16

The Glue Logic: An Integrated Programming/Execution Environment for Distributed Manufacturing Work-Cell Control System

M. Takata
The University of Electro-Communications
*1-5-1, Chofugaoka, Chofu, Tokyo 182, Japan, Tel:+81-424-83-2161 ext.4744, Fax:+81-424-84-1598, email:*takata@cc.uec.ac.jp

E. Arai
Osaka University
2-1, Yamada-oka, Suita, Osaka 565, Japan, Tel: +81-6-879-7555, Fax: +81-6-879-7570, email: arai@mapse.eng.osaka-u.ac.jp

Abstract

In this paper, a infrastructural system designed for the factory automation applications, named "Glue Logic", is described. The Glue Logic provides an environment which supports the programming, controlling and monitoring the manufacturing system, mainly in the levels of manufacturing work-cells. And this environment also supports efficient manufacturing programming, by means of high program modularity and reusability.

Using the service of the Glue Logic, users can easily implement data-sharing and task-interlocking among multiple application processes developed and compiled separately. Furthermore, this system includes event notification message sending and condition monitoring features to eliminate needs of data polling by means of active database technique.

As the result of the use of the Glue Logic, the application system can be build as a collection of the agents working together, and the Glue Logic itself acts as a message exchange and shared data space manager.

Keywords

Factory Automation, Manufacturing work-cell control system, Infrastructural Software System, Distributed Programming / Execution Environment

J. Goossenaerts et al. (eds.), *Information Infrastructure Systems for Manufacturing*

1 INTRODUCTION

The Glue Logic is an infrastructural system which is designed to make building manufacturing work-cell control systems easy and flexible (Takata 1995) (Takata 1993). This system binds multiple application software modules developed and compiled separately, and coordinates those modules by means of inter-process massage passing.

As the Glue Logic supports event notification and condition monitoring features based on active database scheme, users can easily build event-driven application programs. Each program modules are free from polling shared data, waiting for notification messages.

All the data and application modules in a system can be represented by symbolic names defined in the Glue Logic, and are accessed without any knowledge about implementation of other modules. Each modules in an application system can be developed concurrently, and can be added, deleted or changed freely without modifying other existing modules.

As the result, Glue Logic compliant software modules are easy to re-use, and the users can build large libraries of application software modules. Furthermore, the life cycle and the reliability of such modules are extended, and their development cost is greatly reduced.

This paper describes the Glue Logic designed as the infrastructural system for factory automation applications, and is now under prototyping. In Section 2, the characteristics of the manufacturing work-cell control softwares are discussed. Section 3 discusses the description of the Glue Logic, and lastly, Section 4 discusses on the design of a coming global programming language, to which the Glue Logic provides its execution environment.

2 TARGET OF THE GLUE LOGIC

2.1 Target system controlled

The Glue Logic is designed for controlling manufacturing work-cells, which are operated in the flexible floor shop manufacturing systems. The manufacturing work-cell considered here consists of a work-cell control system and devices such as NC machines, assemble machines, robots, conveyers, storage systems, sensors, actuators, and so on.

Currently, there are many micro-processors in a manufacturing work-cell, which are used within FA controllers, NC controllers, robot controllers and Programmable Logic Controllers. But the rapid progress of recent technology implements low-cost high-performance micro-processors, and also implements the possibility to have integrated manufacturing work-cell controllers, which controles NC's, robots and DI/DO directly.

In order to operate the integrated manufacturing work-cell controllers, the software systems should have steady foundations, that is the execution environments. The aim of designing the Glue Logic is to provide an environment described above.

2.2 Requirements for the Glue Logic

Based on the observation of the current manufacturing work-cell control software systems, the manufacturing work-cell control software systems and its support system should have following characteristics:

- Manufacturing control software systems should have sense of time.
- As multiple processes run concurrently, abilities to control other processes are required. For the maintenance, those operation should be denoted concisely in program texts.
- Exception handling and the execution flow control of the exception handlers are very important for the application.
- Abilities to exchange information among multiple processes running on multiple work-cell control systems are vital for the software.
- Abilities of information sharing and keeping information consistent are required.
- Program modules are frequently added, deleted and altered, even in the manufacturing line is under operation.

In order to ease programming application systems, the following system design paradigms are strongly recommended:

- Building manufacturing control program system as the collection of modules.
- An infrastructural system should be introduced to bind application program modules, and each module should be designed to be the infrastructural system compliant.
- The cell models can be build in the shared data space in the manufacturing work-cells, in order to virtualize devices and work-piece in a work-cell.
- The inter-process communication should be done in the form of the message passing, in order to virtualize manufacturing control processes.

Under the paradigms shown above, the infrastructural system should be designed to be the minimum system to bind program modules flexibly, and should fulfill following requirements:

- Data sharing among independent modules should be realized.
- Each module can be separately developed and compiled.
- Modules are bound at run-time and are easy to add, delete and substitute.
- Data and modules can be accessed in fully abstracted ways.
- Keeping shared data being consistent and implementing mutual execution operations.
- Models all objects in the work-cell within shared data space.
- Models all event occurrence by message sending.

2.3 Design goal

The design goals of the Glue Logic are as followings:

- The Glue Logic should assist manufacturing control software systems to have an architecture which ease integrating and coordinating multiple application program modules.
- With the use of the Glue Logic, re-using the manufacturing control system software should be easy and the software life cycle should be extended.
- The global control structure and the local control structures of the manufacturing control software system should be separated.
- In order to integrate modules flexibly, the global control structure should be executed in a interpretive way at execution time.

In order to fulfill the goals shown above, following approaches are taken:

- Implementing the manufacturing control software system as the collection of the *agents*, and realize their coordination by means of message passing.
- Use of the *active database* scheme as the shared data space among the application processes, which is a *blackboard* system with the notification message sending feature to synchronize processes.
- Making all messages to send by way of the Glue Logic, in order to make application processes independent from others.
- Centralize shared data within the Glue Logic, in order to keep shared data consistent.
- Virtualizing devices and shared data by the names, and occurrences of events by message sending.
- Virtualizing application processes by introducing the names representing them, and by implementing the feature to send notification messages on assignment to such names.

With this scheme, following advantages are expected:

- The application modules become highly re-usable and have long life cycle.
- The application modules can be written as event-driven system.
- As the relations among application modules are expressed as the message sending rules, such relations are easily re-configured and evaluated interpretively at the run-time.
- All the machine tools and the objects are represented as agents, and are represented as a name in the shared data space in the Glue Logic.
- The implementation description of the agents are hidden, and their interface become simple.
- It is easy to virtualize things and program modules in the manufacturing work-cell.

3 THE GLUE LOGIC

3.1 Architecture

The Glue Logic has been developed to support application programs by means of data sharing, event notification and condition monitoring. As the system uses inter-process communication internally over the network, the Glue Logic can play the roles of the infrastructure of the distributed manufacturing work-cell control systems. This makes development and maintenance of the event driven application easier.

Furthermore, the Glue Logic is effective not only in the distributed work-cell environment, but also in the single work-cell system, to keep application programs simple and highly independent from other program modules.

The Glue Logic is used in a configuration shown in Figure 1. In this figure, the shaded part shows the Glue Logic, and the boxes in the right half represent application processes which utilize the function of the Glue Logic.

The Glue Logic relays all inter-process communication among its application processes, and manages all data shared by those application process. Because of this, the Glue Logic

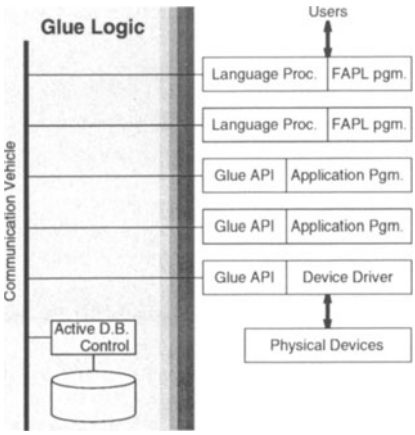

Figure 1 Architecture of the Glue Logic.

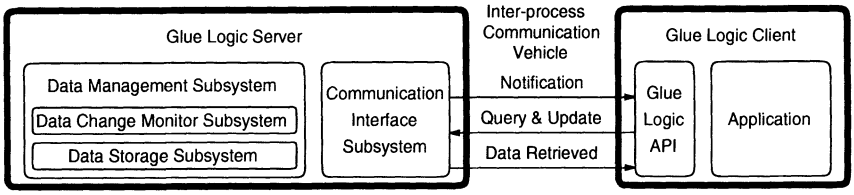

Figure 2 Configuration of the Glue Logic.

can send the change notification messages, when the values of the shared data are altered. As the virtualizing the counterpart of the communication can be achieved by relaying all of the inter-process communication, each application module can be independent from adding, deleting and altering other modules.

The application module programs can be written, using the Glue Logic API (Application Programming Interface) and a general purpose programming language, or using a global Factory Automation Programming Language (FAPL) described later. Developing new programs using FAPL is easy, but to save the existing software assets, users can convert old programs using the Glue Logic API.

3.2 Overall Implementation

In the first phase, the design of the Glue Logic is based on the client-server model of transaction processing, as shown in Figure 2, though there is no need for the users to know about its implementation.

In the prototype phase implementation, The server process of the Glue Logic is a specific process running on a specific processor. All application processes communicate only with this specific process, and there is no redundancy in this phase.

As shown in Figure 2, the Glue Logic consists of two major parts: the communication

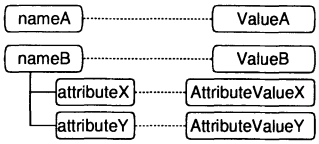

Figure 3 Basic data element of the Glue Logic.

interface subsystem and the data management subsystem. The communication interface exchanges information with other processes running in both the same work-cell controller and remote work-cell controllers connected with the network system.

The data management subsystem consists of also two parts: the data change monitor subsystem and the data storage subsystem. The data storage subsystem manages the association pair of the *name* and the *value* of the object. The data change monitor subsystem monitors the changes in the data storage subsystem and sends out the data change notification messages, and executes depending data evaluation.

3.3 Behavior of the Glue Logic

The atomic element of the Glue Logic is the tiple of a *name* and its *value*, as shown in Figure 3.

The *name* resembles variable identifier in programming languages, and can have a value. The name is a sequence of some identifiers, separated by a period, such as `abc.ijk.xyz`. Using this format, users can denote data structure by the sequence of identifiers.

Using names, the application programmers can implement arbitrary data structures. In the elements of one structure, their names contain same identifier sequence in its leading part. The trailing part of their name differs from each other. The leading common part is called a *stem* and the trailing part is called a *variant*.

Each name may have some *attributes*. The attributes denote optional characteristics of corresponding names, and the Glue Logic changes its behavior according to the values of attributes.

As the value of the name, application programs may specify one of followings; integer, floating point real, character string, expression and link. As the name itself is not typed, users may bind any types of data in turns. If a client accesses the name bounded to an expression value, the expression is evaluated and the result is used. Using the link type value, users can point another name.

3.4 Application Program Interface of the Glue Logic

The types of the API

The Glue Logic APIs can be classified into three types. They are;

- the APIs which establish, disconnect or control communication channel, or exchange messages;
- the APIs which exchange data with the data management subsystem, or operate on the `DataCell` type data;
- the APIs which control the behavior of the data management subsystem.

These APIs are designed not to depend on the implementations of hardwares, operating systems, inter-process communications, or network systems.

Network Control APIs / Communication APIs

The network control APIs open or close the communication channels. The communication APIs wait for the message arrival, or the arrival of the messages which have some special form.

Using these APIs, the application programmers can implement all basic communications with other application programs, without having any knowledge on the inter-process communications or the network programming.

Data Operation APIs

Given the names and / or the attributes, the data operation APIs refer or change the value of name specified. Some of these APIs can handle multiple names and data simultaneously, and others can change the shared data safely, in order to support multi-programming environment.

Some of these APIs are prepared to handle data with the `DataCell` structure, in order to create, destruct, copy them, or manipulate fields of them. Using these APIs, the application programmers can implement a programs which do not depend on the inside of the `DataCell` structure.

Behavior Control APIs

These APIs direct the behavior of the Glue Logic, and implements many miscellaneous features. These inclues;

- The APIs which register / deregister destinations to be informed for each names. Also controls the chance to send out the information messages.
- The APIs which copy values of names to others. Some of these can copy a substructure of the name space to another.
- The APIs which inquire the names or the attributes within the Glue Logic.
- The APIs inquire the statistic information of the Glue Logic itself.
- The APIs which control the Glue Logic itself, such as doing backup / restore the contents of the Glue Logic.

3.5 How to use the Glue Logic

Data Sharing

The most simple usage of the Glue Logic is data sharing. Originally, the Glue Logic was designed as the data management subsystem of the FA programming language system. As it is very important to share some data within multiple application processes, the Glue Logic is prepared as a separate process, in order to provide common data for those related processes.

As there are some data which is strongly related, those data values should be updated simultaneously to keep those values consistent. In the Glue Logic, there are many application program interfaces (APIs) used to access or to change values of multiple names

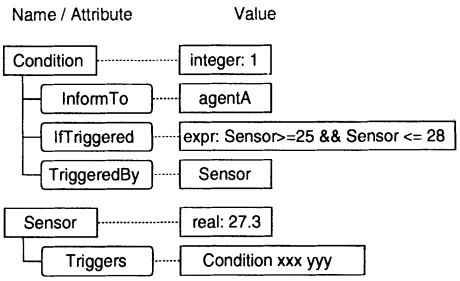

Figure 4 Data used by the condition monitoring.

with only one transaction. For the names which values are updated by multiple clients, there are other APIs to implement a semaphore, and to realize mutual access control.

Data Change Notification

In order to eliminate the needs of data polling and to decrease the network load, the feature of data change notification is used. The clients, which want to receive a change notification of a certain name's value, can register the name of the client itself to the interesting name. The name list of the notification destination processes is kept as the value of *InformTo* attribute. As the clients may register other client's name for the notification destination, the user can implement a kind of dispatcher which dispatches some processes to the events of the data change.

On the time when the Glue Logic server system receives data update request, the system searches for the clients registered as the notification destination, and then notify the fact of change to all the registered clients. In this way, the application programs are freed from polling in order to find status change.

Automatic Update of Dependent Data & Condition Monitoring

Some clients may need to know the value of name being a certain constant value, or the values of names satisfy a certain condition. The Glue Logic can be set to send a notification message only if a certain condition is met.

As shown in Figure 4, each name of the Glue Logic can have a dependence list as the values of *Triggers* and *TriggeredBy* attributes. If one or more elements of the list in the some name's *TriggeredBy* attribute is updated, the value of the name itself is updated to have the result of an expression, which is also registered as the value of *IfTriggered* attribute. If this new value differs from the former value, the data change notification is sent to its notification destinations.

With this mechanism, user can implement multi-way branching by comparing value of a name to some given constants. Usually, only one of them holds true, and the corresponding application program is notified. If multi-way branching is implemented as described above, users can add other alternatives later without changing any existing application program, but only adding new condition expressions comparing a value of name to given constants. This flexibility is implemented by the Glue Logic and the registered conditional expressions.

Message Routing System

The Glue Logic can be used as a message routing system. The message to be sent is once assigned to the name in the Glue Logic by the message sender application. Then the arrival of the message is notified to the message receiver application, which should be registered as a notification destination for the name. Lastly the notified application fetches the message from the name, and interprets the message to know what is requested.

In this case, each name in the Glue Logic represents the message receiver application, and the data assigned to the name specifies the action to be taken by the receiver application. So, from the view point of the message receiver application, the name in the Glue Logic looks like a mailbox. As the message sender need not know the actual message receiver, the interface among those modules becomes simple, and the application program modules themselves become highly reusable.

In the case of selecting only one application module from many depending on the message received, those modules shares the unique name and uses condition monitoring feature to implement multi-way branching. This implementation realizes the concept of the method selection of the object oriented language system.

Furthermore, as the actual receiver application module is selected at run-time, users can change the message processor dynamically. In this way, as the message sent can be processed by the most appropriate application module for the environment at the execution time, the message routing capability of the Glue Logic makes application systems more adaptable and autonomous.

Using Automatic Process Invocation

In the case of the Glue Logic having the process invocation capability, and when the destination process of a notification is not running, the system first invokes an application program and then sends a message to the process just invoked.

This capability is not mandatory because it is usual for real-time application systems that all application programs are started at system initialization, and all are waiting for the resume signals. But in some systems which consists of many application programs, or those which can not predict the number or kind of processes precisely, it is impossible to start all programs at initialization time. And in some applications, the system may not be requested to be operated at a fast pace. In those cases, this capability is vital or useful.

3.6 The Paradigm on using the Glue Logic

In order to utilize the Glue Logic efficiently, and make application systems easy to maintenance and modify, it is important to set up the paradigm on the programming with the Glue Logic.

Execution state of processes

As the Glue Logic uses status information on the application program processes, the information should be kept and updated correctly in the Glue Logic as the value of *status indicator* object names.

The status indicator in the Glue Logic should have a one to one correspondence to the processes of the application program module, not to the module itself.

For each application program process, there are some states that it may take in turn.

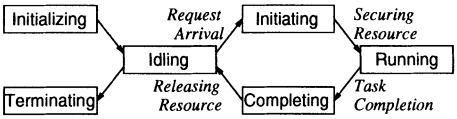

Figure 5 State Transition.

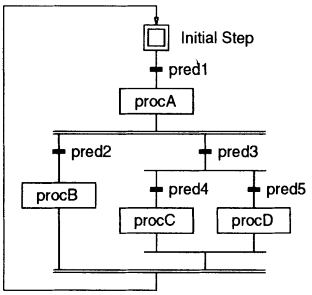

Figure 6 Example of SFC program.

Its status indicator represents its execution state by having predefined flag values for each state. The states are as follows, and the transitions are summarized as Figure 5:

Initializing State In order to start up the operation of an application program, there should be some preparation work such as initialization of the Glue Logic object names. This kind of work is done during the initializing state.

Idling State In this state, the application is idling, waiting for a start up message from the Glue Logic, which tells the meeting of the starting condition of the application process itself.

Initiating State After receiving the start up message, the application process enters the initiating state, in order to secure all shared resources it requires.

Running State In this state, the application process controls machine tools in the work-cell, processes many kinds of data, and sends out the work-piece to another work-cell. After all tasks the application should execute are completed, the process enters the completing state. Use of this transition allows the Glue Logic to start other applications.

Completing State After the task completed successfully, the resources secured by the application process should be released. The completing state is used for such operation. In case of abnormal completion, the fixing procedures vary according to its internal and external status. In this case the application should export its precise status to the Glue Logic or some other fix up processes, and then exit for the next execution.

Terminating State If the application process receives directions to halt, it enters the terminating state. Any other application then knows that the application process has already stopped.

As described above, using status indicator object names in the Glue Logic, the application program modules can be chained or executed concurrently. This feature enables an implementation of the Sequential Function Chart (SFC) (I.E.C. International Standard 1131-3 1993) with the condition monitoring feature (Takata, et.al. 1990) (Takata 1993). An example of the SFC is shown in Figure 6.

Input / Output

The read-outs of various sensors, which are required to test starting conditions for each application processes, are nice to be kept within the Glue Logic. Preparing such names in the Glue Logic, not only the application processes can use any kind of sensor read-outs

by referring to their status indicator object names, but also the starting condition of the application processes are automatically checked by the Glue Logic.

In order to reflect the status of the input in the work-cell to the status indicators in the Glue Logic, some system support processes should be especially prepared. Such process accepts interrupts from the input signal ports or scans them periodically, and updates the sensor status indicators. In some cases, some statistical operations are required in these processes. Taking a moving average value to obtain noise free data, by keeping some recent sampled values, is some of the most general operations.

On the other hand, other system support processes should be prepared in order to control the output signal by updating the values of output control names in the Glue Logic. Those processes use the data change notification feature for the output control names, and should guarantee that the assignment to such output control names are reflected to the status to the output in certain short time period. These support processes can do much more than data passing from the output control names to the output ports. For example, a robot control process can accept a motion command in the world coordinate system, and convert it to a local coordinate system if required. This is the most simple way to implement the Abstracted Manufacturing Devices.

With these system support processes, the application can access devices attached to the work-cell through the Glue Logic, instead of touching the devices directly.

4 FACTORY AUTOMATION PROGRAMMING LANGUAGE

As the language described below has not been named yet, it is referred to as "FAPL" in this paper for convenience.

The FAPL language processor is a kind of object oriented language interpreter system (Goldberg and Robson 1983), which is very powerful to model and simulate automated manufacturing systems (Bodner et.al. 1993). The language specification of FAPL is designed assuming the existence of the Glue Logic, and the language processor relies on the Glue Logic for the global data management, as well as inter-interpreter communication, process synchronization, mutual execution and condition monitoring. In other words, the FAPL language provides an application programmer view of the Glue Logic.

As each process in the application programs is executed by respective language interpreter processes, the language processor system itself has no ability to implement concurrent processing. As the process of the FAPL language interpreter can be started from the Glue Logic, the action of a certain condition can be composed in this programming language. In this case, if the condition is met, the Glue Logic server invokes the language interpreter process and sends a message to the interpreter process just created.

5 CONCLUSION

In this paper, the target and the implementation of the Glue Logic is described. The authors believe in the effectiveness of the concept of infrastructural system for agent based processing, which binds multiple tasks to be executed in the manufacturing work-cells, defined as the methods and the processes.

There are many application program systems and operating systems which have many useful tools. But the successful systems among them have sophisticated mechanisms to integrate ready-made tools to obtain fully customized tools.

The authors would like to emphasise that the smart mechanism of the Glue Logic is the very thing to make the programming system powerful and easy to be programmed, especially in the execution environment which deals with and coordinates large and complicated application programs.

REFERENCES

Bodner, D. et. al. (1983) Object-Oriented Modeling and Simulation of Automated Control in Manufacturing, *Proceeding of 1993 IEEE International Conference on Robotics and Automation* (1993), pp.83-88.

Goldberg, A. and Robson, D. (1983) Smalltalk-80: The Language and Its Implementation, Addison Wesley.

International Electrotechnical Commission, Technical Committee 65, International Standard 1131-3 (1993-03), Programmable controllers - Part 3: programming languages

Takata, M. (1995) An Integrated Environment for Factory Automation, *Integrated Computer-Aided Engineering*, Vol.2, No.4, pp.249-263 (1995), Wiley-Interscience.

Takata, M. (1993) A Programming Environment for Factory Automation, *Proceeding of Symposium on Manufacturing Application Programming Language Environments*, Ottawa, Canada, Oct. 4-5, 1993, pp. 215-224.

Takata, M., Hasegawa, M. and Matsuka, H. (1990) A Unified Environment for Production Operation, *Proceeding of Symposium on Manufacturing Application Programming Language Environments*, Ottawa, Canada, May 14-15, 1990, pp. 85-94.

BIOGRAPHIES

Masayuki Takata, born in 1959, is an assistant professor at the University of Electro-Communications, Tokyo, Japan. He is currently studying on infrastructure systems for manufacturing control systems. His research interest includes the real-time manufacturing work-cell controls and the application of the problem solving systems to the real-time control. He received his Dr. Eng. from the University of Tokyo in 1995, on the study of the application of knowledge-base reasoning systems to the real-time control tasks.

Eiji Arai: born in 1953.1.15, graduated 1975 Univ. of Tokyo, Dept. of Precision Eng., doctorial course: 1979 Graduate School, Univ. of Tokyo, Dr., Eng. In 1979, Research Associate, Kobe Univ.; In 1983, Associate Prof., Shizuoka Univ.; In 1991, Associate Prof., Tokyo Metropolitan Univ.; In 1995, Prof., Osaka Univ.

Main Research Area includes: (1) Intelligent CAD/CAM Systems for Mechanical Products; Conceptual Design Support, Intention Modelling, Kinematic Simulation, Assembly Planning. (2) Intelligent and Distributed Production System Architecture; Production Process Description, Dynamic Scheduling, Active Database System. (3) Advanced Production Process; Intelligent Welding Machine, Modeling of Deformable Objects.