Asset Administration Shell for PLC

J. Maslowski[1] and T. Benešl[1]

[1]Brno University of Technology, Deparment of Control and Instrumentation, Czechia

E-mail: 221003@vut.cz, tomas.benesl@vut.cz

Abstract—This work gradually summarizes key points about Asset Administration Shell (AAS) with respect to requirements for execution in practise. It also shows how AAS could be realized in Siemens PLCs using methods which are called by OPC UA client, as well as evaluates deficiencies in the proposed implementation.

Keywords—AAS, Industry 4.0, PLC, OPC UA

1. INTRODUCTION

There has been achieved great progress in the concept of Industry 4.0 over the last decade and one of its new component is Asset Administration Shell. Nowadays, PLCs are in almost every plant from the Third Industrial Revolution. This work aims to connect these newly developed concepts together with already built infrastructure in order to improve competitiveness of individual companies with little to no requirements to invest in new, expensive devices.

2. WHAT IS ASSET ADMINISTRATION SHELL?

We can imagine Asset Administration Shell (AAS) as some kind of cousin of our well-known Digital Twin, which we can hear about in almost every article (or Ad) about the topic of Industry 4.0. The AAS got much more communication possibilities, stricter rules to follow regarding its data storage and it can decide its actions based on predefined rules. The basic scheme of AAS can be seen in Figure 1.

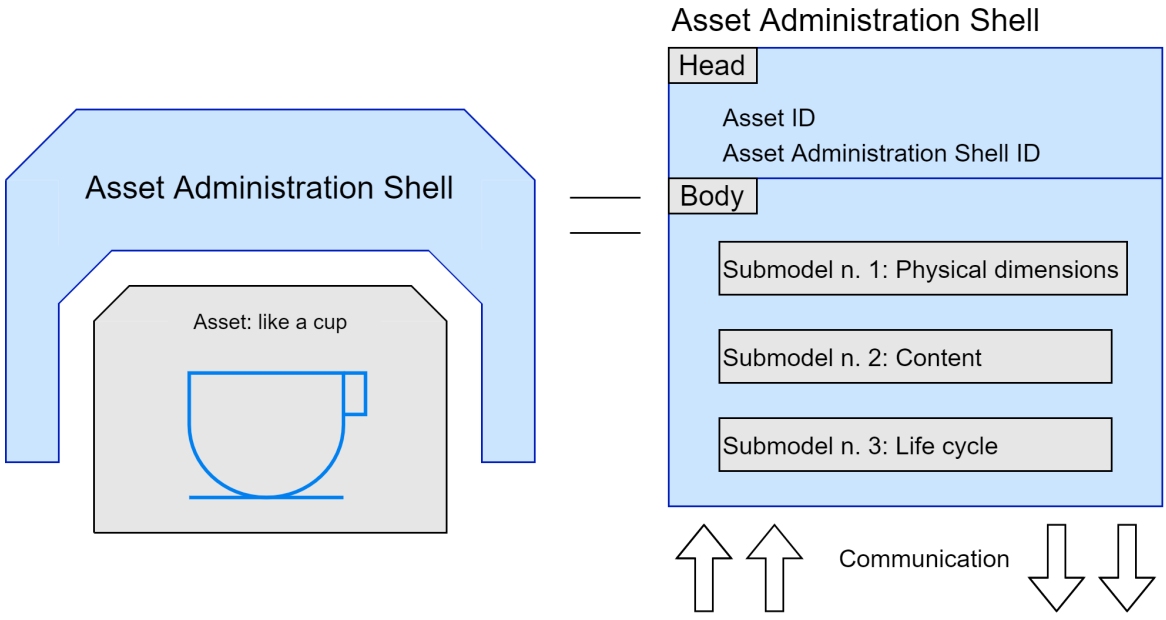

Figure 1: Asset Administration Shell.

2.1. Passive AAS

The passive AAS contains information about devices in Industry 4.0. It is composed of a head, where identification information can be found, and a body, which is divided into several submodels, each containing appropriate information about a specific device. The number of submodels varies depending

on specific requirements. It ought to react to requirements not only from production, but also from commercial or marketing areas. The AAS has to allow also other AAS and its own submodels to be referenced [1].

2.2. Active AAS

The active part of AAS allows communication with other entities that have AAS implemented too. VDI 2193-1/2 defines a unified interface for communication between devices via the I40 language. However, many of these recommendations had to be ignored because of the implementation on PLC, which has its own limitations [1] [2].
Participants in the communication are divided into 2 groups: Service Requesters and Service Pro- viders. During communication, Service Requester (SR) asks several Service Providers for a service they require and after demand negotiations (which is done by **Support** function) choose the best suited Service Provider (SP) for their specific needs. When this happens, SP is reserved by SR and aims to get into its area of influence, where those reserved actions may be taken care of. Every entity with implemented AAS should be able to change its current status from SP into SR and vice versa [3].

3. IMPLEMENTATION

The AAS was implemented into SIMATIC S7-1200 via TIA Portal v17 on 'Barman' testbed. Current version of implementation supports multiple AAS in one PLC, which reduces hardware requirements even more in a trade of added AAS limitations. This means AAS cannot change from SP into fully capable SR as well as it adds additional identification parameters for methods.

3.1. Requirements

The method of implementation this paper suggests is not able to be utilized everywhere and neither should it be. Apart from PLCs, which we can consider proudly available, it also requires infrastructure, where individual products can move across the plant with ease, eg. a robust framework of 'smart' conveyors or manipulators. The second important requirement is diversified production itself. Application of AAS into PLC makes sense only in a plant, which changes its products frequently. Meaning utilization of AAS for static production is much lower, thus usage of the old way of centralized control is better in mentioned situation.

3.2. Data model

Each AAS has one data block (DB) in PLC with all of their data divided into standardised structures. There are 7 cyclically called functions, which have only 1 input as a string message divided by '/' character and return a string with predefined structure too. It is mandatory to send both SR and SP IDs in each message in case of the implementation with multiple AAS in one PLC, so each message consists of at least 'SR ID/SP ID'. Mentioned functions are:

Submodel Each Service Provider is divided into three main groups of services it provides. The input message for this function consists of only 2 words ('SR ID/SP ID'), asking the device which type of service provider it is. It returns 'Submodel/SR ID/SP ID/submodel_type', where submodel_type can be 'Storage', 'Manipulator' or 'Mixer'.

Support The input message contains five word message with an expected format of: 'SR ID/SP ID/operation ID/material/amount'. It returns 'Support/SR ID/SP ID/value', where value is dimensionless, in range of 0-255, where 255 stands for the best and 1 for the worst cost, 0 is reserved for not supporting at all. This gives the service requester an opportunity to choose the most suitable device. The algorithm which decides the returning value is specific for every SR, depending on plant requirements.

Reserve There are currently no queues implemented in PLC, but there are expected to be some of them in the future. The input message for this function contains only 'SR ID/SP ID' and returns 'Reserve/SR ID/SP ID/True or False', depending on whether a specific device did or did not reserve its resources for a specified Service Requester. The AAS stays in reserved state for 2 minutes after receiving and successfully acknowledging a reservation, after which function Free is called.

Free The input message of Free function has the type 'SR ID/SP ID' which cancels the reservation. After that, it returns 'Free/SR ID/SP ID'. This function is used in case of malfunction of SP or just its disconnection, which could cause deadlock in a plant.

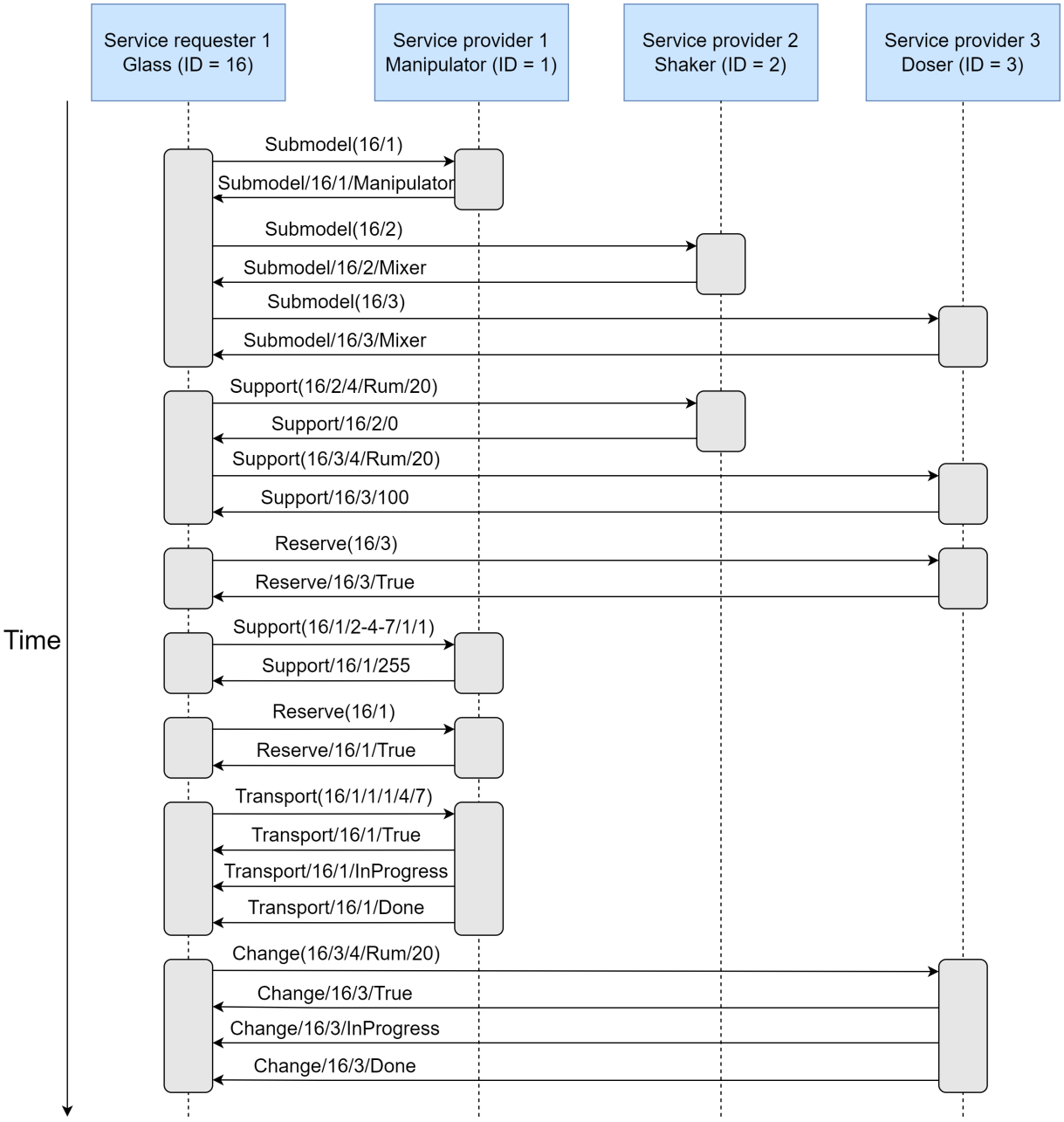

Figure 2: Example of communication.

Transport This function is available for Manipulator submodels only. Input message consists of 5 words: 'SR ID/SP ID/Asset ID/Amount/coded_starting_position/coded_final_position'. This function returns message 'Transport/SR ID/SP ID/True or False, InProgress or Stopped and Done or Failed'.

Give This function is available for Storage submodels only. Input message consists of 4 words: 'SR ID/SP ID/requested material/amount' and returns message 'Give/SR ID/SP ID/True or False, InProgress or Stopped and Done or Failed'.

Change This function is available for Mixer submodels only. The input message consists of 5 words as 'SR ID/SP ID/operation ID/material/amount' and returns message 'Change/SR ID/SP ID/True or False, InProgress or Stopped and Done or Failed'.

3.3. Communication

Communication is supported via OPC UA, where Service Requester as a client is calling predefined functions with string messages.
An example of communication can be seen in Figure 2 where the Glass as the SR is calling SP functions with an aim of carrying out its predefined recipe (saved in SR repository), which is to fill itself with 20 ml of Rum in this simple example. SR first asks what kind of submodels SPs are, then asks Mixers whether some of them can offer a service coded as '4', meaning liquid dosing. Once this service is reserved, SR is then required to get into the right position for dosing (eg. to be transported from position 4 to position 7). When SR is in position, dosing can start as the Glass requested.

3.4. Future work

Unfortunately, there are some difficulties with this implementation which should be addressed and resolved later. The first one is the queue system for reservations, which would improve continuity of production. The second major problem is the unfinished Service Requester. So far, all testing was done manually using UaExpert as a client calling PLCimp methods. Implementation program (eg. in C++), which would stand for the SR and call all required methods of Service Providers, is currently missing.

4. CONCLUSION

This work introduced the novel approach in the PLC control system by implementation of the AAS using OPC UA as a communication protocol. Decentralised control by services is a possible way to greatly improve an efficiency of specific plant types. Mentioned implementation was successfully tested manually via UaExpert and is ready for future improvement by reservation queues. It will create an environment which could be implemented in a plant if SP is implemented to represent devices.

ACKNOWLEDGMENT

The completion of this paper was made possible by the grant No. FEKT-S-20-6205 - "Research in Automation, Cybernetics and Artificial Intelligence within Industry 4.0".

REFERENCES

[1] S. Bader, E. Barnstedt and H. Bedenbender, "Details of the Asset Administration Shell. Part 1 - The exchange of information between partners in the value chain of Industrie 4.0 (Version 3.0RC01)" [online], November 2020, [cit. 03/07/2022]. Available from https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V3.html.

[2] A. Belyaev and C. Diedrich, "Specification "Demonstrator I4.0-Language"v3.0" [online], July 2019, [cit. 03/07/2022]. Available from https://www.researchgate.net/publication/334429449_Specification_Demonstrator_I40-Language_v30.

[3] M. Wenger, A. Zoitl and T. Müller, "Connecting PLCs with their asset administration shell for automatic device configuration" [online], July 2018, p. 74–79, [cit. 03/07/2022]. Available from https://ieeexplore.ieee.org/document/8472022.