Software Architecture for an Active Device Driver in Reconfigurable Manufacturing Systems

Jeongha Shin, Duck Young Kim

Software Architecture for an Active Device Driver in Reconfigurable Manufacturing Systems

Jeongha Shin[1] [0000-0001-5495-6188] and Duck Young Kim[1*] [0000-0003-0072-4693]

[1] Pohang Univ. of Science and Technology, Pohang, South Korea
*dy.kim@postech.ac.kr

Abstract. Flexible and reconfigurable manufacturing systems aim to enable more versatile, connected, and intelligent operations not only from a system-level perspective, but also from a field-level perspective. This paper focuses on field-level applications for peripheral intelligence and proposes a concept and architecture for an Active Device Driver (ADD) that can be actively adjusted to unforeseen situations versus repeating fixed task procedures. This new device driver is designed to permit easy and rapid transformability to systems by enabling device modulization and abstraction, control and configuration under service-oriented architecture, actively supporting control reliability, and producing well-organized manufacturing data. These features can be achieved by implementing the *control I/O encapsulating module, device information model, OPC-UA server interface, and peripheral control module.* Two case studies with robot grippers are conducted to validate the working principle of ADDs.

Keywords: Automation and control, OPC-UA, Device integration, Reconfigurable manufacturing system

1 Introduction

Various customer demands and shorter product lifecycles have increased the need for the flexibility and reconfigurability of manufacturing systems. Making systems more flexible and reconfigurable involves improving three key factors, which are versatilability, connectivity, and intelligence. These factors are defined as follows. Versatilability refers to interoperable and reconfigurable software and hardware, rapidly transformable systems, and modular platforms. Connectivity defines the capacity for the information of the system to be shared asynchronously, based on standardized and robust communication protocols. Finally, intelligence means the ability of the device to rapidly make decisions and predictions by autonomous reasoning and learning from data. Such predictions and decisions allow the process to be quickly adjusted to changes in the local environment.

A variety of heterarchical architectures [1] have been proposed for rapid hardware and software reconfiguration, such as the Reconfigurable Manufacturing System [2], Multi-Agent System [3], and Holonic Manufacturing System [4] among others. Yet, most studies focus on systematic aspects. By comparison, research into field-level device control systems is understudied and more research into these systems is necessary.

Various international standards for information modeling are available, such as Asset Administration Shell (AAS) [5], OPC-UA [6] Information Model, and AutomationML (AML) [7]. These ensure that manufacturing information is sharable, stable, and organized. These international standards, as well as efforts to integrate field-devices into systems with those standards [8], [9], also require further discussions with respect to the adjustment of control logic and parameters.

Manufacturing systems are gradually changing from conventional Programmable Logic Controller(PLC)-based controls to Programmable Automation Controller(PAC)-based controls [10]. Field-level devices are also becoming more extensively connected with higher level controllers. These changes in hardware accelerate the shift from hierarchical control systems, which often have little flexibility, to much more flexible heterarchical control systems.

In practical contexts, the drivers for field-level manufacturing devices should be able to be rapidly reconfigured and easily integrated. Also, unlike existing device drivers, new drivers should actively interact with the system to ensure that objectives are reliably met. Thus, we propose an Active Device Driver (ADD) architecture that exhibits high connectivity, changeability, and intelligence in reconfigurable manufacturing systems.

2 Architecture

2.1 System Requirements

The most basic role of a driver is to make an device easier to use. For example, drivers for Personal Computers enable the device to be used without a detailed understanding of the hardware or software. Once the driver is installed, the device only requires to be supplied with electricity for use, without the need to change or interact with complex settings. We believe that drivers for manufacturing systems should enable similar ease-of-operation. Specifically, the driver should permit the device hardware to be easily controlled and data easily collected without a detailed understanding of the particular hardware and software. Moreover, the driver should support the facile rearrangement of various aspects of the device, including settings, parameters, and others. The device should also be readily usable on any system or platform. Thus, new drivers should be interoperable regardless of system architecture and control platform.

Additionally, the driver should digitize devices and support their digital representation. Device should be managed and well-organized in real-time, and the information should be easily exchanged. Finally, in traditional physical device controls, it is often difficult to understand whether each device is performing adequately or how to fix errors. As a result, oftentimes a significant process bottleneck originated by a single fault. Therefore, the driver should actively and appropriately cope with errors and failures in the system or actions.

2.2 Structure & Components of the ADD

This section describes the architecture of the ADD that was designed to meet the requirements defined in Section 2.1. **Fig. 1** shows the overall structure of the ADD, including the modules.

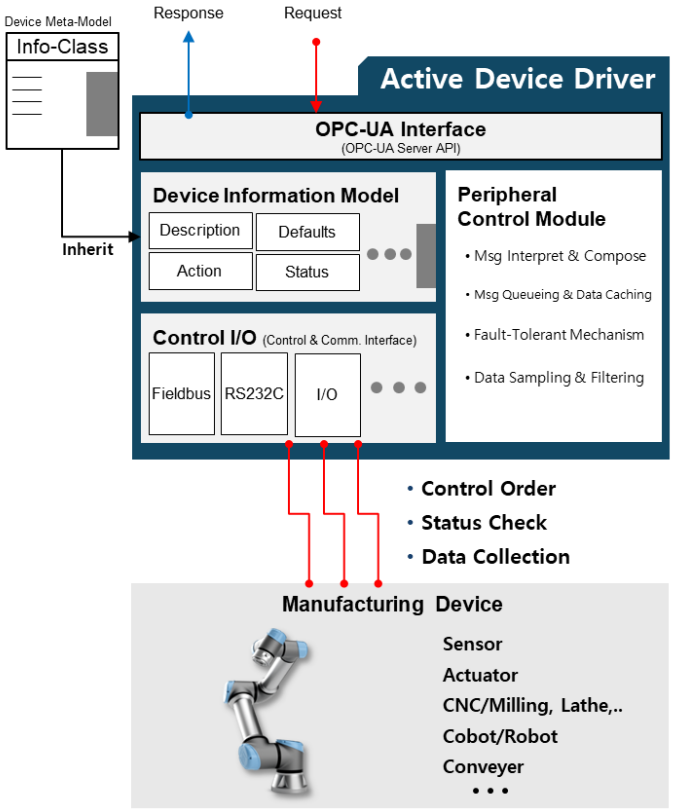

Fig. 1. Architecture of Active Device Driver

Control I/O. ADD exists in a 1:1 correspondence with the device. Here, the ADD requires an encapsulated interface as the fieldbus to physically interact with the device, which is a control I/O module.

Device Information Model. The driver has a device information model, which enables the device to be digitalized, information to be exchanged, and manufacturing data to be formed. An accurate identification code is also required to accurately recognize device information. Thus, device information models are created and managed according to the OPC-UA Information Modeling Rule [6]. An example of an information model for a robot gripper is shown in **Fig. 2**.

The device information model has four objects: *Description*, *Defaults*, *Action*, and *Status*. The *Description object* contains information about a device, such as function, form, range of movement, data unit, and data update rate, among other factors. The *Defaults object* contains information about settings, such as the movement speed of the robot and resolution of the acquired sensor data. While the *Defaults object* does not change frequently, it can be modified during device operation.

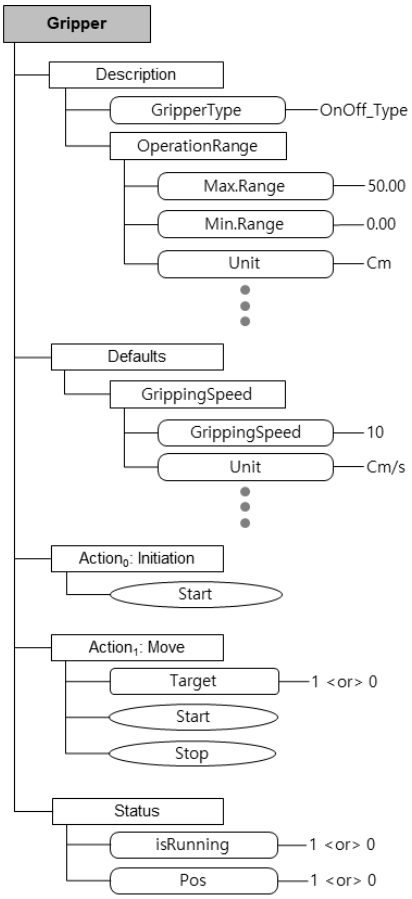

Fig. 2. Example of Device Information Model (Robot On/Off Gripper)

The *Action object* is an object that executes a task, such as a physical movement, data provision, etc. $Action_1$ through $Action_N$ objects are mapped 1:1 to N functions of the device, and the $Action_0$ objects always have an initialization function. The $Action_i$ object contains variables (e.g., target value, reference value, etc.) with the necessary information to perform an action and method objects (e.g., start, stop, etc.), which act as a trigger for real actions. For example, the information model for the robot gripper

includes one *Action object*, which is the 'Move' that opens or closes the gripper. A variable named 'Target' also exists, as well as 'Start' and 'Stop' methods. Finally, the *Status object* contains information that updates in real-time that reports the status of the device, such as manufacturing parameters (e.g., sensor values, position values of actuators, the current speed of motors, etc.). In the gripper information model that is depicted in **Fig. 2**, the gripper has two variable objects under the *Status*, namely ′ isRunning' and 'pos.'

The defined information model represents an abstract basis of the device hardware. This means that devices with the same function should have the same information model to ensure interoperability.

The information model for ADD integrates OPC-UA information modeling standards. For example, other standards such as the Asset Administration Shell (AAS), which are created by a device vendor, and AutomationML (AML), which are formed during plant or process planning, can be modified and used as an information model for ADD. Many studies have mapped AAS or AML into OPC-UA information models [11], [12]. These allow users to build the ADD information models using AAS or AML with only simple mapping information.

OPC-UA Server Interface. An OPC-UA Server API is introduced to exchange information with the system and permit device control as a service. This enables facile and rapid implementation of ADD as a Plug & Play function on any control platform. It also allows more flexible responses to sudden changes in the local environmental of the device by message-based asynchronous operations.

Control Module. The Control Module is responsible for the core functions of ADD and is based on peripheral intelligence. The detailed functionalities of the module are described as follows:

Message Interpret & Compose. Interpreting request messages received from the system and formulating adequate responses and self-configuring the response messages is required to adapt to changes in the local environment. Additionally, if any errors arise, a message that describes the situation should be automatically composed.

Message Queuing & Data Caching. Messages that are processed simultaneously must be queued. Also, temporal data storage is required to prevent data loss if delivery is not efficient due to losses in connectivity.

Data Sampling and Filtering. ADD adequately samples or filters the data based on a comprehensive understanding of the device. This reduces the time and effort to preprocess data.

Fault-Tolerable Mechanism. The fault-tolerance function first detects the faulty actions of devices, such as grasp failure, and then responds to that fault based on the Failure Mode and Effect Analysis (FMEA) rule [13]. In other words, how to respond to the fault should be adequately prepared for each device. This mechanism ensures the reliability of control in a heterogeneous manufacturing system. Faults are classified into three types: undetectable, detectable from data estimation, and self-detectable by the device. The device information model includes the information for fault classification and the fault detection algorithm. When a fault is detected, the ADD attempts to recover the fault by the predefined procedure of FMEA rules.

3 Case Study

We have conducted two case studies to validate the working principle of ADDs.

Case 1. First, we demonstrate how the control logic program for the pneumatic gripper (left in **Error! Reference source not found.**) is adaptively changed to operate the electric gripper (right in **Error! Reference source not found.**) via ADD. Parametric adjustments can be efficiently made by the common and versatile definition of functional method nodes in the device object tree of ADD. Furthermore, OPC-UA clients permit us to easily monitor the status of heterogeneous devices.

Case 2. The second case study shows an example of the fault-tolerable mechanism for gripping action. When an ADD receives a control command to close the gripper (control sequence), a predefined FMEA-based fault-tolerable gripping action will be executed. Fig. 4 illustrates an active fault-tolerable mechanism that retries the gripping action for the predefined period if it failed to get the expected sensor values. If it fails to resolve the fault over the period, then it reports the fault message to the system.

The right flow chart in Fig. 4 depicts the fault-tolerance procedure of ADD for gripping action in a FMEA flow form. The gripping action can be classified by the type of operation for which data-driven fault detection is made. The pressure sensor data are used to determine whether the gripper is holding a target object properly or not. If a defective symptom is detected during the action, an FMEA flow will be executed to correct and report the defect. This fault recovery mechanism of ADD enables devices to interpret control commands in a smart way, thus enhancing the reliability of controls.

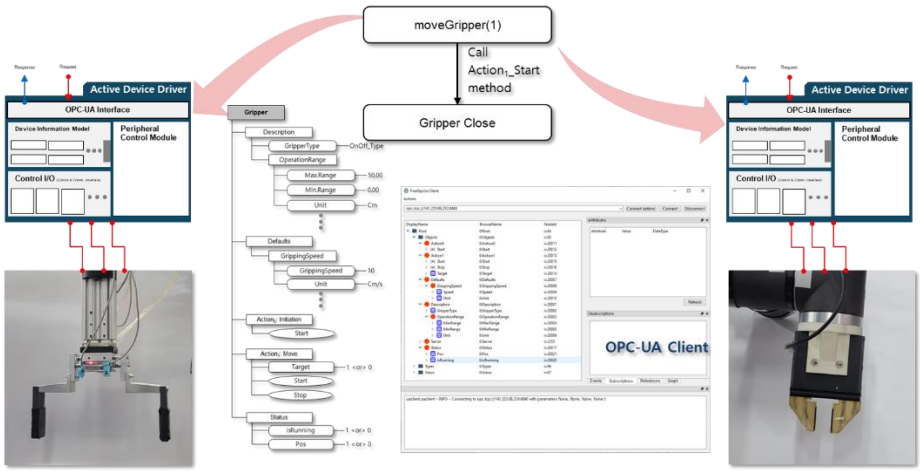

Fig. 3. Case 1: Controlling Two Heterogeneous Devices by a Single Control Logic Program

In general, device drivers operate passively according to the given commands, and they are very inflexible to reprogramming. On the other hand, the presented ADD provides the interoperability of devices based on SOA by applying OPC-UA and the proactive fault-tolerance mechanism as demonstrated in the case studies. These advantages allow abstracted control sequences to be used for control logic generation, thereby enabling more efficient system reconfiguration.

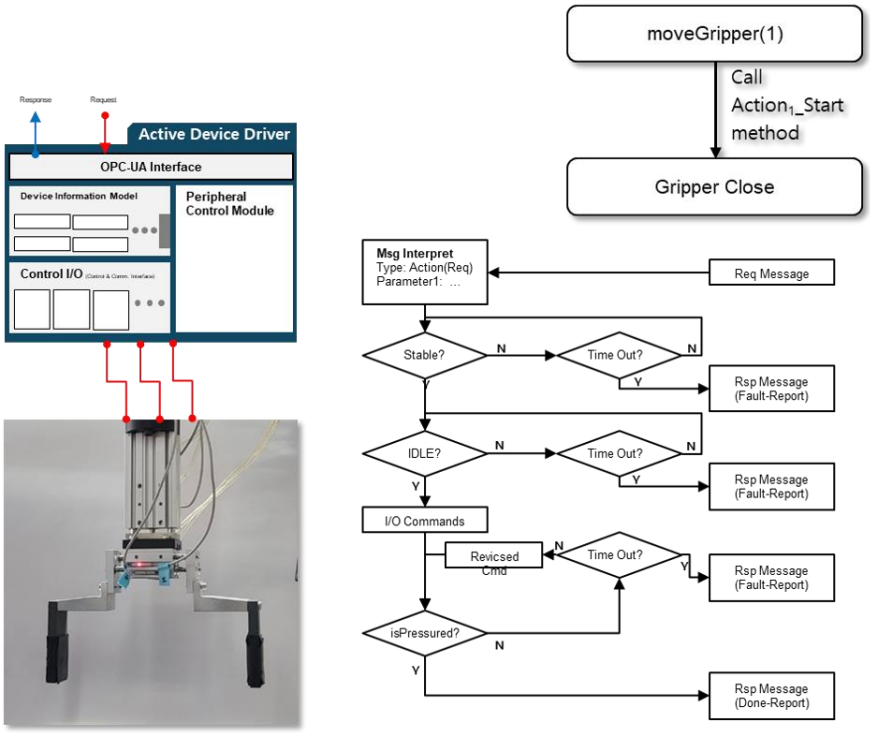

Fig. 4. Case 2: Fault-Tolerable Mechanism for Gripping Action

4 Conclusion

This paper presents the concept of ADD that is highly versatile and reconfigurable in a broad variety of manufacturing systems. We defined the functions and requirements of ADD and then proposed an architecture for ADD that would achieve those requirements. The ADD concept was verified through two case studies.

The ADD concept enables field devices to be easily and rapidly integrated regardless of the architecture and control platform of the system. The parameters or settings of the device can also be quickly changed in the system. These features make ADD a highly modular and extensible system. ADD also reports manufacturing data based on the information model of the device, while also providing hardware abstraction features. These features form the basis for predictions of future events and conditions, rapid decision making, and enabling the system to adjust the control logic. Finally, the use of

secure communications based on OPC-UA Transport and fault-tolerable mechanisms impart reliability for device operation, much like the peripheral nerves in humans, even if system operation is unimpeded. This study establishes that the ADD concept enables systems to operate reliably and respond more flexibly to environmental changes.

Acknowledgements: This research was supported by the National IT Industry Promotion Agency of Korea (NIPA) funded by the Ministry of Science and ICT (S1309-21-1001(001))

References

1. Dilts, D. M., Boyd N. P., Whorms, H. H.: The evolution of control architectures for automated manufacturing systems. Journal of Manufacturing Systems 10(1), 79–93 (1991).
2. Bortolini, M., Galizia, F. G., Mora, C.: Reconfigurable manufacturing systems: Literature review and research trend. Journal of Manufacturing Systems 49, 93–106 (2018).
3. Park, J. W., Shin, M., Kim, D. Y.: An Extended Agent Communication Framework for Rapid Reconfiguration of Distributed Manufacturing Systems. IEEE Transactions on Industrial Informatics 15(7), 3845-3855 (2019).
4. Valckenaers, P., van Brussel, H., Bongaerts, L., Wyns, J.: Holonic Manufacturing Systems. Integrated Computer-Aided Engineering 4(3), 191-201 (1997).
5. Wagner, C., Grothoff, J., Epple, U., Drath, R., Malakuti, S., Grü-ner, S., Hoffmeister, M., & Zimermann, P.: The role of the Industry 4.0 asset administration shell and the digital twin during the life cycle of a plant. In: 22nd IEEE International Conference on Emerging Technologies and Factory Automation(ETFA), pp. 1-8. IEEE, Limassol Cyprus, (2017).
6. Lehnhoff, S., Rohjans, S., Uslar, M., Mahnke, W.: OPC Unified Architecture: A Service-oriented Architecture for Smart Grids, In: 2012 First International Workshop on Software Engineering Challenges for the Smart Grid (SE-SmartGrids), IEEE, Zurich Switzerland, pp. 1-7, (2012).
7. Drath, R., Lüder, A., Peschke, J., Hundt, L.: AutomationML - The glue for seamless Automation engineering. In:2008 IEEE International Conference on Emerging Technologies and Factory Automation, pp. 616-623. IEEE, Hamburg Germany, (2008).
8. Grossmann, D., Bender, K., Danzer, B.: OPC UA based Field Device Integration. In: 2008 SICE Annual Conference, pp. 933-938. IEEE, Chofu Japan, (2008).
9. Pallasch, C., Wein, S., Hoffmann, N., Obdenbusch, M., Buch-ner, T., Waltl, J., & Brecher, C.: Edge Powered Industrial Control: Concept for Combining Cloud and Automation Technologies. In: 2018 IEEE International Conference on Edge Computing (EDGE), pp. 130-134, IEEE, San Francisco USA, (2018).
10. Bell, I.: The future of control [programmable automation controllers]. Manufacturing Engineer 84(4), 36–39 (2005).
11. Cavalieri, S., Mule, S., Salafia, M. G.: OPC UA-based Asset Administration Shell. In: 45th Annual Conference of the IEEE Industrial Electronics Society, pp. 2982-2989. IEEE, Lisbon Portugal, (2019).
12. Ye, X., Hong, S. H.: An AutomationML/OPC UA-based Industry 4.0 Solution for a Manufacturing System. In: IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 543-550. IEEE, Turin Italy, (2018).

Duffie, N. A., Chitturi, R., Mou, J. I.: Fault-tolerant heterarchical control of heterogeneous manufacturing system entities. Journal of Manufacturing Systems, 7(4), 315–328 (1988).