Implementation of a Layer Structured Control System on the 'Glue Logic'

Masayuki Takata

The University of Electro-Communications, Japan
takata@cc.uec.ac.jp

Eiji Arai

Osaka University, Japan
aria@mapse.eng.osaka-u.ac.jp

Keywords Manufacturing work-cell control system, Infrastructural software system, Subsumption architecture, Control architecture

Abstract This paper describes an infrastructural system designed for applications in the factory automation, which is named *Glue Logic*, and a real-time control system with layer structure build on this system. The Glue Logic provides an environment which supports the programming, controlling and monitoring the production system, and also supports efficient manufacturing programming with high program modularity and re-usability. Furthermore, this system includes event notification message sending and condition monitoring features by means of active database technique. The authors proposed an architecture paradigm for scalable intelligent control systems, and implemented a model train control system on the Glue Logic as a sample, which has layered architecture derived from the *subsumption architecture*.

1 INTRODUCTION

The Glue Logic is an infrastructural system designed to make building manufacturing work-cell control systems easy and flexible [1] [2]. This system binds multiple application software modules, referred as *agents*, developed separately, and coordinates agents by means of inter-process message passing. As the Glue Logic supports event notification and condition monitoring features based on active database scheme, users can easily build event-driven applications.

All the data and agents in a system can be represented by symbolic names defined in the Glue Logic, and are accessed without any knowledge on implementation of other agents. Glue Logic compliant agents are easy to

re-use, and the users can build large libraries of application agents. This makes the life cycle and the reliability of such agents 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 the scalable intelligent control architecture paradigm to make up control systems, of which intelligence can be added in steps. Section 2 illustrates description of the Glue Logic. Section 3 discusses the programming paradigm of the layered structured control architecture proposed by the authors, a modification of the *subsumption architecture*. In Section 4, a sample scalable intelligent control system implemented in this paradigm is discussed. Lastly, in section 5, the architecture of the coming multi-agent real-time control systems are discussed, and the Glue Logic is evaluated.

2 THE GLUE LOGIC

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. 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 agents simple and highly independent from others.

The Glue Logic provides a framework shown in Figure 1. In this figure, the shaded part shows the service of the Glue Logic, playing the role of the MES. The boxes above represent application processes referred as agents, which utilize the function of the Glue Logic.

Figure 1: Framework provided by the Glue Logic.

In this figure, *def* represents the formal definition of the data to be imported from or exported to other agents. These information is centralized at the active database part of the Glue Logic.

The Glue Logic relays all inter-process communication among its agents, and manages all shared data. Because of this, the Glue Logic can send change notification messages, when the values of the shared data are altered. As the counterpart of the communication can be virtualized by relaying all the inter-agent communication, each agent can be independent from adding, deleting and altering other agents.

2.2 IMPLEMENTATION

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 general users to know about its implementation.

Figure 2: Configuration of the Glue Logic

The server process of the Glue Logic is a specific process running on a specific processor. All agents' processes communicate only with this server process over network, and there is no redundancy in this system.

The server process consists of two major parts: the communication interface subsystem and the data management subsystem. The former exchanges information with agents running in both the same work-cell controller and remote work-cell controllers over the network.

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 of the *name* and the *value*. 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 BEHAVIOR OF THE GLUE LOGIC

The atomic 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 some identifiers, separated by a period, such as

abc.ijk.xyz. Using this format, users can denote data structure by the sequence of identifiers. The agent 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 an agent accesses the name bounded to an expression type value, the expression is evaluated and the result is returned as the value. Using the link, users can point another name. This is equivalent to the *symbolic link* used in the UNIX file systems.

2.4 ACATIVE DATABASE FEATURES

In order to eliminate 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 one's agent ID the name. The agent ID list of the notification destination is kept as the value of *Inform To* attribute.

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

Some agents using the Glue Logic 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.

Figure 3: Data used by the condition monitoring.

As shown in Figure 3, a 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.

With this mechanism, user can monitor whether sensor read-out stays within a control limit, or can implement multi-way branching by comparing value of a name to some given constants. These checking may be requested by each agents separately, users can add other sensors to be watched or branching target without modifying existing agents.

3 TOWARDS SCALABLE INTELLIGENT CONTROL ARCHITECTURE

To obtain scalable intelligent control systems, the authors started from making a programming paradigm for such control systems. The word *scalable* means that the architecture of the control systems permits the step-wise addition of much more abstracted intelligence.

The *scalable intelligent control architecture* designed by authors consists of multiple layers, and is derived from the *subsumption architecture* proposed by Brooks [3].

3.1 Subsumption Architecture

The subsumption architecture consists of multiple layers, and was designed for mobile robot control hardware systems. Each layer consists of multiple finite-state machines and are connected by low-speed serial communication links.

The layers in the subsumption architecture are divided by task-achieving behavior, not by sequence of data flow which is frequently found. Layers operate simultaneously, where lower layers continue to realize low level feed-back operations (i.e. less abstracted operations), while higher layers controls the behavior of the system by offsetting the control output of next lower layers (i.e. more abstracted, supervisory operations).

With the Brooks' subsumption architecture, it is very easy to add-on much more intelligent controlling and planning, by integrating much more higher level layer which monitors status of lower layer and emits directive information.

3.2 Scalable Intelligent Control Architecture

The authors have modified the subsumption architecture, in order to fit it to multiple software agent control systems.

The *scalable intelligent control architecture* is also one of layer structured architectures. Each layer consists of one or more software agents, and their communications are strongly supported by the Glue Logic. Lower layers process plant operational or tactical tasks, while higher layers process strategic or user-interface tasks. And, lower layers send messages to next higher to show the current *abstracted* status of the controlled system, while higher layers send messages to next lower to show the current sub-goal or the final goal. The scalable intelligent control architecture is illustrated in Figure 4.

Figure 4: The Scalable Intelligent Control Architecture.

4 IMPLEMENTATION OF THE SUBSUMPTION ARCHITECTURE WITH THE GLUE LOGIC

Using the Glue Logic, a real-time N gauge model train control system has been developed, in order to evaluate the Glue Logic itself and visualize an AGV traffic control system. In this system, a few trains are controlled within a single train layout including many points and crossings, and their speed is controlled without crashing. All trains are given their destination sections, and their routing and their speed are planned and controlled adaptively at run-time.

This system consists of about 8 kinds, approximately 20 agents in three layers, including user-interface agents.

4.1 The Model Train Layout

The train layout is divided into 48 sections, as shown in Figure 5. This train layout consists three loops and two of them are overlapping in zones E and G, and those loops are interchangeable with two X crossing points in

zones B and D. There are four escape lines to permit over-taking and passing on-coming trains in zones A, C, F and H.

Figure 5: The Railway Layout

The power driving train is provided to each sections independently. The power is controlled to realize three speeds by means of pulse width modulation. Some sections contain one points each, and those points are controlled electrically by point control agents.

4.2 Attributes of the Trains and Sections

Each trains has *destination section* and *maximum speed* as their attributes. Also they have *maximum reserved section number* as their attributes, which limits the number of sections to be exclusively reserved before the train itself enters into them.

Each section has information of its neighboring sections. There no agent which has entire route map. In the case of the section reserved, the train ID, its destination and its maximum speed are also kept in the section data structure.

4.3 Running Control Layer

In this control system, each train should reserve the section before it enters, and it tries to reserve multiple sections if possible. This section reservation scheme is the base of system integrity, and the most important functionality of this lowest layer. In accordance with the number of reserved successive sections, the running speed of the train is designated.

If another train is ahead of a train, the train fails to reserve the section in which the other train is. As the number of reserved section decreases, the running speed of the following train is limited, or the following one is stopped until the reserved section is released.

4.4 Routing Control Layer

This layer designates route by supplying the check points (i.e. subgoals) to the destination. As the running control layer accept only one destination, this layer should pass an appropriate sub-goal as the train's current position changes.

This layer can accept a sequence of sub-goals or some combinational behaviors, such as loop *forever on outer loop*.

4.5 User-Interface Layer

This layer is the highest one because the next higher layer is the human operator. Generally, as the source of the intension is a human, the uppermost control layer consists of agents working as command receptors, status reporters and operation loggers.

In this model train control system, the user-interface layer accepts sequences of destinations for each trains, and other control parameters for them. There are also status display agents, which shows the status of the track at that moment, the status of each trains and some vital internal parameters.

4.6 Result of the Implementation

With this control system, the authors implemented the trailing behavior of multiple trains, the waiting for other trains at a crossing points and the safe merging at the junction points.

At this time, this system controls up to 4 trains, which raise about 10 events per one second, and the control system operated with the control frequency of approximately 10 Hz.

The authors are going to add on much more higher layer on the current control system, in order to implement much more intelligent behavior, such as scheduling with state forecasting using knowledge-base system agents. Even in that case, lower layers require no modification.

5 CONCLUSION

In this paper, the outline of the Glue Logic and the scalable intelligent control architecture are described. The authors believe in the effectiveness of the concept of infrastructural system for agent based processing and the feasibility of the scalable intelligent control architecture, which binds

multiple tasks to be executed in the manufacturing work-cells, defined as the agents.

To develop flexible manufacturing system control software, it should take less cost, less time and more reliability, as manufacturer 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 the subsumption architecture is the most efficient way to refine manufacturing system stepwisely.

The authors would like to emphasize 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 agents.

6 REFERENCES

- [1] Takata, M., Arai, E. (1997): "The Glue Logic: An Information Infrastructure System for Distributed Manufacturing Control," *Proc. of the Intl. Conf on Manufacturing Milestones toward the 21st Century*, The Japan Society of Mechanical Engineers, pp.545 – 554, Tokyo, Japan, July 23-25, 1997.
 - [2] Takata, M., Arai, E. (1996): "The Glue Logic: All Integrated Programming/Execution Environment, for Distributed Manufacturing Work-Cell Control System." *Information Infrastructure Systems for Manufacturing* (Ed. Goossenaerts, J., Kimura, F., Wortmann. H.), *Proc. of DIISM '96*, pp.181 - 192, Chapman & Hall, 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
-