2 IMPLEMENTATION ISSUES WITH HOLONIC CONTROL DEVICE COMMUNICATION INTERFACES

Jason J. Scarlett[1], Robert W. Brennan[1], Francisco Maturana[2], Ken Hall[2], Vladimir Marik[3] and Douglas H. Norrie[1]

[1]*Department of Mechanical and Manufacturing Engineering*
University of Calgary, 2500 University Dr. N.W. Calgary,CANADA T2N 1N4
[2]*Rockwell Automation Advanced Technologies*
Allen Bradley Drive 1, Mayfield Heights, OH, USA, 44121
[3]*Rockwell Automation*
Americka 22, Praha, CZECH REPUBLIC, 120 00

This paper focuses on implementation issues at the interface between holonic control devices (HCDs) and agent-based systems. In particular, we look at a function block-based approach to communication that is applicable to existing IEC 61131-3 systems and emerging IEC 61499 systems.

1. INTRODUCTION

In this paper we focus on the physical holons or "holonic control devices" (HCDs) that reside at the lowest level of a holonic manufacturing system (HMS) (HMS, 2004). At this level, HCDs must have the capabilities of typical embedded control devices as well as the ability to function in the larger holonic system. In other words, HCDs must interface with the sensors and actuators of the physical processing equipment and provide the real-time control functions that implement and monitor the required sequence of operations; they must also communicate with other holons to negotiate and coordinate the execution of processing plans and recovery from abnormal operations.

Although there has been a considerable amount of progress towards developing collaborative problem solving systems at the planning and scheduling level and the physical device level of the manufacturing enterprise (McFarlane and Bussmann, 2000) there has been very little work on tying these worlds together. In other words, without an effective real-time interface between the information world (i.e., software agents) and the physical world (i.e., physical agents or holons), agents and machines will continue to exist and operate largely apart as they do today.

One of the main barriers is the very different approach to software development at these two levels. This is primarily because of the need to satisfy real-time requirements at the device level, but also because of the historical evolution of

industrial control (e.g., ladder logic's relationship to relay wiring diagrams). Recent international standards efforts such as the International Electrotechnical Commission's IEC 61131-3 (Lewis, 1996) and IEC 61499 (IEC, 2000) standards have made progress in addressing the issues of open programming languages and distributed control models, however the issue of interfacing industrial control software to agent-based software remains.

A second area of concern is that of inter-holon communication. Within each HCD, the distributed intelligence that sets them apart from typical embedded controllers is enabled by software agents that are capable of communicating with other agents (and holons) through message passing. Although the approach to inter-agent communication is well established at the higher levels of the manufacturing enterprise by the services of agent platforms such as FIPA-OS (FIPA, 2004) and JADE (JADE, 2004), inter-agent communication at the device level becomes more problematic. On the software agent side, well-established communication protocols (e.g., Ethernet) are typically used. However, because of the more stringent requirements for latency, reliability and availability on the physical side, specialised communication protocols (e.g., CAN (Robert Bosch, 1991) and DeviceNet (DeviceNet, 2004)) are required.

In this paper, we investigate how the low-level control (LLC) and high-level control (HLC) domains can be interfaced. The LLC and HLC architecture proposed for this integration uses function blocks for the LLC domain and software agents for the HLC domain (Christensen, ????).

The paper begins with an introduction to two possible approaches to interfacing the agent and machine worlds. We then focus on the issues that arise when implementing these approaches. In particular, we look at the advantages and disadvantages of using existing programming approaches (IEC 61131-3) at the device level and discuss the potential advantages of an IEC 61499 based approach. As well, we investigate current approaches to implementing deterministic inter-holon communication at the device level and propose an alternative approach to this problem. We also investigate the requirements for integrating low-level control language with the agent level language and communication. The paper concludes with a summary of our experiences with the real-time interface problem as well as with our suggestions for further research in this area.

2. A LOW-LEVEL INTERFACE

In this section, we look at two possible approaches to interfacing the agent and machine worlds: (i) a data-table approach as illustrated in Figure 1(a), and (ii) a function block adapter approach as illustrated in Figure 1(b).

2.1 Data Tables

Given the architecture of a programmable logic controller (PLC), the first approach is arguably the most obvious since it takes advantage of the basic memory structure and execution model of common PLCs. For example, in Figure 1 a *data table* is used to allow "messages" to be passed between the agent world and the control world. During each PLC scan cycle, state information (e.g., input and output image

table data and other addressable data) is written to a data table, which is then transformed to a format that is understandable to the agent system (e.g., FIPA Agent Communication Language (ACL) (FIPA, 2004)). As well, agent messages to the low-level control system are transformed to the appropriate data table format and read by the PLC (i.e., written to its RAM memory) during each PLC scan cycle.

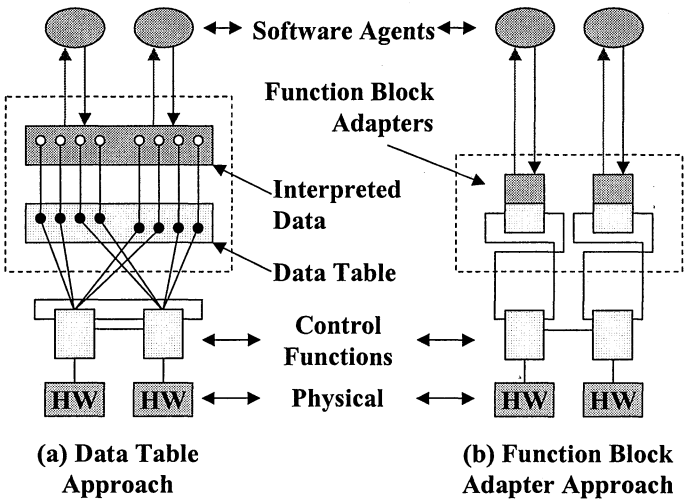

Figure 1 – A transformation interface

Although this approach is quite straight forward, it is very hardware and application dependent. For example, explicit knowledge of the PLC's addressing structure is required for this to work. As well, limitations on the amount of RAM available in the PLC for this type of data exchange may result in customisation of exactly what is read and written for each specific application.

For the remainder of section we will focus on the second approach, function block adapters, which was first proposed by Heverhagen and Tracht (2002) for IEC 61131-3 based systems. Given the "open systems" focus of the IEC 61131-3 industrial programming standard, this approach has the potential to overcome the drawbacks of the data table approach.

2.2 Function Block Adapters

Function block adapters were first proposed by Heverhagen and Tracht (2002) to provide a means of unambiguously expressing the interface mapping between IEC 61131-3 based control systems and object-oriented or agent-based software systems. To achieve this mapping, they propose a hybrid IEC 61131-3 function block, called a function block adapter (FBA) that expresses the mapping between IEC 61131-3 function blocks (Lewis, 1996) and Real-time Unified Modelling Language (RT-UML) capsules (please refer to Lyons (1998) for more information on RT-UML capsules, and Fletcher et al. (2001) for the relationship to IEC 61499 function blocks).

Given that the agent side of the system can be developed using a UML-based tool, it follows that an interface between the control software (e.g., IEC 61131-3 function blocks) and a RT-UML capsule is all that is needed for the transformation interface between the agent world and the control world.

As shown in Figure 2, Heverhagen and Tracht (2002) suggest that a hybrid IEC 61131-3 function block / RT-UML capsule can be used to map between the control world (i.e., the IEC 61131-3 function block, MyFB) and the object/agent world (i.e., the RT-UML capsule MyCapsule). The convention for IEC 61131-3 and IEC 61499 function blocks is that inputs are shown on the left and outputs are shown on the right. In Figure 2, MyFB can send messages to the object/agent system via outputs D, E, and F; messages are received from the object/agent system via inputs A, B, C. The black and white squares connecting MyCapsule and MyFBA represent the RT-UML ports.

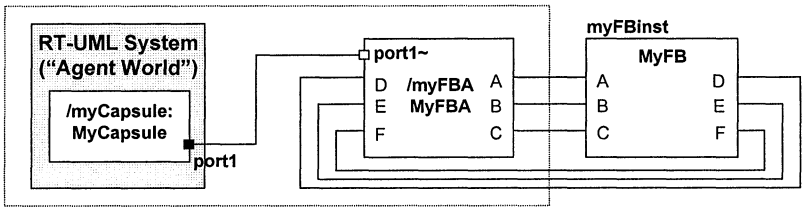

Figure 2 – IEC 61131-3 function block adapters
(from (Heverhagen and Tracht, 2002))

In order to unambiguously express the mapping between MyFB and MyCapsule, Heverhagen and Tracht proposed a simple FBA language. They note that the key to this working properly is that the interface should be simple: i.e., the interface should not specify what happens after a signal is translated and sent to a capsule or to a function block.

Figure 1(b) illustrates how we can now modify the transformation interface using function block adapters. In a more complex application however, multiple function block adapters may be used as well as multiple capsule interfaces on the agent side in order to reduce the complexity of the FBA interfaces.

Since IEC 61131-3 shares the same scan-based execution model with conventional PLC systems, the implementation of function block adapters is not as simple as Figures 1 and 2 imply. For example, Heverhagen and Tracht suggest two approaches: (i) with the FBA implemented on the object/agent side, and (ii) with the FBA split across both sides. In the next section, we investigate the use of IEC 61499 function blocks to implement FBA's. The FBA concept appears to be a closer fit with this model because of IEC 61499's event-based model and its use of service interface function blocks. This approach will be discussed in the next section.

3. IMPLEMENTATION ISSUES

In this section we summarise our experience implementing the second approach discussed in the previous section. We begin with a description of the IEC 61499

model and compare this with Heverhagen and Tracht's IEC 61131-3 approach. Next, we look at the issue of inter-object communication in a distributed real-time environment.

3.1 Function Block Adapter Implementation

On the surface, the IEC 61499 implementation of function block adapters appears to be very similar the IEC 61131-3 implementation as is illustrated in Figure 3.

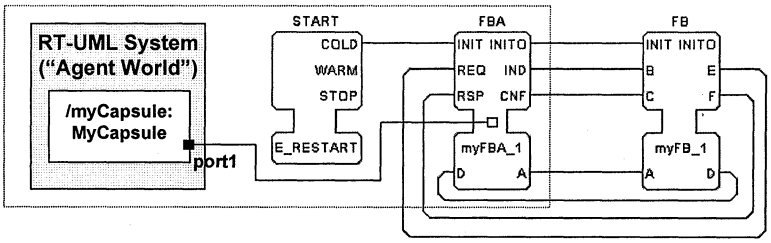

Figure 3 – IEC 61499 function block adapters

Comparing this with Figure 2 however, one can see that some of the interface is now implemented with IEC 61499 events (upper portion of the function blocks in Figure 3). In Figure 2, signals B, C, F and E are used to signal events. For example, a "true" value on B indicates that data is available to be read by input A; a "true" value on C indicates that MyFB has read the data on input A. As well, some additional information can be made available using the standard IEC 61499 protocols. For example, when MyFBA sends an event signal to MyFB's input B, it will set its QI input to "true" if data is available to be read on A; alternatively, it will set QI to "false" if there is no data available.

In order to illustrate this approach, we show the two basic forms of data transfer in Figures 4 and 5: agent or capsule initiated transfer and function block initiated transfer respectively.

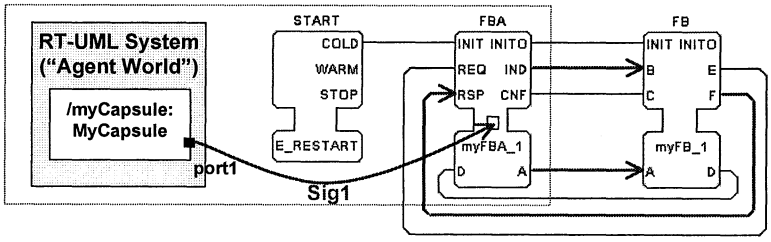

Figure 4 – Capsule initiated data transfer

In Figure 4, communication is initiated by a capsule (i.e., representing a software object or agent) in the "agent world". The capsule sends its data (i.e., "Sig1") via port1. This data is then made available on output "A" of the IEC 61499 function block adapter (i.e., "FBA"). FBA next indicates that data has been received and is

available by initiating event "IND". "FB" then acknowledges receipt of the data by issuing event "F" (this is received on FBA's "RSP" event input). It should be noted that no message is sent to the capsule if communication is asynchronous.

Figure 5 illustrates synchronous communication that is initiated by the low-level control system. In this case, data is made available at output "D" of FB. When FB is ready to send this data to the higher-level agent system, it signals FBA with output event "E". This initiates an "REQ" event on FBA's input, which in turn results in the data being sent to the agent system (i.e., "Sig2"). In this case, the agent system acknowledges the transmission with "Sig3" via port1, allowing FBA to confirm to FB that its data was received (i.e., FBA issues a "CNF" event to FB).

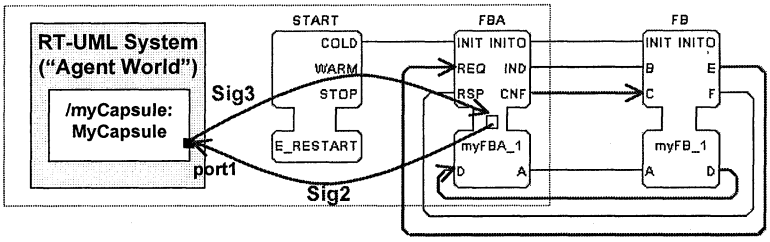

Figure 5 – Function Block initiated data transfer

As noted previously, the use of IEC 61499 event connections simplifies this approach. Arguably, the more significant difference in the implementation however, is that MyFBA is implemented as an IEC 61499 service interface function block (SIFB). As the name implies, interface function blocks provide services to the function block application. For example, resource initiated services such as a subscriber interface (to an Ethernet connection) or an analogue-to-digital converter interface can be implemented as a SIFB. Similarly, application initiated services such as a publisher (to an Ethernet connection) or a digital-to-analogue converter interface can be implemented as a SIFB.

As a result, the specialised hybrid function block / capsule (shown in the centre of Figure 2) is no longer required. For example, in the IEC 61499 implementation, the FBA shown in Figures 4 and 5 is a composite function block consisting of a FBA controller and a publisher/subscriber pair as shown in Figure 6.

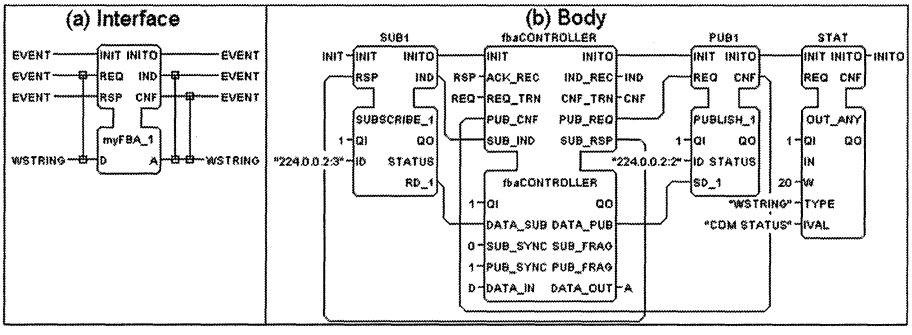

Figure 6 – Composite Function Block Adapter in IEC 61499

The FBA controller (fbaCONTROLLER) carries out the same basic functionality as the IEC 61131-3 FBA; the publisher/subscriber pair consists of two standard IEC 61499 SIFB's (SUB1 and PUB1) that in this case access Ethernet communication services. For agent-to-function block communication, the Ethernet protocol is sufficient in most cases. However, for function block-to-function block communication, a deterministic communications protocol is more appropriate as will be discussed in the next subsection.

3.2 Communication Protocols

Like other safety-critical systems, holonic systems at the device level inhabit an environment where incorrect operation can result in the harm of personnel and/or equipment (Storey, 1996). In a real-time distributed system, the overall integrity of the system is tightly linked to the integrity of the communication network. The suitability of a specific protocol for safety-critical applications must consider a wide range of issues such as redundancy, data validation, fault isolation, and timing. At the device level, or the level of inter-HCD communication, it is important to be able to guarantee the delivery of messages. As a result, a real-time embedded system protocol such as TTCAN (Marsh, 2003), FTT-CAN (Ferreira et al., 2001), TTP/C (Marsh, 2003), Byteflight (Kopetz, 2001), or FlexRay (Kopetz, 2001) is appropriate at this level.

Real-time protocols fall into two main categories: event-based and time-based protocols. Much of the discussion about choosing a protocol begins with the assumption that time-triggered protocols are the only ones suited to safety-critical applications. This assumption is based on the belief that time-triggered schemes are deterministic (higher degree of predictability) and event-based schemes are not (Claesson et al., 2003). For example, it is argued that it is not possible to predict the latency of event-based systems because of the uncertainties involved with arbitration. Another way to state this is that in an event-based system, the latency of messages changes depending on the volume of network traffic. This variation introduces a sense of uncertainty that some claim cannot be tolerated in a safety-critical environment. On the other hand, a purely time-triggered system will always have the same delivery delay times, bringing a sense of certainty to the network.

However, given the event-based model described in the previous section (i.e., IEC 61499), an event-based communication protocol would provide a closer match. Traditionally the uncertainty in message delivery makes time-triggered the preferred option. However, introducing a priority to an event-based system may be able to address the issue of uncertainty. The literature on safety-critical communication protocols does not include an event-based protocol that employs message priorities to deterministically describe the messaging delays. The authors are currently investigating an alternative approach to existing time-triggered protocols that uses dynamic priority setting (Scarlett et al., 2004). This approach appears very promising, resulting in a protocol that nicely matches the interface implementation described in the previous subsection.

4. CONCLUSIONS

In this paper we have presented two approaches to implementing the low-level interface between the information world (i.e., object/agent systems) and the physical world (i.e., PLC systems). The focus of our work has primarily been on the second approach, which involves the use of a special type of function block (a function block adapter or FBA) that allows unambiguous mapping between both sides. Given the event-based, distributed nature of the IEC 61499 model, this approach appears to be well suited to the notion of a FBA service. In this case, implementing a FBA in IEC 61499 does not require a hybrid function block as it does in IEC 61131-3; instead, the FBA can be thought of as a specific SIFB type.

Our current work in this area is focusing on refining the implementation of holonic control devices. In particular, we are focusing on the issue of inter-HCD communication as noted in section 3.2. Initial simulation results with our proposed event-based, dynamic priority communication protocol have indicated that the protocol is very flexible and result in real-time performance that is comparable to existing time-based protocols (Scarlett et al., 2004). We are now investigating a physical implementation of this communication protocol using the Systronix aJile Euroboard (SaJe, 2004) platform.

5. REFERENCES

1. Christensen, J. H., HMS/FB Architecture and its Implementation in S.M. Deen (ed.), Agent-Based Manufacturing. Berlin/Heidelberg: Springer-Verlag, 2003, pp. 53-87.
2. Claesson, V., Ekelin, C., Suri, N., (2003) "The event-triggered and time-triggered medium-access methods," In *Proceedings of the IEEE International Symposium on Object-Oriented Real-Time Distributed Computing* (ISORC'03).
3. DeviceNet (2004) ODVA Web Site, http://www.odva.org/.
4. M. Fletcher, R. W. Brennan, and D. H. Norrie, "Design and evaluation of real-time distributed manufacturing control systems using UML Capsules," *7th International Conference on Object-oriented Information Systems*, Springer-Verlag, pp. 382-386, Calgary, 27-29 August, 2001.
5. Ferreira, Pedreiras, Almeida & Fonseca, "The FTT-CAN protocol for flexibility in safety-critical systems," *IEEE Micro*, pp. 81-92.
6. Foundation for Intelligent Physical Agents (2004) Web Site, http://www.fipa.org/.
7. Heverhagen, T. and Tracht, R. "Implementing function block adapters", *Lecture Notes in Infomatics*, Verlag, pp. 122-134, 2002.
8. IEC TC65/WG6 (2000) Voting Draft – Publicly Available Specification - Function Blocks for Industrial Process-measurement and Control Systems, Part 1-Architecture, International Electrotechnical Commission.
9. Java Agent Development Framework (2004) Web Site, http://sharon.cselt.it/projects/jade/.
10. Kopetz, H., (2001) "A comparison of TTP/C and FlexRay," TU Wien Research Report 2001/10.
11. Lewis, R. (1996) *Programming Industrial Control Systems using IEC 1131-3*, IEE.
12. A. Lyons, "UML for real-time overview," *Technical Report of ObjecTime Ltd*, 1998.
13. Marsh, D., (2003) "Network protocols compete for highway supremacy," *EDN Europe*, pp. 26-38.
14. McFarlane, D. C., and S. Bussman (2000) "Developments in Holonic Production Planning and Control", *International Journal of Production Planning and Control.*
15. Robert Bosch GmbH. (1991). Bosch CAN Specification version 2.0, Retrieved April 8, 2003 from http://www.can.bosch.com/docu/can2spec.pdf
16. SaJe, "Real time native exectution," http://www.systronicx.com/saje/index.html, 2004.
17. Scarlett, J.J., Brennan, R.W., and Norrie, D.H., "A proposed high-integrity communication interface for intelligent real-time control," Submitted to *Intelligent Manufacturing Systems Forum (IMS-Forum)*, 2004.
18. Storey, N. (1996) *Safety-critical Computer Systems*, Addison-Wesley.