Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 217 (2023) 217–226

Procedia Computer Science

www.elsevier.com/locate/procedia

4th International Conference on Industry 4.0 and Smart Manufacturing

Architecture for managing AAS-based business processes

William Ochoa[a,1,*], Felix Larrinaga[a,2], Alain Pérez[a,3]

[a]*Mondragon Unibertsitatea - Department of Information Systems, Loramendi Kalea 4, Arrasate-Mondragon, 20500 Gipuzkoa, Spain*

Abstract

Industries frequently encounter the need to orchestrate services provided by devices as business processes. These industrial business process models need to meet Industry 4.0 (I4.0) specifications to handle unpredictable scenarios in the manufacturing process. Asset Administration Shell (AAS) is considered the cornerstone of interoperability between machines and applications that compose manufacturing systems. AAS facilitates the digitization of physical things (assets) for virtual representation, turning an object into an I4.0 component. This paper investigates the usage of AAS in the context of business process orchestration and a proposal is presented based on those drawbacks. The contributions of this paper are 1) Present an architecture for the management of AAS-based business processes. 2) Introduce an AAS Submodel template that enables the description and registration of the RestServices of an asset. 3) Present a plugin for Camunda Modeler that enables the Service-Discovery mechanism from a chosen AAS repository and maps assets services into BPMN Service-Tasks. And, 4) Outline opportunities for future work between AAS and business process management systems with a primary focus on context-aware capabilities for enhancing the dynamicity of workflows.

Keywords: Service Orchestration; Asset Administration Shell; Service Discovery; Business Process Modeling; Internet of Things; Industry 4.0

1. Introduction

Smart Manufacturing systems have rapidly increased the need for orchestrating Internet of Things (IoT) services to respond in near-real-time and satisfy the demands and changing conditions of industries, supply networks, and customer needs [1]. The orchestration of these services occurs in decentralized environments in what is called Microservice architecture [2]. Often, organization and control of these decentralized machine services becomes a challenge [3, 4]. To palliate this, Reference Architectural Model for Industrie 4.0 (RAMI 4.0) was introduced. RAMI 4.0 allows

* Corresponding author. Tel.: +(34) 62786493 + Ext. 6238
E-mail addresses: william.ochoa@mondragon.edu (William Ochoa)., flarrinaga@mondragon.edu (Felix Larrinaga)., aperez@mondragon.edu (Alain Pérez).

[1] 0000-0002-4286-9914
[2] 0000-0003-1971-0048
[3] 0000-0002-6200-589X

the standardization and digital representation of I4.0 components (assets) through a three-dimensional layer model [5]. The objective is to enable interoperation and identification of assets within the network by other systems by providing their technical and operational data. To achieve this, RAMI 4.0 introduces the AAS concept [5]. By definition, AAS is a standardized digital representation of assets, the cornerstone of interoperability between applications managing manufacturing systems [6]. AAS Allows information about whole machines and their specific parts and components to be stored in the administration shell [7].

The main motivation of this paper is to introduce an architecture for managing AAS-based business processes. To the best of our knowledge, at the current stage of the state-of-the-art, there is not any approach that performs the orchestration of device services using business processes and AAS. There are some approaches using other mechanisms for service orchestration than business processes [8, 9] or they use AAS but not for enabling the Service-Discovery characteristic [10, 11]. To fill this weakness, this study introduces an architecture that aims to help industries willing to implement AAS on their architectures, in the design of manufacturing business processes.

The layout of this study is organized as next: Section 2 provides the review of the state-of-the-art concerning orchestration of services using AAS. Section 3, introduces the architecture for managing AAS-based business processes. In addition, this section presents the contribution to this architecture in the form of an AAS submodel and a plugin for a BPMN modeling tool. Section 4, provides an experiment where the contribution of the paper is implemented. Finally, Section 5 outlines the conclusions and proposes future research directions.

2. Related Work

Orchestration of IoT services is a trending topic in the domain of Smart Manufacturing systems where IoT Service-discovery plays an important role in searching for the services provided by IoT devices [12]. Orchestrations are usually formalized into business processes through the use of standard-globally known formats like BPMN, BPEL, YAWL, etc [13]. Moreover, the Object Management Group (OMG) ratified Business Process Modeling and Notation (BPMN) as the standard language for the design of business processes [14]. Concerning Workflow Management Systems (WfMS) and Business Process Management Systems (BPMS), these are systems that have the capabilities to cover the full lifecycle of business processes including design-time, run-time, and monitoring-time [15].

According to Gartner [16], an intelligent BPM Suite (iBPMS) is *"... a licensed software that supports the full cycle of business process and decision, discovery, analysis, design, implementation, execution, monitoring, and optimization"*. In [17], Gartner identifies the "top" 20 iBPMS vendors in where Camunda BPMS Platform is among them. Moreover, according to Slintel [18], Camunda has 13.47% of market share in the Workflow Automation category, just behind competitors such as Apache Airflow with 30.79% and REDCap with 14.99%. Camunda is a robust open-source platform for managing the full lifecycle of business processes [19], offering a set of components for the design, execution, and monitoring of business processes. Among those components, there is Camunda Modeler which is a business process modeling tool featuring a user-friendly interface for the design of business processes.

Concerning AAS, an asset can be turned into a digitized I4.0 component by describing and referencing one or more submodels. An asset can be i.e. physical components, hardware, documents, software, human beings, etc [20, 21]. Their properties can be described using submodels in a reflexive interface, providing high-level information about the functions, events, references, relationships, location, status, etc [22]. For instance, the *Nameplate* submodel described in [23] provides the properties to register the nameplate of an asset. This way, parties involved can use this submodel to query the physical address, serial number, year of construction, manufacturer name, etc. Thus, enabling the provision of a wide variety of data. Other submodel templates can be seen in [24, 25, 26]. However, at the current stage of the state-of-the-art, there does not exists a submodel for representing devices REST services.

Moreover, there is a lack of work done in the domain of AAS and business processes together. This is noticeable by examining the publications in each discipline. Articles in the AAS discipline deal with Cyber-Physical Systems (CPS). For instance, the authors in [11] introduced a platform for the design of production lines based on AAS by orchestrating CPS services. However, the authors designed the recipe in another format than business processes. On the other hand, papers in the domain of business processes are often related to IoT and manufacturing systems. Though, without any specific proposal about using AAS since it is still an emerging topic. For instance, [27] introduced an architecture to make BPMS aware of IoT devices, allowing the orchestration of the IoT services, at the same time, maintaining the recipe in business process format. In addition, the authors in [28], proposed an extension of BPMN to support accurately the modeling of CPS processes, adding as well resources and context data to the recipe. Similarly, the authors in [29], introduced the Web Services Business Process Execution Language (WS-BPEL) which is an extension of BPEL. WS-BPEL allows the definition of IoT-aware business processes using BPEL and runs in any business process execution engine. Another example is the work of [30], the authors introduced a framework that includes Service-Discovery and Service-Composition mechanisms for IoT service orchestration. Notwithstanding, none of these authors considered the AAS concept in their implementations.

In summary, two challenges are identified. The first one is to be able to describe the services offered by an asset in its AAS digital representation. The importance of mapping asset services into AAS comes from the importance of composing complex orchestrations in manufacturing systems [30]. The second is to provide the architecture and tools to enable the usage of AAS for the design and execution of business processes using BPMN. Therefore, this paper presents an architecture that facilitates the design of manufacturing business processes.

3. Architecture for managing AAS-based business processes

This architecture aims to enable the orchestration of AAS-based business processes. Figure 1 shows the different components of the architecture. First, *Assets* represents I4.0 physical devices at plant level. These assets are digitized through the AAS concept and their digital representations are stored in the *AAS Repository*, which could be AAS Server and/or AAS Registry depending on the vendor's implementation. AAS repository implementations should offer APIs (Application Programming Interface) for the management of the stored AASs [5]. As mentioned earlier, assets can be described using submodels. In order to describe machine services, a novel *RestServices* submodel is introduced. The *BPMN Modeling Tool* layer contains both, the Camunda Modeler and our novel Camunda plugin that enables the Service-Discovery mechanism. Thus, business analysts can design business processes out of asset services. The orchestration is then handled by the *BPMN Execution Platform*, this component comprehends any workflow executor or business process executor that can interpret and run recipes written in BPMN. Different technologies can be used at this stage, e.g WSO2, Camunda, or a Node-RED Workflow Manager presented in [31]. Thus, asset services are orchestrated accordingly.

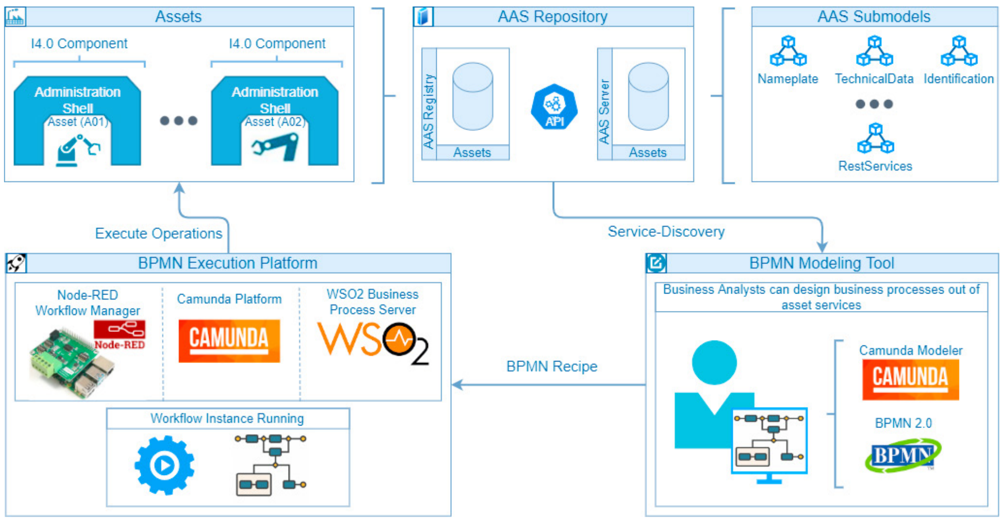
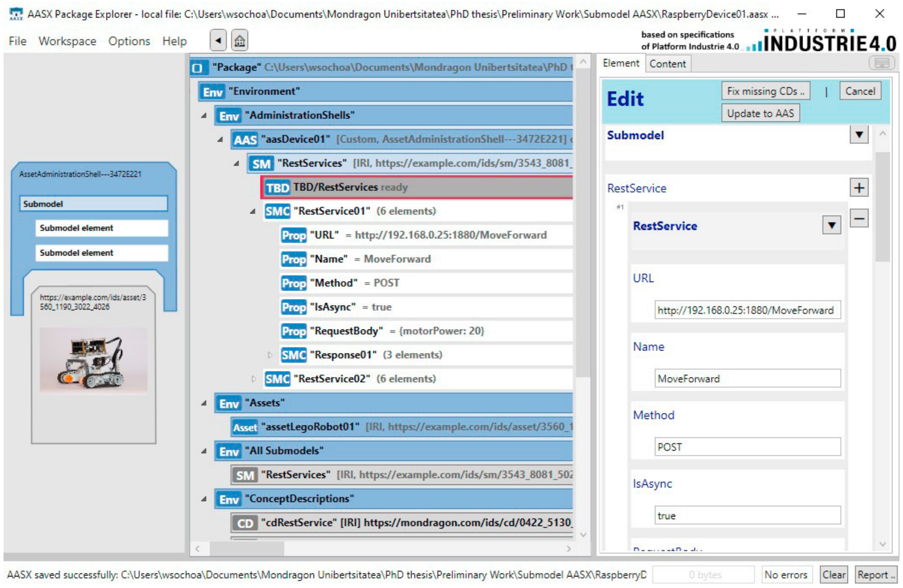


Fig. 1: Architecture for managing AAS-based business processes

The novel components of this architecture comprehend 1) An AAS submodel template called *RestServices* that allows the description of machine services and 2) A Camunda Modeler plugin called *AAS Web Service Discoverer* that enables Camunda Modeler to discover the services of the assets from a chosen AAS Repository. Moreover, these components are licensed under the Apache-2.0 license for use of the community[4]. In the next subsection, the two novel components are explained in detail.

3.1. RestServices Submodel template

In this subsection, a new *RestServices* AAS submodel is proposed in order to allow the registration of machine services. This submodel permits characterizing properties of REST services such as URL, Name, Method, IsAsync, RequestBody, and Response. These properties correspond to the basic components of a typical REST service [2]. Furthermore, the submodel could be extended with more properties according to diverse manufacturing scenarios. Figure 2 shows the definition of the new submodel on the AASX Package Explorer [5].



```
AASX Package Explorer - local file: C:\Users\wsochoa\Documents\Mondragon Unibertsitatea\PhD thesis\Preliminary Work\Submodel AASX\RaspberryDevice01.aasx ...

"Package" C:\Users\wsochoa\Documents\Mondragon Unibertsitatea\PhD
  Env "Environment"
    Env "AdministrationShells"
      AAS "aasDevice01" [Custom, AssetAdministrationShell---3472E221]
        SM "RestServices" [IRI, https://example.com/ids/sm/3543_8081_
          TBD TBD/RestServices ready
          SMC "RestService01" (6 elements)
            Prop "URL" = http://192.168.0.25:1880/MoveForward
            Prop "Name" = MoveForward
            Prop "Method" = POST
            Prop "IsAsync" = true
            Prop "RequestBody" = {motorPower: 20}
            SMC "Response01" (3 elements)
          SMC "RestService02" (6 elements)
    Env "Assets"
      Asset "assetLegoRobot01" [IRI, https://example.com/ids/asset/3560_1
    Env "All Submodels"
      SM "RestServices" [IRI, https://example.com/ids/sm/3543_8081_502
    Env "ConceptDescriptions"
      CD "cdRestService" [IRI] https://mondragon.com/ids/cd/0422_5130_

AssetAdministrationShell---3472E221
Submodel
  Submodel element
  Submodel element
https://example.com/ids/asset/3560_1190_3022_4026

Submodel
RestService
#1
  RestService
    URL: http://192.168.0.25:1880/MoveForward
    Name: MoveForward
    Method: POST
    IsAsync: true

AASX saved successfully: C:\Users\wsochoa\Documents\Mondragon Unibertsitatea\PhD thesis\Preliminary Work\Submodel AASX\RaspberryD    0 bytes
```

Fig. 2: AAS RestServices Submodel template

Tables 1, 2, and 3 break down the attributes specifications of the *RestServices* submodel. At the top level, the *RestServices* submodel contains one property of type SubmodelElementCollection and its multiplicity is set to 1..* to allow the registration of 1 to many services of an asset. Next, the *RestService* SubmodelElementCollection allows to characterize an individual service by using properties such as URL, Name, Method, IsAsync, RequestBody, and Response. Finally, the *Response* SubmodelElementCollection allows to specify how a service would respond to a request.

idShort:	RestServices		
Class:	Submodel		
semanticId:	[IRI] https://mondragon.com/ids/sm/7471_5180_3022_3423		
Parent:	RestServices		
Explanation:	A submodel template for mapping device REST services@en, Una plantilla de submodelo para mapear propiedades de un servicio REST de un dispositivo@es		
[SME type] idShort	**semanticId = [idType]value Description@en**	**[valueType] example**	**card.**
[SMC] RestService	[IRI] https://mondragon.com/ids/cd/0422_5130_5022_9484 Information about an individual Rest Service	[-] 5 elements	1..*

Table 1: RestServices Submodel Template - Full attributes specifications

[4] https://github.com/MUFacultyOfEngineering/AASBPM
[5] https://github.com/admin-shell-io/aasx-package-explorer

idShort:	RestService		
Class:	SubmodelElementCollection		
semanticId:	[IRI] https://mondragon.com/ids/cd/0422_5130_5022_9484		
Parent:	RestService		
Explanation:	Information about an individual Rest Service@en, Información respecto a un Servicio Rest@es		
[SME type] **idShort**	**semanticId = [idType]value** **Description@en**	**[valueType]** **example**	**card.**
[Property] URL	[IRI] https://mondragon.com/ids/cd/0241_0190_3022_9102 The REST service endpoint. Make sure it is reachable within the network.	[string] http://127.0.0.1/myRestService/?parameter1=val1	
[Property] Name	[IRI] https://mondragon.com/ids/cd/7052_0190_3022_2921 The name of the REST service	[string] My REST service name	
[Property] Method	[IRI] https://mondragon.com/ids/cd/1561_0190_3022_7753 The type of method	[string] GET POST PUT PATCH DELETE	
[Property] IsAsync	[IRI] https://mondragon.com/ids/cd/2582_3130_5022_8052 Specify whether or not the operation is executed asynchronously	[boolean] false true	
[Property] RequestBody	[IRI] https://mondragon/ids/cd/6321_8081_5022_2401 Payload or request body. The actual data to be sent to the HTTP REST endpoint.	[string] {} < / > 0 text	
[SMC] Response	[IRI] https://mondragon.com/ids/cd/2262_0190_3022_6903 Service response	[-] 3 elements	1..*

Table 2: RestService SubmodelElementCollection - Full attributes specifications

idShort:	Response		
Class:	SubmodelElementCollection		
semanticId:	[IRI] https://mondragon.com/ids/cd/2262_0190_3022_6903		
Parent:	Response		
Explanation:	Service response@en, Respuesta del servicio@es		
[SME type] **idShort**	**semanticId = [idType]value** **Description@en**	**[valueType]** **example**	**card.**
[Property] Code	[IRI] https://mondragon.com/ids/cd/9562_0190_3022_6374 Responde code	[int] 200 201	
[Property] MediaType	[IRI] https://mondragon.com/ids/cd/4572_0190_3022_0456 Media type	[string] application/json application/xml text/xml	
[Property] ExampleValue	[IRI] [IRI]https://mondragon.com/ids/cd/6192_0190_3022_608 An example of response body	[string] {} < / > 0 text	

Table 3: Response SubmodelElementCollection - Full attributes specifications

Once AAS packages are deployed to the AAS Repository, the stored information can be queried through the AAS Repository's API. In matters of this proposal, a business process modeling software has to be modified to consume the API and get the RestServices of all the assets in the network. In this way, the Service-Discovery mechanism can be enabled. Thereby, Camunda Modeler[6] is selected as the business process modeling tool due to its open-source nature and its plugins characteristic. Moreover, Camunda Modeler allows adding new functionalities by the development of separate components (plugins), in such a way that the main code of the modeler tool is never touched [32].

[6] https://github.com/camunda/camunda-modeler

3.2. AAS Web Service Discoverer plugin

In this subsection, a plugin is proposed to enable a Service-Discovery mechanism during the design phase of business processes. For the prototype presented in this paper, the desktop variant of Camunda Modeler is selected because it is standalone and runs on Linux, MacOS, and Windows [33]. The plugin lets the user input initial settings for the modeler to reach the AAS Repository. The settings are saved into the configuration file of Camunda Modeler under the *aasWebServiceDiscoverer* key-value-pair. The plugin's algorithm first identifies what kind of AAS implementation is in the server by using the given settings. Once identified, it makes use of the AAS Repository's API. As of this writing, the proposed plugin can recognize the endpoints offered by Basyx [22] and Admin-shell-io [34]. The endpoints used by the plugin are shown in Table 4.

1) The AAS Repository Interface is used to get the list of registered AAS in the server	
admin-shell-io	/server/listaas
Basyx	/shells
2) The submodel interface is used to get complete details of a given submodel for a given AAS	
admin-shell-io	/aas/{aasId}/submodels/RestServices/complete
Basyx	/shells/{aasId}/aas/submodels/RestServices/submodel/submodelElements

Table 4: AAS Server API endpoints used in this prototype

The algorithm in this plugin iterates each of the AASs found and gets all submodelElements whose submodel idShort is *RestServices*. Each submodelElement represents an individual service of an asset. Thereby, all the services of all the assets can be obtained in JSON format. Finally, the algorithm converts those asset services from JSON to BPMN Service-Tasks and adds them into the modeler's palette in a fashion manner. Consequently, business users can design business processes using the services offered by machines.

4. Prototype Implementation

In this experiment, the feasibility of our modified modeling tool for enabling the Service-Discovery mechanism of the workflow management lifecycle is demonstrated. In order to conduct this experiment, the following tools are installed: 1) AASX Package Explorer 2022-01-13.alpha. 2) AASX Server 2022-01-13.alpha implementation from admin-shell-io, specifically the blazor variant. 3) Camunda Modeler 5.0.0-alpha.1. 4) Camunda Platform. And, 5) the aasx packages given in `https://github.com/MUFacultyOfEngineering/AASBPM/tree/main/aasResources/aasxs`. In addition, two Lego robots are used as assets. For this, Raspberry Pi HAT (Hardware Attached on Top) is installed on both to connect Raspberry Pi with Lego robots.

4.1. Workbench and Lego robots

As seen in Figure 3, the workbench for this experiment is composed of: A) Lego color sorter: A robot that can sort pieces by their color. The sorting is done by moving the feed tray through the conveyor belt to far right and left. The color sensor detects the color of the current piece and the feed motor ejects pieces one by one. B) Lego two axis robotic arm: A robot that can move its arm up, down, left, and right, and can close or open its claw. All these actions are exposed as REST services. C) Non-red pieces container: A container to store non-red pieces. and, D) Red pieces container: A container to store red pieces only.

In order to control the Legos' motors and sensors, a Python and Node-RED based software is built for both robots. This software allows to switch the execution environment between *Digital twins* and *Physical robots*. Thus, robots' functions are exposed as REST services that make OPCUA invocations in behind. Table 5 lists some REST Services offered by the devices.

4.2. BPMN recipe design

Assets and their descriptions are created using AASX Package explorer with the *RestServices* submodel introduced in this paper. The .aasx files are placed into the *..AasxServerBlazor/aasxs* path to allow the AASX Server to read them.

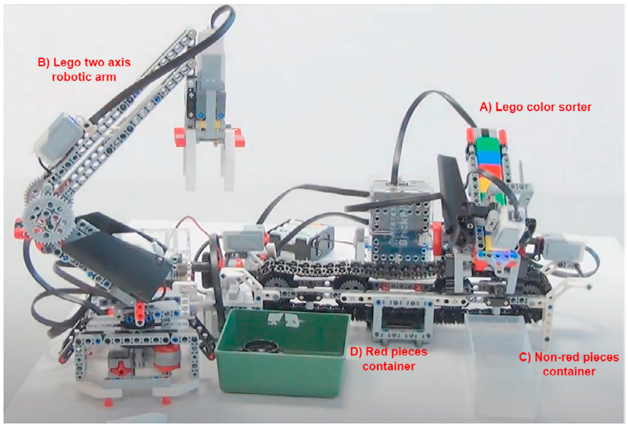
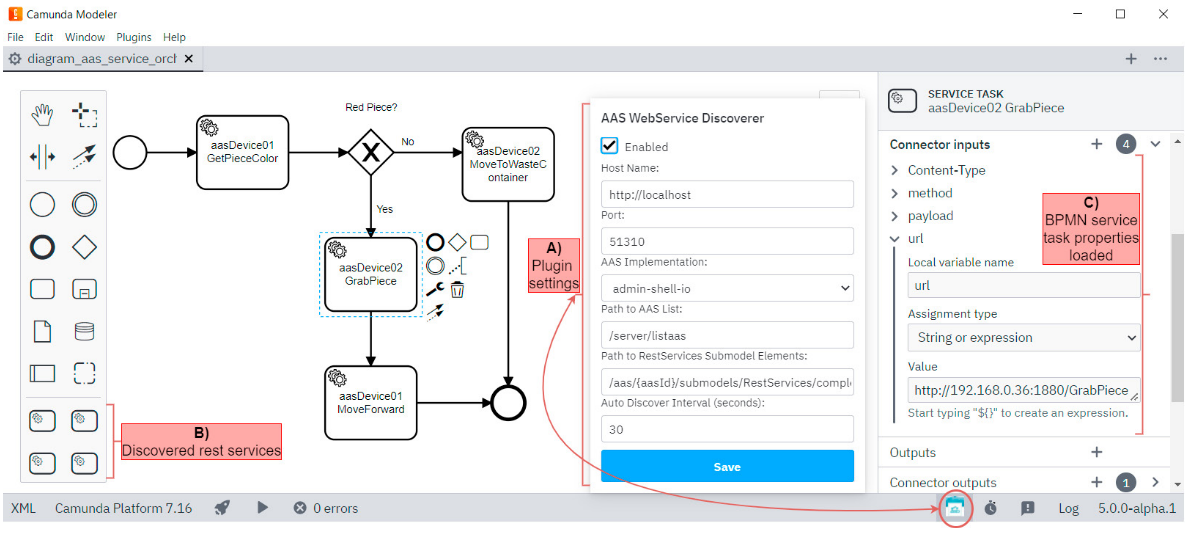


Fig. 3: Workbench and Lego Robots

Lego Robot	EndPoint	Description	Return
	/GetPieceColor	Gets current piece color	String
	/MoveLeft	Moves feed tray though the conveyor to far left	bool
Color Sorter	/MoveRight	Moves feed tray though the conveyor to far right	bool
	/ThrowPiece	Throws the current piece out of the feed tray	bool
	/TurnLeft	Turns the arm 90º to the left	bool
	/TurnRight	Turns the arm 90º to the right	bool
Robotic Arm	/OpenClaw	Fully opens the claw	bool
	/CloseClaw	Closes the claw	bool

Table 5: Endpoints Lego robots

At this point, Camunda Modeler and the *AAS Web Service Discoverer plugin* can be started. Figure 4 shows our plugin in action. A) The user has to click the AAS icon to show-up a modal that lets the user input some settings such as the location of the AAS repository within the network, type of AAS implementation, and EndPoints path to AAS list and SubmodelElements. A *Save* button lets the settings to be saved into the Camunda Modeler's configuration file. B) The plugin will immediately query the AAS repository. The discovered services are displayed at the bottom of the palette within a separate group. C) Once the user drags and drops an asset service into the canvas, a BPMN service-task is drawn-up and all its property values are loaded from the AAS accordingly. In this example, an AAS service has been imported to the BPMN editor (GrabPiece Service-Task). BPMN information is stored in XML format. Figure 5 shows the generated XML for the GrabPiece Service-Task.



AAS WebService Discoverer

[✓] Enabled

Host Name: http://localhost

Port: 51310

AAS Implementation: admin-shell-io

Path to AAS List: /server/listaas

Path to RestServices Submodel Elements: /aas/{aasId}/submodels/RestServices/compl

Auto Discover Interval (seconds): 30

Save

SERVICE TASK
aasDevice02 GrabPiece

Connector inputs 4
- Content-Type
- method
- payload
- url

Local variable name: url

Assignment type: String or expression

Value: http://192.168.0.36:1880/GrabPiece

Start typing "${}" to create an expression.

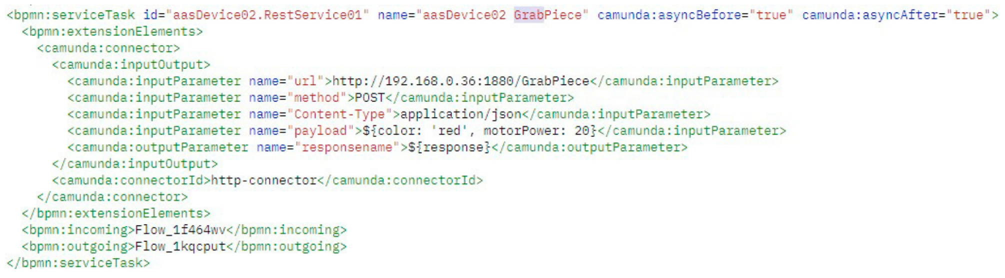
Outputs

Connector outputs 1

Fig. 4: Camunda Modeler - AAS Web Service Discoverer Plugin

4.3. BPMN recipe deployment and execution

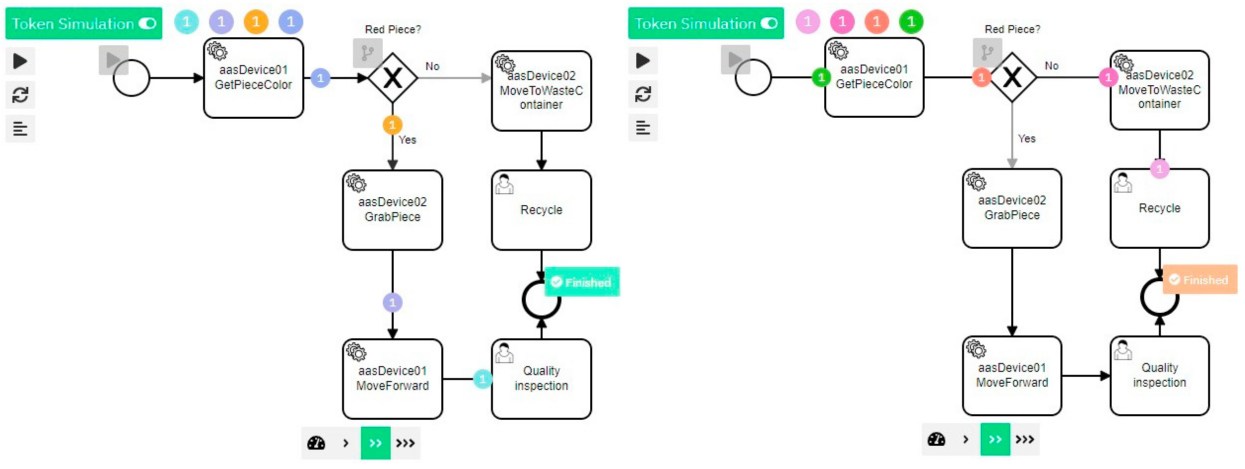
Besides BPMN recipes can be directly deployed into Camunda Platform. We decided first to simulate the execution using the *Token Simulation* plugin for Camunda Modeler in order to anticipate any error that may occur during workflow execution. Figure 6 shows the simulation in both scenarios; when the current piece is red and when the cur-



```
<bpmn:serviceTask id="aasDevice02.RestService01" name="aasDevice02 GrabPiece" camunda:asyncBefore="true" camunda:asyncAfter="true">
  <bpmn:extensionElements>
    <camunda:connector>
      <camunda:inputOutput>
        <camunda:inputParameter name="url">http://192.168.0.36:1880/GrabPiece</camunda:inputParameter>
        <camunda:inputParameter name="method">POST</camunda:inputParameter>
        <camunda:inputParameter name="Content-Type">application/json</camunda:inputParameter>
        <camunda:inputParameter name="payload">${color: 'red', motorPower: 20}</camunda:inputParameter>
        <camunda:outputParameter name="responsename">${response}</camunda:outputParameter>
      </camunda:inputOutput>
      <camunda:connectorId>http-connector</camunda:connectorId>
    </camunda:connector>
  </bpmn:extensionElements>
  <bpmn:incoming>Flow_1f464wv</bpmn:incoming>
  <bpmn:outgoing>Flow_1kqcput</bpmn:outgoing>
</bpmn:serviceTask>
```

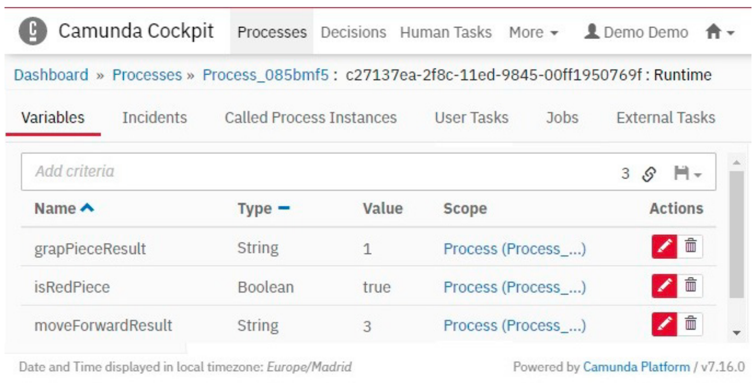
Fig. 5: Camunda Modeler - XML GrabPiece Service-Task

rent piece is other than red. Finally, the BPMN recipe is deployed into Camunda Platform and executed accordingly. Figure 7 shows the process instance in Camunda Cockpit (part of Camunda Platform) and the variables' values for each Service-Task during execution.



(a) Red piece (b) Other than red piece

Fig. 6: BPMN recipe simulation (prior deployment)



Dashboard » Processes » Process_085bmf5 : c27137ea-2f8c-11ed-9845-00ff1950769f : Runtime

Name	Type	Value	Scope	Actions
grapPieceResult	String	1	Process (Process_...)	
isRedPiece	Boolean	true	Process (Process_...)	
moveForwardResult	String	3	Process (Process_...)	

Date and Time displayed in local timezone: *Europe/Madrid*   Powered by Camunda Platform / v7.16.0

Fig. 7: BPMN recipe execution in Camunda Platform

5. Conclusions

This paper introduced an architecture and tools for manufacturing business process management using the Asset Administration Shell concept. This approach aims to allow the design of orchestrations of services provided by devices in a standardized manner. For this, a revision of similar approaches is presented identifying their characteristics and whether they use or not the AAS concept. The links to the source code of both tools, AAS submodel template and Camunda Modeler plugin are provided for community use.

An experimental phase is conducted, showing that this approach can be easily implemented on already existing AAS architectures. As demonstrated during the modeling phase, this solution provides assistance to business analysts for the design of manufacturing business processes by providing assets' services within the modeller's palette. Thereby, it is possible to design orchestration of assets' services in BPMN format. As well, the execution of BPMN recipes showed that the workflow maintains its compatibility with BPMN workflow executor tools as it creates instances and executes each Service-Task accordingly. The limitation of our prototype comes from the fact that our modeling tool employs the AAS repository's API to query assets' services. As of this writing, this prototype supports only the admin-shell-io and Basyx AAS implementations. This is because the structure of the data delivered by the API differs from each AAS implementation vendor.

Our future work includes testing this solution using the edge embedded workflow manager presented in [31] and using a real industrial use case. The advantage of executing workflows at the edge environment is more accessibility to logs, low latency, and scalability [35]. Consequently, we plan to enhance the dynamicity of workflows during execution by enabling context-awareness capabilities in our architecture [36]. Thereby, workflows could change their normal control flow based on context changes. Our idea is that devices' logs could serve as input for a context-analyzer component. Sensors' logs can be gathered in real-time using a publish/subscribe model, which fits the AAS concept [37].

Acknowledgment

This research work has been funded by the European Commission, through the European H2020 research and innovation program, ECSEL Joint Undertaking, and National Funding Authorities from 18 involved countries under the research project Arrowhead Tools with Grant Agreement no. 826452 and the Department of Education, Universities and Research of the Basque Government under the projects Ikerketa Taldeak (Grupo de Sistemas Embebidos).

References

[1] M. Moghaddam, M. N. Cadavid, C. R. Kenley, A. V. Deshmukh, Reference architectures for smart manufacturing: A critical review, Journal of Manufacturing Systems 49 (2018) 215–225. `doi:https://doi.org/10.1016/j.jmsy.2018.10.006`.

[2] C. Richardson, Microservices patterns: with examples in Java, Simon and Schuster, 2018.

[3] F. Pauker, J. Mangler, S. Rinderle-Ma, C. Pollak, Centurio. work-modular secure manufacturing orchestration, 16th International Conference on Business Process Management 2018 (2018).

[4] A. Giret, E. Garcia, V. Botti, An engineering framework for service-oriented intelligent manufacturing systems, Computers in Industry 81 (2016) 116–127, emerging ICT concepts for smart, safe and sustainable industrial systems. `doi:https://doi.org/10.1016/j.compind.2016.02.002`.

[5] ZVEI, The reference architectural model rami 4.0 and the industrie 4.0 component (2022).
URL `https://www.zvei.org/en/subjects/industry-4-0/the-reference-architectural-model-rami-40-and-the-industrie-40-co`

[6] P. I. 4.0, Details of the asset administration shell part 1 (2021).
URL `https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V3.html`

[7] P. I. 4.0, Details of the asset administration shell part 2 (2021).
URL `https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part2_V1.html`

[8] F. Pauker, I. Ayatollahi, B. Kittl, Service orchestration for flexible manufacturing systems using sequential functional charts and opc ua, Dubrovnik 9 (2015) 11–09.

[9] Y. Lu, F. Ju, Smart manufacturing systems based on cyber-physical manufacturing services (cpms), IFAC-PapersOnLine 50 (1) (2017) 15883–15889.

[10] S. Cavalieri, M. G. Salafia, A model for predictive maintenance based on asset administration shell, Sensors 20 (21) (2020) 6028.

[11] Q. Lu, X. Li, G. Li, M. Li, J. Zhou, J. Liu, X. Shen, F. Zhu, A general asset administration shell platform for production line design, in: 2021 IEEE 1st International Conference on Digital Twins and Parallel Intelligence (DTPI), IEEE, 2021, pp. 192–195.

[12] M. Achir, A. Abdelli, L. Mokdad, J. Benothman, Service discovery and selection in iot: A survey and a taxonomy, Journal of Network and Computer Applications 200 (2022) 103331. doi:https://doi.org/10.1016/j.jnca.2021.103331.

[13] S. Ivanov, A. Kalenkova, Comparing process models in the bpmn 2.0 xml format, in: Proceedings of the Spring/Summer Young Researchers' Colloquium on Software Engineering, Vol. 27, 2015, pp. 255–266.

[14] OMG, Business process model and notation (bpmn) (2022).
URL https://www.omg.org/spec/BPMN/2.0/About-BPMN/

[15] D. Hollingsworth, U. Hampshire, Workflow management coalition: The workflow reference model, Document Number TC00-1003 19 (16) (1995) 224, publisher: Citeseer.

[16] Gartner, Gartner Magic Quadrant for Intelligent Business Process Management Suites (2022).
URL https://www.gartner.com/en/documents/3899484

[17] Gartner, Market Guide for Intelligent Business Process Management Suites (2022).
URL https://www.gartner.com/en/documents/3993207-market-guide-for-intelligent-business-process-management

[18] Slintel, Top competitors of workflow automation (2022).
URL https://www.slintel.com/tech/workflow-automation/camunda-market-share

[19] Camunda, Camunda platform 8 – the universal process orchestrator (2022).
URL https://camunda.com/platform/

[20] P. I. 4.0, ZVEI, Reference architectural model industrie 4.0 (rami 4.0) (2022).
URL https://ec.europa.eu/futurium/en/system/files/ged/a2-schweichhart-reference_architectural_model_industrie_4.0_rami_4.0.pdf

[21] M. Hankel, B. Rexroth, The reference architectural model industrie 4.0 (rami 4.0), ZVEI 2 (2) (2015) 4–9.

[22] Basyx, Basyx (2022).
URL https://wiki.eclipse.org/BaSyx

[23] P. I. 4.0, Submodel templates of the asset administration shell - zvei digital nameplate for industrial equipment (2022).
URL https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Submodel_templates-Asset_Administration_Shell-digital_nameplate.html

[24] P. I. 4.0, Submodel templates of the asset administration shell - generic frame for technical data for industrial equipment in manufacturing (2022).
URL https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Submodel_templates-Asset_Administration_Shell-Technical_Data.html

[25] P. I. 4.0, Submodel templates of the asset administration shell - submodel for contact information (2022).
URL https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Specification_Submodel-Templates.html

[26] I. https://industrialdigitaltwin.org/, Repo submodel-templates (2022).
URL https://github.com/admin-shell-io/submodel-templates

[27] S. Schönig, L. Ackermann, S. Jablonski, A. Ermer, An integrated architecture for iot-aware business process execution, in: Enterprise, business-process and information systems modeling, Springer, 2018, pp. 19–34.

[28] I. Graja, S. Kallel, N. Guermouche, A. H. Kacem, Bpmn4cps: A bpmn extension for modeling cyber-physical systems, in: 2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2016, pp. 152–157. doi:10.1109/WETICE.2016.41.

[29] R. Martinho, D. Domingos, F. Martins, C. Cândido, Internet of things aware ws-bpel business processes context variables and expected exceptions, Journal of Universal Computer Science 20 (2014) 1109–1129. doi:10.3217/jucs-020-08-1109.

[30] K. Thramboulidis, D. C. Vachtsevanou, A. Solanos, Cyber-physical microservices: An iot-based framework for manufacturing systems, in: 2018 IEEE Industrial Cyber-Physical Systems (ICPS), IEEE, 2018, pp. 232–239.

[31] F. Larrinaga, W. Ochoa, A. Perez, J. Cuenca, J. Legaristi, M. Illarramendi, Node-red workflow manager for edge service orchestration, in: NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, IEEE, 2022, pp. 1–6.

[32] Camunda, Camunda modeler - plugins documentation (2022).
URL https://docs.camunda.io/docs/components/modeler/desktop-modeler/plugins/

[33] Camunda, Camunda - overview components (2022).
URL https://docs.camunda.io/docs/components/

[34] IDTA, admin-shell-io aasx server (2022).
URL https://github.com/admin-shell-io/aasx-server

[35] S. Hamdan, M. Ayyash, S. Almajali, Edge-computing architectures for internet of things applications: A survey, Sensors 20 (22) (2020). doi:10.3390/s20226441.

[36] A. Carrera-Rivera, F. Larrinaga, G. Lasa, Context-awareness for the design of smart-product service systems: Literature review, Computers in Industry 142 (2022) 103730.

[37] G. Di Orio, P. Maló, J. Barata, Novaas: A reference implementation of industrie4.0 asset administration shell with best-of- breed practices from it engineering, in: IECON'19 - IEEE 45th Annual Conference of the IEEE Industrial Electronics SocietyAt: Lisbon, Portugal, 2019, pp. –. doi:10.1109/IECON.2019.8927081.