30

IMPLEMENTATION OF A DATA GATHERING SYSTEM WITH SCALABLE INTELLIGENT CONTROL ARCHITECTURE

Masayuki Takata[1] and Eiji Arai[2]
[1] *Information Processing Center, The Univ. of Electro-Communications, Japan*
[2]*Dept. of Manufacturing Science, Graduate School of Eng., Osaka Univ., Japan*
e-mail: takata@cc.uec.ac.jp

Abstract: This paper describes a system named "Glue Logic", which is a infrastructural system designed for factory automation control system, and a sample implementation of a layer structured control system architecture, which is named "Scalable Intelligent Control Architecture" reported in the DIISM2000. The "Glue Logic" supports the real-time controlling and monitoring system, by means of realizing communication and synchronizing among multiple agents. Using the active database technique, this system includes event notification message sending and condition monitoring features to eliminate data polling. This system also supports efficient programming environment, by increasing modularity and reusability of the software assets. Using the Glue Logic, we are now designing a real-time data gathering system in practical manufacturing lines according to the Scalable Intelligent Control Architecture, which permits expansion of control systems not only in spatial dimension but also in intelligence.

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

1. INTRODUCTION

The system named ``Glue Logic" is an infrastructural system which is designed to make building manufacturing work-cell control systems easy and flexible[1,2]. This system binds multiple application software modules,

referred as ``agents", and coordinates those agents 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 real-time event-driven application agents, waiting for notification messages from the Glue Logic.

As all of the data and agents in a system are abstracted, and are handled with symbolic names defined in the Glue Logic, agents can be built without any knowledge on implementation of others. As the result, the Glue Logic compliant agents are easy to re-use, and the users can build up large libraries of application agents. As some agents having rather general purpose may be shared among users, and the software development cost is greatly reduced.

This paper describes the Glue Logic designed as the infrastructural system for factory automation applications, and also a sample implementation of the real-time intelligent control system. In Section 2, description of the Glue Logic is discussed. Section 3 discusses ``Scalable Intelligent Control Architecture (SICA)," which is designed to fit multi-agent control systems, and required extension in order to implement single control system with multiple server processes of the Glue Logic. Lastly, in Section 4, the architecture of the coming multi-agent real-time control systems are discussed, and the extended Glue Logic is evaluated.

2. THE GLUE LOGIC

The Glue Logic is the minimal set of functionality to coordinate multiple real-time application agents, which consists of an active database and message passing mechanism. There are some other "middleware" systems alike, such as CORBA, but we think those systems are too complex to implement real-time embedded applications with compactness.

2.1 Architecture

The Glue Logic has been developed to support application programming 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.

The Glue Logic relays all inter-process communication among its agents, and manages all data shared by those. Because of this, 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-agent communication, each agent can be independent from adding, deleting and altering other agents.

2.2 Overall Implementation

In the first implementation of the Glue Logic, its design is based on the client-server model of transaction processing, as shown in Figure 1, though there is no need for the users to know about its implementation. All agents' processes communicate only with this server process over network, and there is no redundancy in this system.

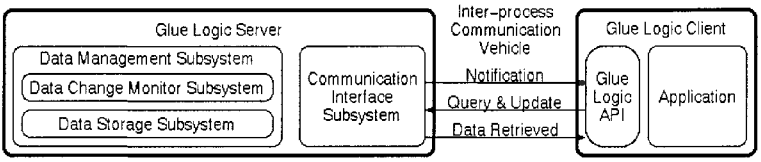

Figure 1. Configuration of the Glue Logic.

The Glue Logic consists of two major parts: the communication interface subsystem and the data management subsystem. The communication interface exchanges information with agents running in both the same work-cell controller and remote 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.

2.3 Data Handling of the Glue Logic

The atomic data element of the Glue Logic is the tuple of a *name* and its *value*. The *name* resembles variable identifier, and can have a value. The name is a sequence of identifiers, separated by a period, such as abc.ijk.xyz. Using this format, the agent programmers can implement arbitrary data structures. For all elements of one structure, their names contain same identifier sequence in their leading part, but trailing part of names differs from each other. The common part is called a *stem* and the trailing part is called a *variant*.

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 an agent accesses the name bounded to an expression value, the expression is evaluated and the result is returned as the value. Using the link, users can point another name. 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.

2.4 Interlocking Features

To share resources safely among multiple agents, the Glue Logic has the feature of interlocking operation. Single transaction from an agent to Glue Logic server process is processed indivisible way, and it may include multiple accesses, for which three access methods are prepared; read, write and compare. In general, the transaction of the Glue Logic may contain arbitrary number of accesses, and there is no restriction on order of accesses.

The transaction for interlocking compares the value of name with the value shown in the transaction, and assigns another value shown in the transaction to the name if the compared data are identical. This transaction provides means of updating shared data. When one transaction includes more than two sets of comparison and conditional assignment, users can maintain a linked list safely from insertion and deletion.

2.5 Active Database Features

In order to eliminate the needs of data polling and to decrease the network load, the feature of data change notification is introduced. The agents, which want to receive a change notification of a certain name's value, can register the ID of the agent itself to the name. The agent ID list of the notification destination is kept as the value of InformTo attribute. On the time when the Glue Logic server system receives data update request, the system searches for the agents registered as the notification destination, and then notify the fact of change to all the registered agents.

Some agents 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 2, each name in the Glue Logic can have dependence lists 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, the data change notification is sent to its notification destinations.

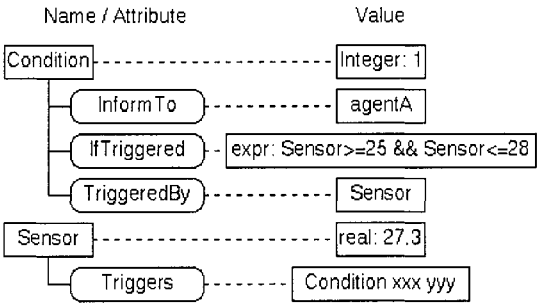

Figure 2. Data used by the condition monitoring.

3. SCALABLE INTELLIGENT CONTROL ARCHITECTURE

3.1 Overview

The control systems ever built with the Glue Logic are implemented in accordance with the ``Subsumption Architecture" proposed by Brooks[3]. With his original architecture, it is very easy to add-on much more intelligent controlling and planning abilities, by integrating much more higher level layer which monitors status of lower layer and emits directive information.

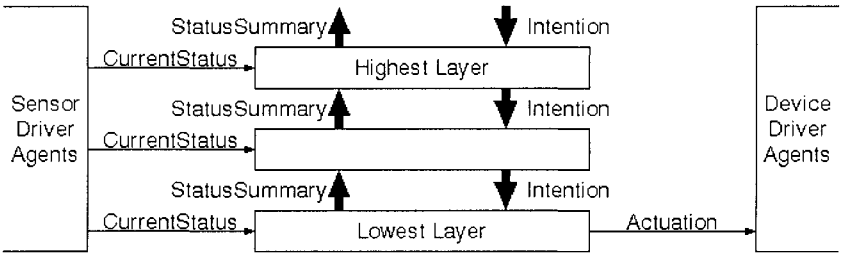

Figure 3. The Scalable Intelligent Control Architecture

The authors have modified this architecture to fit it to multi-agent control systems. In authors' ``Scalable Intelligent Control Architecture (SICA)" illustrated in Figure 3, which permits expandability not only on the breadth of system to be controlled but also on the abstractness of the controlling aim. Each layer consists of one or more agents, and their communications are supported by message exchange feature of the Glue Logic.

In our SICA paradigm, lower layers send messages to next higher ones to show the current ``summarized'' or ``abstracted'' status of the controlled system. On the other hand, higher layers send messages to next lower to show the ``intention'' of the control or ``targeting goal state,'' not to show how the lower layer to behave.

3.2 Adapting the Glue Logic to the Scalable Intelligent Control Architecture

In order to make the control systems scalable, it is necessary for the Glue Logic to have capability of data locality control. In the Scalable Intelligent Control Architecture, detail data are used frequently in lower layers, and such data are accessed from limited number of agents within those layers. This is the locality of data, and it is better for the Glue Logic to separate data into multiple Glue Logic server processes.

3.2.1 The Name Space

To achieve this aim, the naming convention of the Glue Logic should be extended to control locality of the data. Multiple Glue Logic server processes consisting the unique control system must share same name space, and the name of the data should imply the locality of the data, and also imply the place (i.e. process) the data is kept.

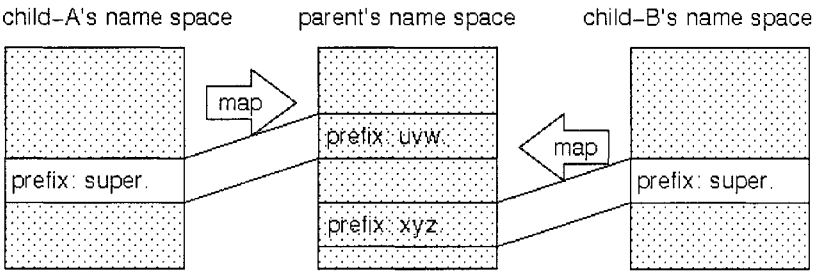

Figure 4. Name Spaces of Parent and Two Child Server Processes.

In the current implementation of the Glue Logic, we structured server processes into tree structure. The names begin with string super. show that the data with those names reside in the remote Glue Logic server process, and other names show they are not. This remote server process is called ``super'' server or ``parent'' server. In the name space of a parent server, each child server is associated to each name prefix, named ``anchor''. The data with those name prefixes are kept in the parent server itself, but are also

accessible from one of child server process of the parent server, which is associated to the name prefix. This relation is illustrated in Figure 4.

3.2.2 Communication Among Glue Logic Server Processes

With this naming convention, no server process can communicate with its brother or sister server process directly. To achieve this, a relay agent attached to the parent server process must copy information between two data area resides in the parent server itself. Such an agent is overhead for this information system, but it frees other application agents from knowing of their brothers and sisters.

Alike this, a relay agent is also required to communicate between one server process and its grandparent server process. In this case, as there is two levels of difference, the characteristics of data are also different. The data to be handled by agents in the higher layer should be summarized, in order to reduce data volume and to make those data more abstracted.

So the relay agent which relays ``vertically upward" from one server process to its grandparent server process must have summarizing feature. In the reverse direction, relaying data ``vertically downward" from one to its grandchild, a relay agent should have some functionality which divides one intention into a set of multiple sub-intentions. In many cases, this process is called ``goal deduction," and needs intelligence to solve complex problem.

3.2.3 Inter-Layer Interlocking

As the implementation of interlocking operation is based on the fact that one transaction is indivisible, there may be some troubles if the names to be interlocked resides in two Glue Logic server processes.

To keep consistent operations, we introduced some restrictions on multiple ``comparison and assignment" sets within one transaction. Thought the restriction rules are now under evaluation, the current restriction set on the order of comparison and assignment for the names implementing interlocking includes; all assignment should come after all of comparison, all comparison of name with super. prefix should come after all comparison of name without super. prefix, and all assignment of name with super. prefix should come before all assignment of name without super. prefix.

3.3 Implementation Example

Accordance with the Scalable Intelligent Control Architecture, we are designing a data gathering system. This system gathers point-of-production

data and process equipment condition data from multiple remote production facilities. Processed data can be monitored at multiple sites, and warning messages are given to the operators. This system consists of two layers, lower one is for collecting data and warning to manufacturing floor when unusual data is found, and upper one is for logging and warning to management office when abnormal data is found.

The upper layer has its own Glue Logic server process, while lower one has multiple Glue Logic server processes each of which are associated to physical manufacturing lines or cells. The lower layer gathers raw data continuously, and each data is checked against its own control limit.

4. CONCLUSION

In this paper, the new implementation of the Glue Logic and Scalable Intelligent Control Architecture are described. The authors believe in the effectiveness of the concept of infrastructural system for control agents, and the paradigm of the scalable intelligence, through some application implementation.

To develop flexible manufacturing system control software, it should take less cost, less time and more reliability, as one may have to develop a new manufacturing control software for producing only one instance. The library which consists of widely used agents is strongly required for this requirement, and the concept of our control architecture is one of the most efficient way to grow manufacturing systems up step-wisely.

The authors would like to emphasize that the smart mechanism of the Glue Logic is the very thing to make the control system powerful, intelligent, and easy to be programmed.

REFERENCES

1. Takata, M., Arai, E. (2000): ``Implementation of a Layer Structured Control System on the "Glue Logic"", *Global Engineering, Manufacturing and Enterprise Networks (Ed. Mo, J., Nemes, L.), Proc. of DIISM 2000*, pp.488 - 496, Kluwer Academic Publishers on behalf of IFIP, 2001.
2. Takata, M., Arai, E. (1997): ``The Glue Logic: An Information Infrastructure System for Distributed Manufacturing Control," *Proc. of the Int'l Conf. on Manufacturing Milestones toward the 21st Century*, pp.549 - 554, Tokyo, Japan, July 23-25, 1997.
3. Brooks, R. A. (1986): ``A Robust Layered Control System For A Mobile Robot," *IEEE Journal of Robotics and Automation*, Vol. RA-2, No. 1, 1986