equipment controllers depending on the complexity of coordination necessary. See Figure 1. for an example.

In general it is expected that, once established, the control hierarchy will remain fixed as long as the shop is in operation. However, it is possible to dynamically reconfigure the control hierarchy. It is intended that dynamic reconfiguration will only be used to remove a dysfunctional equipment controller, to bring new equipment on line, or to adapt the configuration to accommodate changes in manufactured products.

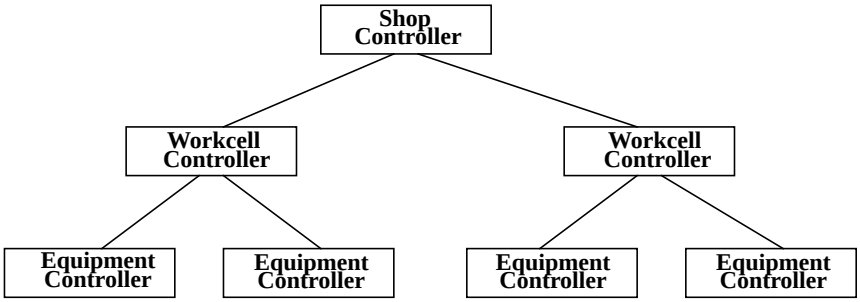
In the MSI architecture, at any fixed time, there is a *single* control hierarchy originating at the shop controller. This control hierarchy specifies communication paths between controllers and the type of information which may be communicated between controllers. A controller may also access data repositories to obtain additional data; thus data flow does not mirror control flow.

3.6.1 Controllers in a Hierarchical Structure

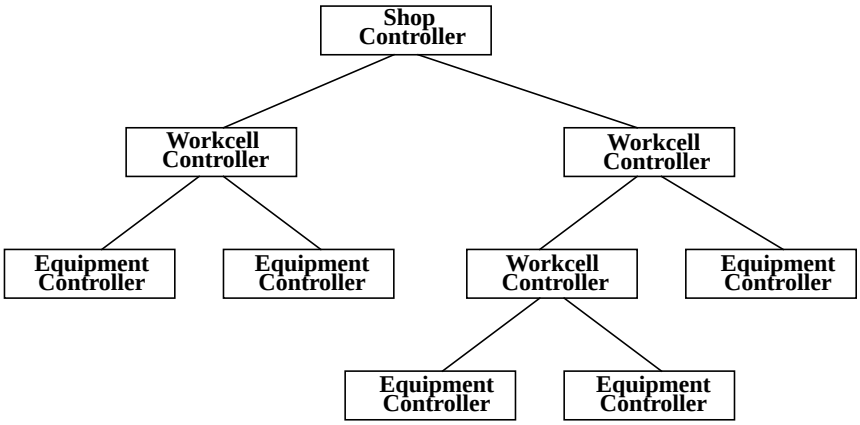
Once a control hierarchy has been chosen, the impact of this decision upon shop operations must be examined. Before discussing this impact, we must first describe how an automated shop functions with a hierarchical control structure. Shop operations will be described by identifying the major functions and their required inputs and outputs:

- Process Planning. When orders enter the shop, process plans must be generated. Due to the fact that a hierarchical control structure is required by the MSI architecture, process plans must also be hierarchical. There must be process plans for every level in the hierarchy except the equipment level, where machine-specific programs (e.g. numerical code, robot programs, etc.) may be used.
- Production Planning. By accommodating batching, part mix, and material handling constraints, production-managed plans (in MSI terminology) are produced from process plans. The production planner must schedule these hierarchical production-managed plans, allocating appropriate resources, and produce hierarchical production plans.

6. Internal task decomposition is possible, but not addressed by the MSI project.



(a) This is an example of a control hierarchy, in which the number of levels between each equipment controller and the shop controller is the same.



(b) This is an example of a control hierarchy, in which the number of levels between equipment controllers and the shop controller differs.

Figure 1. Examples of Valid Control Hierarchies

There must be production plans for every level of control, except possibly for the equipment level. For equipment controllers which use machine-specific programs, production plans are not generated; the higher level production plans specify the scheduled execution times of these programs.

- Administrative Control. Prior to executing tasks the control hierarchy must be established: controllers must be activated and must identify their supervisor and subordinates. Additionally, the health of controllers must be monitored and a mechanism to deactivate controllers must be provided. In the MSI architecture, administrative control information is communicated through the control hierarchy.
- Task Control. After the control hierarchy has been established, the shop is ready to accept manufacturing tasks. Tasks must be routed to controllers for execution through communication paths provided by the control hierarchy. A controller routes tasks to subordinate controllers and monitors the subordinate's execution of the tasks.

To function in a hierarchical control environment, the production planning system must support

invocation by an automated system (i.e., its invocation must not require human interaction) and be able to produce production plans for all levels in the control hierarchy which require production plans.[7] In an error-free shop, these functions are sufficient to integrate a production planing system. When errors occur, the relationship between production planning and control is much more complex. Even in shops where automated scheduling is in place, errors are handled by humans. The shop schedule must be updated to reflect this human performed recovery, or it will become increasingly obsolete. This situation could continue until a new schedule is in place. In a completely automated shop, where the schedule *drives* the shop, it is clearly unacceptable to have an obsolete schedule.

3.6.2 Error Handling

A driving force behind many decisions made in the MSI architecture is the motivation to provide adequate mechanisms for error recovery. The ability to recover from an error is intimately related to the capabilities of the production and process planning systems. Since production and process planning systems are not likely to be capable of fully supporting error recovery in the foreseeable future, a mechanism for human intervention is included in the MSI architecture.

Human intervention for controllers is provided through the *Guardian Interface* which allows an operator to observe the internals of a controller and permits the operator to remove a controller from the control hierarchy, debug and test through direct modification of system internal state data, diagnose errors through examining internal state data, and affect the outcome of executing tasks. The Guardian Interface is the mechanism through which dynamic reconfiguration of the control hierarchy can take place: a controller can be inserted into or removed from the control hierarchy based upon the judgement of the Guardian operator.

The Guardian Interface consists of two components: the communications interface which transports raw data from the controller and a translator which converts raw data into a human readable representation. The separation of the translator and communications interface decouples the operator from the controller, allowing automated systems to replace the Guardian operator as such systems are developed.

Errors can be grouped into three different classes based upon their cause: resource error, task error, and tooling error. A resource error occurs when a piece of equipment, whose controller is part of the control hierarchy, becomes impaired (e.g. the machine tool bed jams). A task error is an error which affects a specific task only, the resource on which it is being performed is unaffected (e.g. the robot drops the workpiece). Note that while a resource error usually causes a task error, task errors may occur without a resource error.A tooling error occurs when a tool is damaged (e.g. a cutter breaks) or unavailable (e.g. the tool was not delivered at the proper time). Tools differ from other resources in that they are not permanently associated with any member of the control hierarchy, but are moved from resource to resource as needed. Recovery from a tooling error is not currently addressed by the MSI architecture.

For any error occurring in the shop, there are two types of error recovery which may be required: resource recovery and task recovery. Resource recovery is the process of determining the condition of the affected equipment, possibly repairing or replacing it, and ensuring that this equipment

7. Note that an arbitrary off the shelf scheduling system need not provide these functions, but rather it must be possible to encapsulate the system in such a way as to provide these functions.

is not assigned tasks while it is disabled. Task recovery is the process of making it possible for the task to complete, or replacing it with a new task which accomplishes the same goal. One or both of these recovery mechanisms may be used depending upon the specific error.

Recovering from a resource error requires resource recovery and may require task recovery. The Guardian operator is responsible for diagnosing the equipment failure which caused the resource error and initiating the resource recovery. The Guardian operator is responsible for assessing the state of the equipment and determining whether it needs to be repaired or replaced. In addition, the Guardian operator may choose to remove the controller from the current shop configuration. If the Guardian operator determines that the equipment needs to be repaired or replaced, the production planner needs to be notified of the equipment's unavailability. For the period for which the equipment is unavailable, it is the production planner's responsibility not to allocate that resource to perform tasks and to reschedule tasks which have previously been assigned to that resource. Resource errors frequently cause the currently executing tasks to be delayed or unable to complete. In this case, task recovery is necessary.

For task error recovery, it may be necessary to perform one or all of the following operations: dynamic scheduling, dynamic resource allocation, and dynamic processing planning. The term *dynamic* is used here to indicate that the function is being performed in response to the situation on the shop floor. An example illustrates this use of the term. Suppose that an error occurs in making a workpiece and the workpiece is damaged. Although a process plan was available for the production of the workpiece from a blank, this plan may not be applicable to the damaged workpiece. If the Guardian operator decides to recover this workpiece, rather than scrap it, a new process plan must be created. The need for this plan was not anticipated, because the workpiece is not normally in a damaged state, hence the plan is said to be dynamically generated. Once the new process plan has been created, dynamic resource allocation and scheduling are required in order to use these process plans.

A primary goal in error handling is to minimize the number and type of adjustments necessary to fix the error. This is not always as easy as it would first appear: error recovery can be surprisingly complex. Consider, for example, the situation which occurs when a task takes longer than planned. Intuitively, since neither an equipment, task or tooling failure, nor damage to the workpiece has occurred, one would expect recovery from this error to be straightforward. However, under certain conditions, recovery from this error may require dynamic process planning for material handling steps. To illustrate how this may happen, the possible recovery sequences will be discussed in detail.

Before proceeding with the discussion, some terminology is necessary. In hierarchical control systems, it is necessary to distinguish between *tasks* and *subtasks*. A task is the action which is requested of a controller. The controller decomposes this high level task into an ordered sequence of subtasks. A controller generates its own subtasks, when these are dispatched to the controller's subordinates, they are the subordinate's tasks.

When a subtask takes longer to complete than scheduled, the subtask's late completion may or may not affect the starting of a subsequent task or subtask. In the latter case, no action is necessary. In the former case, the impact of the initial subtask's tardiness upon subsequent tasks and subtasks must be assessed. When there is a subsequent subtask, this subtask must be rescheduled so that enough time is allocated for the initial subtask to complete. If the resource(s) allocated to the next subtask are available during the new scheduled time, no further adjustments are neces-

sary. If, however, this change in scheduled time cannot be accommodated by its assigned resource, dynamic resource allocation must occur (i.e., that subtask must be re-assigned to another resource). When dynamic resource allocation is necessary, the physical locations of affected workpieces must be considered. If a workpiece does not have to be moved in order for the newly assigned resource to work on it, the schedule has been fixed. If however, a workpiece must be moved, then dynamic process planning, resource allocation and scheduling must be done to generate and schedule material handling plans for moving the workpiece. The impact of recovery of this subtask upon the subsequent subtask is then assessed. This process of fixing subsequent subtasks continues until a subtask is found which is unaffected or until the last subtask of the task. (Note that a subtask may have a *task* rather than a subtask as its immediate successor exactly when the initial subtask is the last subtask in the task.) If the entire task can be performed within its original scheduled time or the task can be rescheduled without affecting the task which comes after it, the recovery is localized. If the rescheduling of the task affects the next task in the schedule, then the error is reported to the supervisory controller, which notifies the production planner to perform the rescheduling and evaluation procedure to handle the error. This process of percolating error recovery upwards continues until the problem is resolved.

This example illustrates how the use of a hierarchical control system facilitates the localization of task error-handling. When an error occurs in the execution of a task, if it is possible to resolve it by affecting only subtasks of the task, controllers at all levels of control above the supervisory controller are unaffected by the error. If localized error recovery is not achieved at this level of control, the recovery for the error is handled by the next higher control level in the hierarchy. At each level, there is potential for error resolution. Only in the event that the error cannot be resolved at any lower level is a global solution to the error required.

Analysis of the error scenario reveals two capabilities which a production planner must have in order to be effectively used. The first and more general requirement is that the production planner must be able to do re-planning. Re-planning is the ability to localize the error to a subset of the tasks and only re-plan those which are affected. If re-planning capability does not exist, automated error recovery will be extremely limited. The production planner must schedule for the entire shop again. The availability of resources can be fed back into the scheduler, but the scheduler can not plan for the completion of partially executed production plans. Human intervention is required to avoid scrapping everything in execution when an error occurs.

The second requirement is the need for the production planner to work with the hierarchical control system. In order to localize an error at a given level of the hierarchy, the production planner must be able to plan for that level. Additionally, it must be possible for the production planner to be informed that a resource or task error has occurred at a given level and re-planning may be necessary. Information associated with the error which is needed to re-plan must be made available to the production planner (e.g., how many minutes late a machining task is expected to be). The architecture does not specify whether the production planner has an interface to each controller or whether error-related information transfer is accomplished through a database interface. As a minimum, however, the production planner must be able to be notified by the shop controller that a resource or task error has occurred and re-planning may be necessary. It is the controller's responsibility to notice the error and inform the production planner; the production planner does not monitor either the health of the controller, or the execution of tasks. Beyond these interface requirements, the internal architecture of the production planner is not specified.

4.0 Conclusion

When complete, the MSI architecture will assist shops in progressing toward total automation by providing a target architecture for complete integration and a methodology for incremental integration. A totally automated and integrated shop may be possible after dynamic process and production planning systems are commercially available. These topics are currently the subject of active research, some of which are on the verge of producing commercial systems. However, with today's technology it is possible to integrate automated workstations into a coordinating control structure and provide some degree of informational integration. Moreover, further integration can be performed incrementally. As a shop progresses toward total automation and integration, shop operations will greatly benefit from the partial automation and integration that becomes available.

5.0 Acknowledgments

The authors wish to acknowledge the contributions of the MSI architecture committee. This committee developed some of the concepts presented in this paper. In 1991, the architecture committee members were Ed Barkmeyer, Mark Luce, Steven Ray, M. K. Senehi, Evan Wallace and Sarah Wallace.

References

[1] Simpson, J., R. Hocken, J. Albus; "The Automated Manufacturing Research Facility," *Journal of Manufacturing Systems*, Volume 1, Number 1, 1982.

[2] Albus, J., A. Barbera, R. Nagel; "Theory and Practice of Hierarchical Control," *Proceedings of the 23rd IEEE Computer Society International Conference*, September, 1981.

[3] Lee, K. B., A. Donmez, C. Yang; "An Automated Turning Workstation in the AMRF," *Proceedings of the Third Biennial International Machine Tool Technical Conference*, Chicago, IL, September, 1986.

[4] McCain, H. G., R. D. Kilmer and K. N. Murphy; "Development of a Cleaning and Deburring Workstation for the AMRF," *Proceedings of the Deburring and Surface Conditioning '85 Conference*, July 1985.

[5] McLean, C. R. and C. E. Wenger; "The AMRF Material Handling System Architecture," *Proceedings of Control 86 Conference and Exposition*, Barrington, IL, May, 1986.

[6] McLean, C. R.; "The Vertical Machining Workstation of the AMRF: Software Integration," *Proceedings of the ASME Symposium on Integrated and Intelligent Manufacturing*, Anaheim, CA, December, 1986.

[7] Moncarz, H. T.; "Architecture and Principles of the Inspection Workstation," National Bureau of Standards Interagency Report 88-3802, June, 1988. (Available from the National Technical Information Service, Springfield, VA 22161.)

[8] Scott, H. and K. Strouse; "Workstation Control in a Computer Integrated Manufacturing System," *Proceedings of Autofact* VI, Anaheim, CA, October 1984.

[9] McLean, C. R.; "Interface Concepts for Plug-Compatible Production Management Systems," *Proceedings of the IFIP WG5.7 Working Conference on Information Flow in Automated Manufacturing Systems*, Gaithersburg, MD, August 1987. Reprinted in *Computers in Industry*, Volume 9, pp. 307-318, 1987.

[10] Hopp, T; "Proceedings of the Manufacturing Data Preparation (MDP) Workshop," unpublished report, June 1988. (Available from the Factory Automation Systems Division, National Institute of Standards and Technology, Building 220, Room A127, Gaithersburg, MD 20899.)

[11] Senehi, M.K., et al.; "Manufacturing Systems Integration Initial Architecture Document," National Institute of Standards and Technology Interagency Report 91-4682, September, 1991. (Available from the National Technical Information Service, Springfield, VA 22161.)

[12] Senehi, M.K., et al.; "Manufacturing Systems Integration Control Entity Interface Document," National Institute of Standards and Technology Interagency Report 91-4626, June 1991. (Available from the National Technical Information Service, Springfield, VA 22161.)

[13] ISO 9506, "Industrial Automation Systems Manufacturing Message Specification, Part 1: Service Definition." (Available from the International Organization for Standardization, Geneva, Switzerland.)

[14] Barkmeyer, E., et al.; “An Architecture for Distributed Data Management in Computer Integrated Manufacturing,” National Bureau of Standards Interagency Report 86-3312, January 1986. (Available from the National Technical Information Service, Springfield, VA 22161.)

[15] Catron, B. and S. Ray; “ALPS – A Language for Process Specification,” *International Journal of Computer Integrated Manufacturing*, Volume 4, Number 2, 1991, pp.105-113.

[16] Ray, S.; “Using the ALPS Process Plan Model,” *Proceedings of the Manufacturing International Conference*, 1992, Dallas, Texas.