ATLANTIS PRESS
Advances in Intelligent Systems Research, volume 130
6th International Conference on Mechatronics, Computer and Education Informationization (MCEI 2016)

Research on the connection of UML-RT capsule and Function block with Adapters

Xia Zhuang
Civil Aviation Flight University of China,
cafuczx@163.com

Keywords: UML-realtime; Control system; Function block adapter; Capsule.

Abstract. The modeling of complex control system is usually based on the UML, thus the distributed control system is generally constructed by function block. This paper studies the modeling element: Function Block Adapter, which responsible for the connection of UML-RT capsule and IEC 61131 function block. By an application to a realistic manufacturing system, the author describes the concept, structure and interface of FBA. This paper also discusses a special FBA-language, which defines the operation of FBA.

Introduction

Today's industrial manufacturing systems have to face the problem of fast changes in demand of products and in the product spectrum. The increasing complexity of the controlling software for manufacturing systems leads to the need for more powerful specification languages. Latest developments in object oriented technology like UML-RT face this need [1]. But in most cases it is not possible to substitute PLCs in existing plants completely with object oriented systems. Therefore, our approach is to integrate object oriented technology (UML-RT) into an existing PLC-environment in the case of extending a manufacturing system with new components without throwing away the PLC [2].

In this paper we discuss the new UML stereotype, the Function Block Adapter (FBA), which is responsible for the connection of UML-RT capsules and function blocks of the IEC 61131 standard. FBAs contain an interface to capsules as well as to function blocks and a description of the mapping between these interfaces. For this description a special FBA-language is provided. The FBA-language is easy to understand both to UML-RT and to IEC 61131 developers, so they can unambiguously express the interface mapping. An important advantage of the FBA-language is the possibility to use it at nearly design state of the UML-RT system [3].

Example Scenario

To explain FBAs by an example we design an assembly line case study which consists of 3 autonomous, cooperative assembly robot cells, part storage, product storage, a quality control system, and a transport system. An outline of the case study is shown in Fig.1.

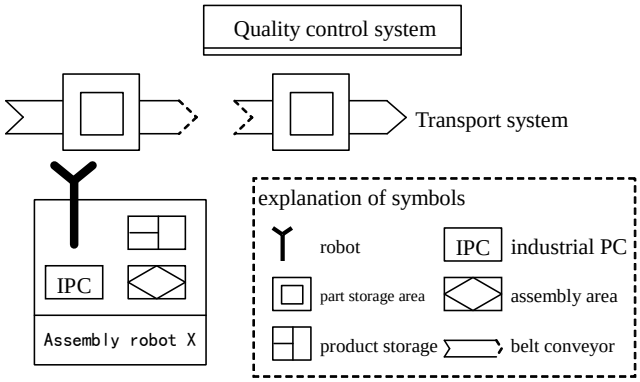

Figure 1. Outline of the assembly line case study

We assume that the assembly robot X has produced a product of type A. Now the product must be transported to the quality control. Therefore, assembly robot X sends a transport request to the transport system. This transport request contains the information about the type, sender, and receiver of the product, assembly robot X, and quality control, respectively. When the transport system receives the request, it needs some time to coordinate the transport request with transport requests of other stations. Much more time is spent on the fact, that a pallet, which can take up the product, has to be carried to the assembly robot. When such a pallet has arrived at the assembly robot X, the transport system tells the assembly robot to put the product on the pallet. This also needs some time for the robot to perform its movements [4]. During this time the transport system may not move the pallet. It has to wait until the assembly robot sends a pallet free signal. Then the transport system can carry the pallet to the quality control, where a similar communication takes place.

In the following we concentrate on the communication between the assembly robot and the transport system. This communication consists of the three signals: transport request, put product and pallet free.

The Function Block Interface

As stated in section 2 each IEC 61131 program can be viewed as a Function Block. In Fig. 2 the Function Block for the transport system is displayed. Its name is FB_TRANS.

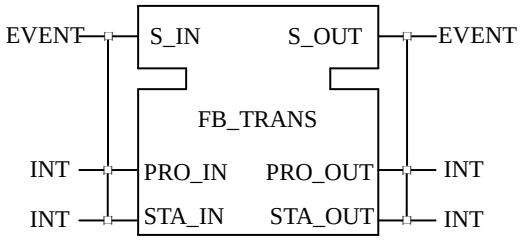

Figure 2. The Function Block interface of transport system

It contains three input variables S_IN, PRO_IN, STA_IN and three output variables S_OUT, PRO_OUT, STA_OUT. In the PLC system product and part types are called with numbers. Therefore PRO_IN and PRO_OUT are integer variables. They are used for product and part numbers in transport requests. Stations like robot cells or the quality control system are also called with numbers. In transport requests STA_IN and STA_OUT contain the information about the participating stations. Furthermore there are two Boolean variables S_IN and S_OUT which are used for acknowledgements in transport requests.

The transport request signal from our example scenario is shown in Fig3 by a timing diagram. Such timing diagrams are the normal way for PLC developers to show valid assignments of input and output variables of Function Blocks.

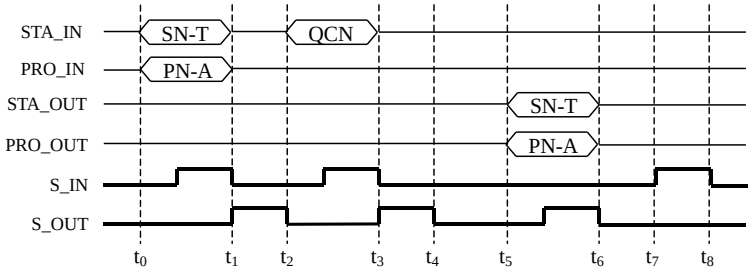

Figure 3. Timing diagram for the transport request signal

In timing diagrams time is going from left to right. On the very left side the used variables are written. In this section we have explained how a PLC developer would describe software interfaces of the transport system needed for our example scenario. The next section explains the UML-RT software interface of the assembly robot X.

The UML-RT Interface

When object oriented systems are built it is a good idea to introduce a software system architecture at an early design state. We chose UML-RT because of its clear separation of interface and implementation. UML-RT uses special stereotypes to specify interfaces of active objects. Instead of normal classes capsules are applied to model an active class. A capsule communicates over ports with its environment [6]. We use the important Capsule, Port and Connector, Protocols that have been introduced into UML-RT to support the modeling of complex real-time systems.

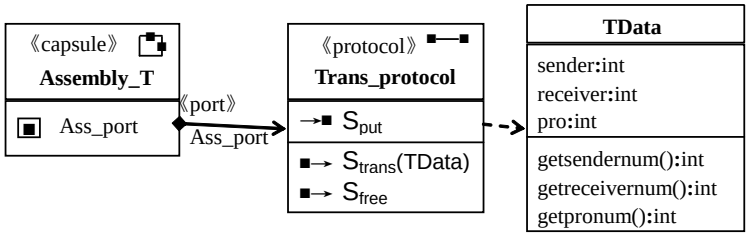

Figure 4. Class diagram of UML-RT capsule and its protocol

The Port named Ass_port is used in capsule Assembly_T, which mapping with the protocol named Trans_protocol. Trans_protocol contains an input signal Sput and two output signal Strans and Sfree. The definition of signal includes signal name and its data class. As shown in Fig. 4, the data class of Strans is TData. TData contains not only some required attribute, for example receiver, sender and product number, but also its function method.

Function Block Adapter

In Fig5 we show an extended structure diagram with a capsule, a Function Block Adapter, and a Function Block. The capsule is the system mediator. The system mediator is connected through the port ~Ass_Port to the Function Block Adapter called Trans_FBA. Trans_FBA contains a port called FBA_Port and connection lines to the interface variables of the Function Block FB_TRANS. In the structure diagram the Function Block FB_TRANS is instantiated with the name c. The Function Block Adapter is responsible for the translation of the signals coming from port FBA_Port to assignments of the input and output variables of Trans_FBA. From the side of the transport system the FBA looks like a Function Block. The variables of the transport system are assigned like explained in Fig. 3. From the view of UML-RT the FBA looks like a capsule. The system mediator can relay the messages to the Trans_FBA as it would do it to a transport system capsule which is directly designed in UML-RT.

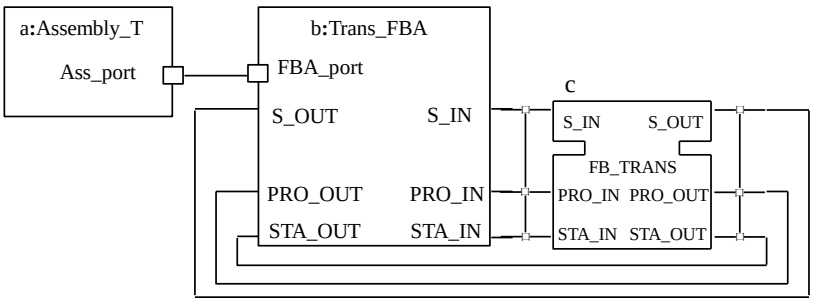

Figure 5. Extended structure diagram with a FBA

A FBA is a stereotype of a UML class which contains all properties of a capsule. A FBA uses ports to establish connections to other capsules. Additionally FBAs define interface variables for the communication with function blocks. A FBA can graphically display in an extended structure diagram (Fig.6). The FBA Trans_FBA contains a port FBA_port which is connected to Ass_port of Assembly_T as show in Fig5. Interface variables of Trans_FBA which are input variables like PRO_OUT, STA_OUT, and PRO_IN are output variables of Trans_FBA. The declaration of the interface variables has the same syntax like in IEC 61131-3 for function blocks. Ports are displayed like ports of normal capsules. Connections to other capsules or function blocks are only shown in the extended structure diagram.

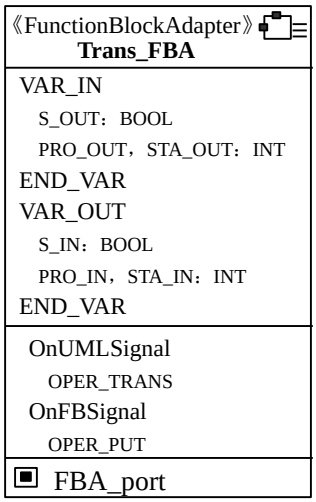

Figure 6. Class symbol for Trans_FBA

The second list compartment of Fig. 6 shows two operations of Trans_FBA. Keyword raises maps corresponding UML-Signals to FB-Signals and vice versa. This mapping is used by the priority-based selection algorithm for conflicting transitions. The FBA-Language defines operations which are called when signals arrive from a port or from the Function Block. We distinguish between operations for the translation from UML-Signals to Function-Block-Signals (FB-Signals) and operations for the translation from FB-signals to UML-Signals.

In operations of the first category two functions are needed. Delay (time) is a function that delays the execution of following commands for the time given as a parameter. Wait For (bool, time) is a function that delays the execution of following commands until the Boolean expression given as first parameter evaluates from false to true. The second parameter is a timeout, which assures that the FBA is not able to hang up. Additionally to these two functions we only need assignments. In assignments access to properties of the FBA class and used data classes is possible. Properties of UML classes are Attributes, Operations, and Association Ends. An example operation for the translation of the UML-Signal Transport_request to the FB-Signal specified in Fig3 is the following:

```
ON_UML_Signal(S1:FBA_port. Strans)
BEGIN
  S_IN:=false;
  STA_IN:=S1.getsendernum();
  PRO_IN:=S1.getpronum();
  Delay(2ms);
  S_IN:=true;
  WaitFor(S_OUT,1s);
  S_IN:= false;
  STA_IN:=0;
  PRO_IN:=0;
```

```
  WaitFor(S_OUT:=false,1s);
  STA_IN:=S1.getreceivernum();
  Delay(2ms);
  S_IN:=true;
  WaitFor(S_OUT,1s);
  S_IN:=false;
  STA_IN:=0;
ON_EXCEPTION
  S_IN:=false;
END_ON UML_Signal
```

Conclusions

Within this paper we introduced our concept of Function Block Adapters. The specification of a Function Block Adapter is completely hardware-independent. It describes only the "What" should be done for the integration and not the "How". This aspect is very important because the "How" is highly hardware-dependent.

An approach related to our FBA-Language is proposed in the State mate Approach. In state mate reactive mini-specs are used to specify data-driven activities. In our approach FBA-operations are only executed on associated signal events, which is a different semantic than data-driven activities have. For this reason we introduced the notion of a FB-Signal. Conditions on data-values are evaluated with the Wait For function. The decision if conditions are computed continuously or interrupt-driven is left to the implementation. Whereas data-driven activities are suitable for raw sensor data the FBA-Language is easier to use with IEC 61131-3 Function Blocks. With a specification given in the FBA-Language a developer has an unambiguous description of the requirements for connecting the UML-RT system to the PLC. Because of the simplicity of the FBA-Language both UML-RT developers and IEC 61131-3 developers can understand and validate the specification.

References

[1] L.Ferrarini,C.Veber.Implementation approaches for the execution model of IEC 61499 applications[J].Industrial Informatics,2012,(6):612-617.

[2] T.Heverhagen, R.Tracht.Integrating UML-RealTime and IEC 61131-3 with function block adapters [J].Object-Oriented Real-Time Distributed Computing, 2013, (5):395-402.

[3] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 2013, pp. 271–350.

[4] T.Heverhagen, R.Tracht,R.Hirschfeld.Integrating function blocks into the unified modeling language[R].SVERTS Workshop,San Francisco 2013.

[5] T. Heverhagen, R. Tracht, Negotiation Scenarios between autonomous Robot Cells in Manufacturing Automation: A Case Study, Proc. Of Tunisian-German Conference SmartSystems and Devices, Hammamet, March 2014.

[6] R.Tracht,Echtzeitanforderungen bei der Integration von Funktionsbausteinen und UML Capsules, PEARL 2001, Echtzeitkommunikation und Ethernet/Internet (P.Holleczek, B.Vogel-Heuser (Hrsg.), Informatik aktuell,Springer-Verlag 2015, S. 87-96, in german.

[7] Aymen Louati; Kamei Barkaoui; Chadlia Jerad, Time properties verification of UML/MARTE real-time systems [J]. Information Reuse and Integration, 2015, (2):386-393.