Generic Manufacturing Controllers

Bryan A. Catron
Bruce H. Thomas

Factory Automation Systems Division
Center for Manufacturing Engineering
National Institute of Standards and Technology
Gaithersburg, MD 20899

Presented By:

Bryan A. Catron
IEEE Conference on Intelligent Control
Arlington, Virginia
August 24-26, 1988

Bibliographic Reference:

Catron, B. A., Thomas, B. H., "Generic Manufacturing Controllers", Proceedings of the IEEE Conference on Intelligent Control, Arlington, Virginia, August 1988.

Generic Manufacturing Controllers

Bryan A. Catron
Bruce H. Thomas
Factory Automation Systems Division
National Institute of Standards and Technology
Gaithersburg, MD 20899

Introduction

The cost of developing software for the factory is continuing to increase due in part to the problems of integrating equipment and controllers from multiple vendors. A consistent philosophy is needed for designing reusable code for integrating manufacturing control systems. Development of a "generic controller" will minimize the problem of redundant software. [4]

For a better understanding of this particular approach to a generic controller, an overview of the Automated Manufacturing Research Facility (AMRF) facility, a definition of a generic controller, and a description of results from development and implementation of a version of the generic controller are presented.

AMRF Overview

A major goal of the AMRF project has been the establishment of a test bed small-batch manufacturing system at the National Institute of Standards and Technology (NIST) Gaithersburg, Maryland site. The test bed, which became operational in 1983, is designed to be used by government, industry, and academic researchers for the development, testing, and evaluation of potential interface standards for manufacturing systems. To ensure that as many critical system integration issues as possible could be addressed, component modules were chosen from many different vendors. Therefore, the NIST test bed differs from virtually all other flexible manufacturing systems in the variety of "off-the-shelf" components that have been integrated into a single coordinated operation.

The industrial machinery of the AMRF occupies a 5000 square-foot area of the NIST machine shop. The factory systems that are located on the floor of the AMRF include: two machining centers, a turning center, a coordinate measuring machine, six robot manipulators, a vision system, two wire-guided vehicles, storage and retrieval systems, tray roller tables, tool setting stations, vacuuming and other cleaning equipment, part fixturing and robot gripper systems. Integration test runs of the AMRF have demonstrated the successful automated production of small batches of machined parts. This paper outlines one particular implementation of a controller composed of generic parts, and is being developed to support the Cell, Vertical Workstation, Inspection Workstation, and the Material handling Workstation of the AMRF.

Generic Controller

The key to defining our generic controller is the decomposition of controller activities into sub-activities called *managers*. A well-defined manager will encapsulate a sub-activity and provide a clean interface. This decomposition is analogous to information hiding techniques used to decompose a program into subroutines [6]. Decomposition of tasks provides for localized decision making and allows for incremental enhancements without affecting other code (provided the interface is preserved).

A controller built using generic controller principles will have two major components, a generic component and an application component. AMRF researchers believe that a major segment of the functionality of a manufacturing controller is identical to any other controller on any level of a factory. This identical part of the controller software is termed the generic component. The remaining part of the software is functionally specific to that controller and is termed the application component.

AMRF research efforts have focused on the identification of generic control principles which could serve as a basis for the development of future interface standards for factory control systems. By using generic functional requirements for control systems, AMRF researchers believe that redundant software development efforts can be minimized. For example, the Cell, Vertical Workstation, Inspection Workstation, and Material Handling Workstation controllers within the AMRF must support the following set of basic functions:

1) the processing of control messages expressed in the AMRF command/status message format,
2) the exchange of data with the AMRF through the Integrated Manufacturing Data Administration System (IMDAS),
3) the retrieval, interpretation and execution of process plans that have been prepared in a standard AMRF process plan format,
4) the transition of initialization and shutdown states according to a protocol developed for the AMRF by the University of Virginia,
5) the transmission of messages through AMRF common memory areas and communications networks, and
6) the input and display of controller data through a human interface system.

Goals

The architecture of the generic controller under development at the AMRF revolves around four fundamental goals: 1) functionally decompose activities into generic modules, 2) allow (but do not require) multiprocessing and distributed processing, 3) build libraries of generic utility routines, and 4) use configuration data files for initial setups. A further description of these goals and the rationale behind them is given below.

Decomposition of activities should reduce interdependence between managers allowing each manager to operate individually for testing and simulation. With a strictly defined interface, each manager can be developed and implemented independently. Managers with well defined functionality facilitate reuse of source code. These reusable managers form the core of the generic controller.

Secondly, the generic controller must be able to operate in a distributed multiprocessing environment. With the advent of inexpensive computers and networks, a single processor environment is no longer a constraint. Multiprocessing is not required, however, and all modules may operate as a single process depending on implementation. Conceptually though, the managers are independent processes capable of concurrent execution across a network. To accommodate the distributed multiprocessing environment, communication between managers must be handled in a manner which hides implementation concerns: Separate processes communicating transparently over a network will function identically to multiple processes on a single computer.

The third goal of a generic controller is the reduction of duplicate software. To address this issue, libraries of routines will be developed whenever two or more managers share a common need. The communication routines are the best example because every manager must communicate with other managers. Other candidates for function libraries are message parsers, protocol translators, string manipulation, and debugging routines.

The final goal is to develop data configurable managers. The managers should be self-contained entities with a limited outside view of the world. Configuration with the outside world and with other functions will be data driven to reduce the amount of custom code required.

The intent of the architecture is to allow greater flexibility and reusability by providing a set of managers from which an application designer may select. A controller is built by selecting the required functionality and combining those managers with application-specific code. The internal algorithms of managers are independent of other managers and modifications to one manager will not affect others.

Development and Implementation

Development of the described generic controller began at the AMRF in late 1987 as a research effort to design a controller for manufacturing data preparation [9]. The Manufacturing Engineering Control System (MECS) provided the test vehicle for the generic controller. Other AMRF manufacturing control applications were analyzed during the design phase including a cell controller, vertical workstation controller, inspection workstation controller, and a material handling controller. The initial decomposition of activities into managers was based on previous controller architectures [3] and extensions were made to improve flexibility and reusability. The generic controller was designed to support the current AMRF functionality. However, every effort has been made to expand the applicability of the architecture outside the AMRF.

An initial set of managers was identified which represent some major generic functions of a factory controller. Additional generic functions will be identified as development and implementation continue. Figure 1 shows the key managers which have been identified. The defined managers provide functionality to control and synchronize tasks, schedule tasks, facilitate a user interface, communicate with other controllers, and interface with a database.

In addition, each manager provides a set of variables which can be monitored by outside processes. Data files are used to configure each manager with the rest of the controller.

The Transition Manager provides initialization, start-up, and reconfiguration protocols between controllers. A state-transition model is used to provide deadlock-free synchronization between controllers through various stages of readiness [1] [5]. Managers within a controller use the Transition Manager as a central synchronization point to facilitate internal synchronization during start-up and shutdown sequences. Since all controllers require a Transition Manager, generic code has been developed to accommodate these needs.

The User Interface Manager is responsible for all manual input to the controller. In addition, most screen output will be processed by the User Interface Manager to provide a uniform look and feel to the controller. The proposed User Interface Manager is built with a user interface editing tool on top of X windows [7][8]. The combination of an editing tool and a portable windowing environment provide greater flexibility and portability.

Managers communicate with one another through an internal AMRF standard set of communication utility routines providing access to the communications scheme. The Inter-Process Communications Manager (IPC) is responsible for the implementation of the communication primitives. The IPC is based on a common memory scheme of communication which has been used in the AMRF. The exact communication paradigm implemented should be transparent to a calling routine. This stresses flexibility and reusability by providing a uniform interface to communications. In addition to the IPC process, there is a library of routines which define the external interface to the IPC manager. A communication configuration data file is used by all managers to initialize the communications.

The Database Interface Manager provides access to application data files. These may reside either in a local database or in a remote database. The actual location of the files and the exchange protocol of the database system are transparent to other managers. The

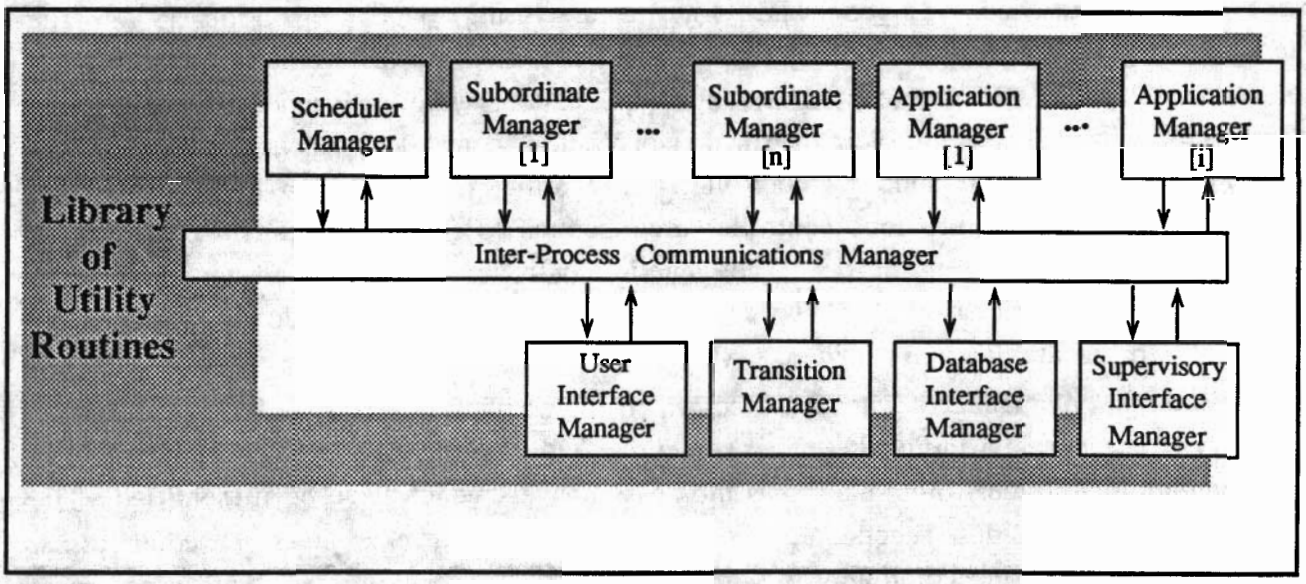

Figure 1

Database Interface Manager provides a shell around the AMRF Integrated Manufacturing Data Administration System (IMDAS) distributed database [2].

In a hierarchical control system, a controller has a single supervisory controller and one or many subordinate controllers. The Supervisor Interface Manager handles protocol conversion from the supervisory controller to the internal managers. Supervisory commands may originate from a remote supervisory controller or a local operator. The local operator is always given priority over the remote supervisor to allow local intervention or error recovery. Commands are translated and sent to the appropriate manager for further processing.

A Subordinate Interface Manager handles interfacing with a subordinate controller or device. Each subordinate or device has a unique Subordinate Interface Manager to handle communication and protocol translations. At the lowest level in the control hierarchy, the subordinates are actually equipment and the Subordinate Interface Manager is an application-specific device driver. Subordinate Interface Managers at higher levels communicate with the Supervisor Interface Manager of lower level controllers. However, the internals of the Subordinate Interface Manager are hidden to other managers behind a uniform interface. Subordinate managers are also responsible for error recovery of the subordinate controller. Emulation of the subordinate allows system integration testing prior to installing the hardware.

The Scheduler Manager performs the scheduling of jobs at a local level. Scheduling arranges the current jobs in priority order before they are executed. A generic interface to scheduling processes allows the controller to choose between available schedulers.

The Application Manager encapsulates all non-generic portions of code and provides overall sequencing and job control for a specific application. This manager utilizes and coordinates the other managers to perform work. The Application Manager is intended to encapsulate all of the non-generic source code and may be divided into multiple processes. The current generic controller is still under development and has four managers designed and implemented: 1) Transition Manager, 2) Supervisor Interface Manager, 3) Database Manager, and 4) Inter-Process Communication Manager. The Database Manager is implemented in LISP while the others are implemented in the C programming language. Different programming languages were intentionally used to further illustrate the reusability of the managers and their generic interfaces.

Future Work

The Application Manager will be implemented in LISP and will provide the fundamental control for the MECS controller. Additional generic managers may be extracted from the Application Manager as the generic controller is more fully developed, further reducing the amount of application specific code required. Although application specific code will continue to be developed, the generic controller can significantly reduce the expense of developing new factory controllers.

The implementation of this generic controller is intended for the AMRF environment. However, the basic ideas are not specific to the AMRF and could be implemented elsewhere.

References

[1] Catron, B.A. "Implementing a Transition Manager in the AMRF Cell Controller," *Proc. of 3rd International Conf. on CAD/CAM Robotics and Factories of the Future*, Southfield, Michigan, Aug. 1988

[2] Furlani, C. et al. "The Integrated Manufacturing Data Administration System (IMDAS)," to be published as an NIST Internal Report, 1988

[3] McLean, C.R. "A Cell Control Architecture For Flexible Manufacturing," *Proc. of the 1987 Advanced Manufacturing Systems Conf.* , Chicago Illinois, June 1987

[4] Naylor, A.W. and Volz, R.A., "Design of Integrated Manufacturing System Control Software", *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 17, No. 6, pp.881-897, 1987

[5] O'Halloran, D.R., and Reynolds, P.F., "A Model for AMRF Initialization, Restart, Reconfiguration, and Shutdown," NBS/GCR 88-546, May 23, 1986

[6] Parnas, D.L. "On the Criteria To Be Used in Decomposing Systems into Modules," *Communications of the ACM*, vol 15, pp 1053-1058, Dec. 1972

[7] Scezur, M., Stephens, M., Perkins, D., and Moe, K. "TAE Plus: Evolution of a NASA User Interface Management System", *Presented at 26th Annual Technical Symposium of the DC Chapter of the ACM*, 1987

[8] Scheifler,R.W. and Gettys, J., "The X Window System", *ACM Transactions on Graphics*, Vol. 5, No. 2, April 1986

[9] Thomas, B.H. and McLean, C.R., "Using Grafcet to Design Generic Controllers", *1988 International Conference on Computer Integrated Manufacturing,* Rensselaer Polytechnic Institute, Troy, NY, May 1988

The NIST Automated Manufacturing Research Facility is partially supported by the Navy Manufacturing Technology Program.

Certain commercial equipment, instruments, or materials are identified in this paper in order to adequately specify the experimental procedure. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

This is to certify that the article written above was prepared by United States Government employees as part of their official duties and is, therefore, a work of the U. S. Government and not subject to copyright.