embeddedworld2019
Exhibition&Conference
... it's a smarter world

# Dynamic LwM2M Data Model Mapping to OPC UA

Dovydas Girdvainis
Software Solutions
Hahn-Schickard
Villingen-Schwenningen, Germany
Dovydas.Girdvainis@hahn-schickard.com

Christoph Rathfelder
Software Solutions
Hahn-Schickard
Villingen-Schwenningen, Germany
Christoph.Rathfelder@hahn-schickard.com

***Abstract*—the digital transformation in production leads to a growing number of sensor systems integrated into production facilities. In addition to the physical integration these systems also need to be integrated into the IT backend. OPC UA is one standard promising to close the gap between shop floor and IT backend. However implementing an interface on a custom sensor with limited memory can be quite challenging, especially so when dealing with devices that use wireless communication. To solve this problem we propose a solution based on Lightweight Machine to Machine (LwM2M) and OPC UA technologies. In this paper, we present our approach showing how LwM2M communications protocol works in conjunction with OPC UA technology. We briefly discuss the advantages of LwM2M in the context of smart automation and Industry 4.0 before explaining how sensor information is translated into OPC UA compliant data. Finally, we give some insights in our implementation and the various technologies we used to achieve this task.**

***Keywords—LwM2M; OPC UA; automation; smart sensors; Industry 4.0***

## I. MOTIVATION

Integrating smart sensor systems into production facilities is the basis of Industry 4.0, nonetheless it is not a straightforward task, due to the vast amount of choices between manufactures and their offered products. One of the possible solutions for this problem is to use a standard communication protocol like the widespread OPC UA (Open Platform Communications Unified Architecture) [1]. However, implementing an OPC UA interface can be impossible on embedded systems with resource constraints, thus requiring the use of a lightweight communication model integrated into the OPC UA Server.

In our approach, we achieve this task by using the LwM2M (Lightweight Machine to Machine) protocol to facilitate communications between the OPC UA Server and one or more sensor endpoints. To benefit from the OPC UA information model, which represents clients within OPC UA *Address Space*, we developed an automated data mapping technology from LwM2M data descriptions to the OPC UA information model.

With our approach, the user is able to integrate sensor systems, based on LwM2M communication, into the OPC UA Server by using IPSO (IP for Smart Objects) definition files, which are automatically mapped to the OPC UA information model. Furthermore, the user is able to extend these definition files with his own descriptions, by following the IPSO Work Group's defined standard and adding them into our generic OPC UA Server's object definitions directory.

## II. FOUNDATIONS

This chapter provides an overview on the technologies, we use in our approach covering OPC UA as communications protocol, LwM2M as device management protocol and IPSO as information model. We discuss the basics of each technology that build the basis for our approach.

### A. Open Platform Communications Unified Architecture

OPC UA is a platform independent communication protocol primarily aimed at industrial applications. Its main goal is to achieve interoperability between multiple vendors and their products as well as to close the gap between the shop floor and the backend IT systems. This technology is predominantly based on the *Client-Server* model, but it recently included a specification for the *Publish-Subscribe* model as well. The standard consists of nineteen specifications, which cover security, information model, historical data access, *Publish-Subscrib*e communications model and others [2]. The roots of these specifications can be found in its predicator - OLE (object linking and embedding) for Process Control, which is now known as OPC Classic.

OPC UA allows the Client to interact with the Server in four different ways, called *Data Access, Alarms and Conditions, Programs* and *Historical Access*. Each way deals with different Server functionalities and each is commonly described as an access type. *Data Access (DA)* deals with the representation and the use of automation data [3]. *Alarms and Conditions (A&E)* deals with state representations of a system or its components [4]. *Program* deals with complex and state-full functionalities of the system [5]. Finally, *Historical Data Access (HDA)* deals with storing and viewing of system's historical events [6].

Most of these access types have a specialized way of representing information to the Client, which is called a *View*. A *View* is a specific subset of information the OPC UA Server provides to the Client. The total sum of this information is known as the Server's *Address Space* [7]. Information within the *Address Space* is represented in a hierarchical structure of objects called *Nodes*. An example of such hierarchical structure could be an *Object Node*, which is further described by various

different *Variable Nodes*, which intern are linked by *References* to the original *Object Node*.

Every *Node* within OPC UA must consist of *Attributes*, and may have *References* to other *Nodes*. An Attribute is an elementary data element that describes a specific part of the OPC UA *Node*, for example the *Node ID Attribute*. Some *Attributes* are mandatory for *Node* instantiation, while others are optional. A Client can access these *Attributes* via specialized *Read/Write/Query/Subscribe* services provided by the Server [8].

OPC UA *Services* are grouped into nine *Service Sets*:

*1)* The *Discovery Service Set*, which defines the Services that allow a Client to discover the Endpoints implemented within the Server.

*2)* The *Secure Channel Service Set*, which defines the Services that allow for secure communication between the Client and the Server.

*3)* The *Session Service Set*, which defines the Services that allow the Client to manage and authenticate Sessions.

*4)* The *Node Management Service Set*, which defines the Services that allow the Client to add, modify and delete various Nodes within Server's Address Space.

*5)* The *View Service Set*, which defines Services that allow the Client to utilize various Views of Servers Address Space.

*6)* The *Attribute Service Set*, which defines Services that allow the Client to read and write into Node Attributes.

*7)* The *Method Service Set*, which defines Services that allow the Client to call Methods.

*8)* The *Monitored Item Service Set*, which defines services that allow the Client to create, modify and delete Monitored Items, which in term are used to monitor Attributes.

*9)* The *Subscription Service Set*, which defines Services that allow the Client to create, modify and delete Subscriptions for the Monitored Items.

In short – OPC UA is a service oriented and platform independent communication protocol aimed at monitoring and controlling industrial automation process. It seamlessly integrates products from various vendors and provides standardized ways of representing the information to the Client.

### *B. Lightweight Machine-to-Machine (LwM2M)*

LwM2M is a device management protocol based around the Application Layer within the Open Systems Interconnect (OSI) model. It specializes in managing lightweight low power devices over a variety of network types. The standard is structured around a *Client-Server* model and defines four interfaces for the *Bootstrap*, *Client Registration*, *Device Management* and *Information Reportin*g processes as well as an internal resource model. Each interface may consist of *Downlink* (from Server to Client) and *Uplink* (from Client to Server) operations [9].

*Bootstrap interface* defines operations that provide essential information to enable LwM2M Client to execute the *Register* operation. This interface supports four different modes – factory bootstrap, bootstrap from Smartcard, client initiated bootstrap and Server initiated bootstrap.

*Client Registration interface* describes the available operations for the client registration procedure. This interface consists of three operations – *Register*, *Update* and *Deregister*. All of these operations are fully *uplink* in regards to the LwM2M Server

*Device management and service enablement interface* describes the available operations for the standard device management. It is used to access and manage *Object Instances* and their *Resources*. This interface consists of seven operations: *Read*, *Write*, *Execute*, *Create*, *Delete*, *Write attribute*, and *Discover*. All of these operations are *downlink* in regards to the LwM2M Server

*Information reporting interface* describes the available operations for the data observation. It is used to notify the *Observers* when new values are available for processing. This interface consists of three operations: *Observe*, *Cancel Observe*, and *Notify*.

All of the mentioned operations are mapped to the Constrained Application Protocol (CoAP) bindings within the LwM2M Server implementation.

LwM2M declares a simple resource model, which defines a LwM2M Client as a set of *Objects* and *Resources*. Each LwM2M *Object* defines a vector of *Resources* and has a unique *Object identifier* assigned by Open Mobile Alliance Naming Authority (OMANA). For a Client to use a specific LwM2M *Object*, it first must be instantiated with the *Resource* defined by OMANA and assigned an instance id. Each LwM2M *Resource* may contain a value, if it is described as *Readable*/*Writable*, or contain a link to an action, if it is described as *Executable*. *Resources* can be declared as *Mandatory* or *Optional*. If a *Resource* is defined as *Mandatory* within the LwM2M *Object* then it must be instantiated by the client for the LwM2M *Object* instantiation to succeed, while *Optional Resources* may be omitted by the *Object* instantiation procedure.

To summarize LwM2M is application centered device management protocol, which defines four common application interfaces and a common resource model, which extends CoAP technology.

### *C. Internet Protocol for Smart Objects (IPSO)*

IPSO is an information model based on OMA LwM2M specification. It declared the *Object Representation*, available *Data Types* and *Operations* as well as supported *Content Formats* [10].

Each *Object* is mapped into the Uniform Resource Identifier (URI) path as declared by OMA LwM2M *Resource Model*. This URI represents the *Object ID*, *Object Instance ID* and the *Resource Type ID*. Each field is an unsigned 16-bit value. This template follows the Web Linking [11] and CoRE Link Formats [12], thus allowing for simplistic API designs.

IPSO declares seven *Data Types*:

*1)* A *String*, which is defined as a standard UTF-8 string type.

*2)* An *Integer*, which can be a 64/32/16 or 8 bit signed integer type.

*3)* A *Float*, which can be a 64 or 32 bit floating point value.

*4)* A *Boolean*, which is defined as an integer with the value of 0 or 1.

*5)* An *Opaque*, which is defined as a sequence of binary octets.

*6) Time*, which is defined as a signed integer, representing the number of seconds since Jan 1st 1970, UTC0.

*7)* An *Object Link*, which is defined as two 16 bit long unsigned integers in Network Byte Order (NBO), referring to an instance of another Object.

This information model implements operation types declared in OMA LwM2M specification. These operations are split into four groups - *Resource operations*, *Object instance operations*, *Object operations*, and *Attribute operations*. *Resource operations* declare the available operation per *Resource*. Each *Resource* must have at least one *Read*, *Write* or *Execute* operation and it may have both *Read* and *Write* or *Execute* operations at the same time. These operations are restricted by *Access Type* field within the *Resource* declaration. *Object instance operations* allow for *Create* and *Delete* of multiple instances of the same *Object type*, if the *Multiple Instances* field is set to *true*. *Object operations* are similar to *Resource operations* except they lack the *Execute* operation type. *Attribute operations* allow to *Set* and *Discover* LwM2M *Attributes*.

In brief, IPSO is a standard, which further defines the LwM2M Resource Model to improve interoperability of various LwM2M devices and applications.

## III. Approach

As already mentioned, our LwM2M *Adapter* acts as an enabler for the OPC UA Server and provides real data for the *Node* abstraction, while the OPC UA Server handles the hierarchy of *Device Nodes* and the communication between the OPC UA Clients. “Fig. 1.” shows an overview of our approach for the OPC UA Server. Within this figure, we see three major parts of an OPC UA *Client-Server* model: *OPC UA viewer application* (1), which displays the content of OPC UA *Address Space* within the OPC UA *Server* (2) and the *Database* (3). We structured our OPC UA *Server* (2) into five modules: *OPC UA Server implementation (2.1)*, *Technology adapter manager (2.2)*, *Technology adapter implementation (2.3)*, *Historizer (2.4)* and the *Utilities module (2.5)*. The Server communicates with *OPC UA viewer application* (1) via the *OPC UA Service Set*, which enables the user to access the endpoints and Server nodes, as well as historical values from the *Database* (3) via the *Historizer* module (2.4).

*OPC UA Server implementation (2.1)* handles the OPC UA *Address space*, *Services* and *Views*, while the *Historizer (2.4)* is responsible for writing data values from endpoint nodes into the database as described by the OPC UA *Historical Access*.

Our *Technology adapter manager* controls the communication between *OPC UA Server* and all of the managed *Technology Adapters*, as well as the initialization of supported *Technology Adapters*. A *Technology Adapter* is an abstraction of different communication protocols, in this case,

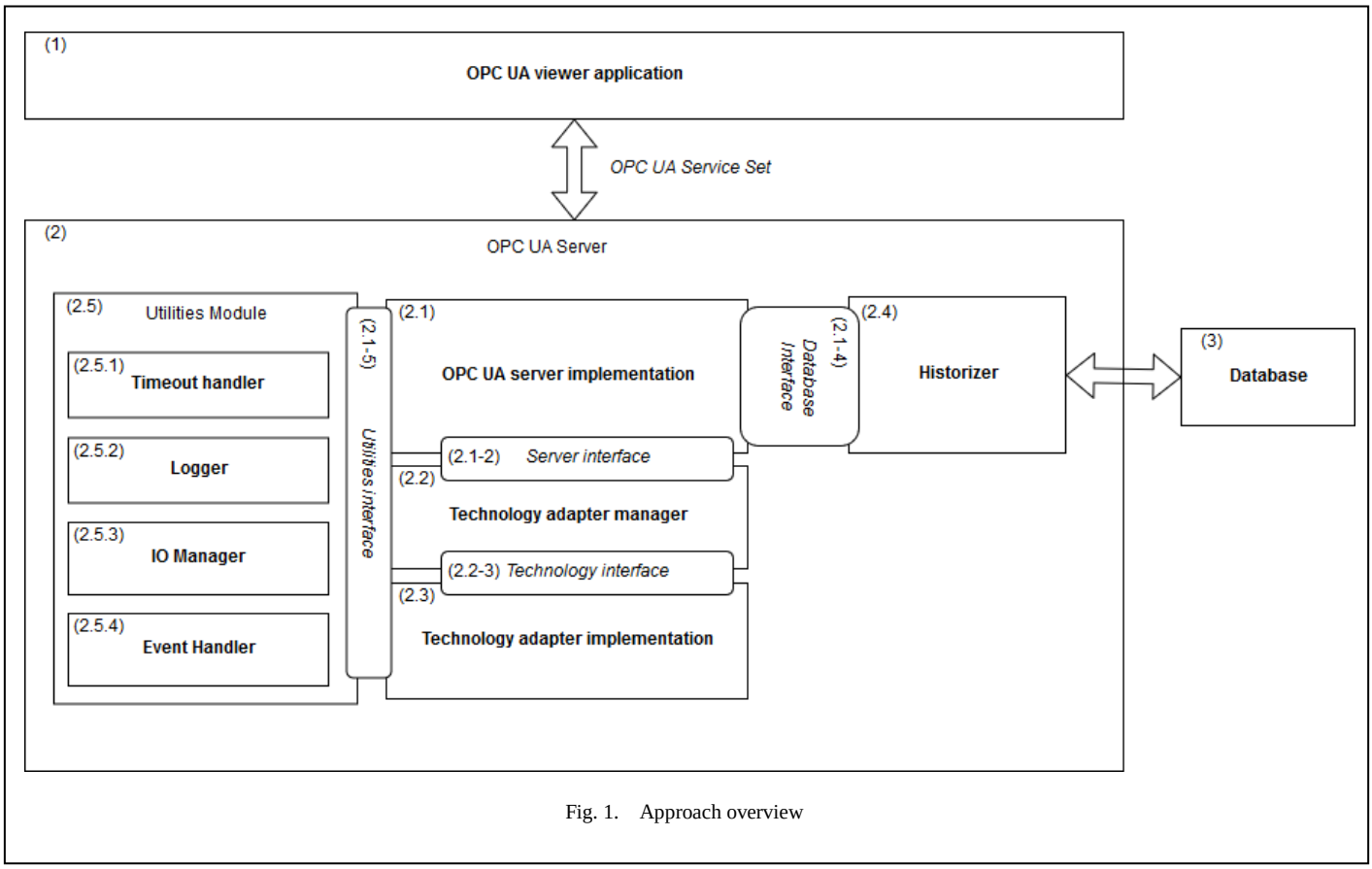


Fig. 1. Approach overview

it is the LwM2M communication protocol, and hence it is called the *LwM2M Adapter*.

Finally, our *Utilities Module* enables event driven communication between all other modules and allows *I/O* between the *file system* and the Server, to process Server configuration, technology adapter's files and event logging.

In this chapter, we focus on the conversion from LwM2M information model to the OPC UA *Address Space*, thus covering the device registration interaction, during which such mapping occurs.

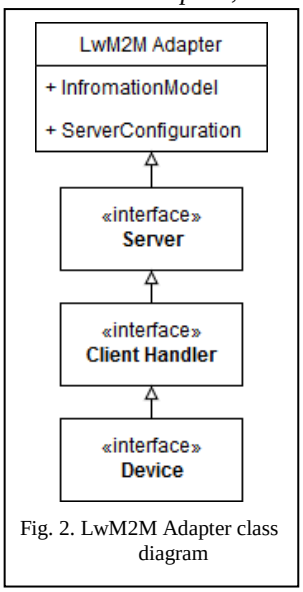

Fig. 2. LwM2M Adapter class diagram

### A. *Approach Overview.*

In our approach, *Technology Adapter* is realized as a *LwM2M Adapter*. It holds the *IPSO Information Model*, which is described in more detail in the following section, and the *Server Configuration*, which specifies LwM2M Server parameters, like the port number and the IPv6 prefix. Our *Technology Adapter* class diagram can be seen in "Fig. 2.", from which it is clear, that the adapter depends on the *Server interface*, which is responsible for managing incoming and outgoing LwM2M events and commands. *Server interface* internally depends on the *Client Handler interface* that handles various LwM2M events generated by the LwM2M Client, and keeps track of all know LwM2M Device within the network. These LwM2M Devices are described by the *Device interface*, which holds information like device name, device ID and IPSO Object dictionary.

"Fig. 3." depicts an interaction diagram illustrating the actions when a new device connects to the network. Once a device is connected to the LwM2M Server (1), a LwM2M event (2) will be triggered in the *Client Handler*, which will start the processing of newly found LwM2M Device. This reads information, like the *device name*, *ID*, then checks if this data is valid (4). After that, *IPSO* data will be read from the registration request. IPSO *Object* description (7) is then read and checked against the currently loaded IPSO Information Model (8). If the object is malformed or unsupported by the IPSO Information Model, it is discarded and processing of the next object is started. If the IPSO *Object* is supported, related *Resources* (10) are processed and checked in a similar manner (11) as previously. Once all of the *Objects* have been checked and created the system creates an appropriate device (16) from all of the previously processed information. A notification (17) to the *Technology adapter manager* is sent with a valid pointer to this device, if it was processed correctly or an *Invalid Device* event, if information has been malformed or unsupported. This notification triggers OPC UA Server to start building the *Device Node* (1).

Our *Device Node* is a complex *Node* made out of many sub *Nodes*, linked by *References* (21). To simplify terminology we call these *Sub Nodes* - *Children* of the *Device Node*. These *Children* are complex *Nodes* in the case of IPSO *Objects*, and simple *Nodes* in the case of IPSO *Resources*. We differentiate these types by calling them *Object Node* (19) and *Resource Node* (20).

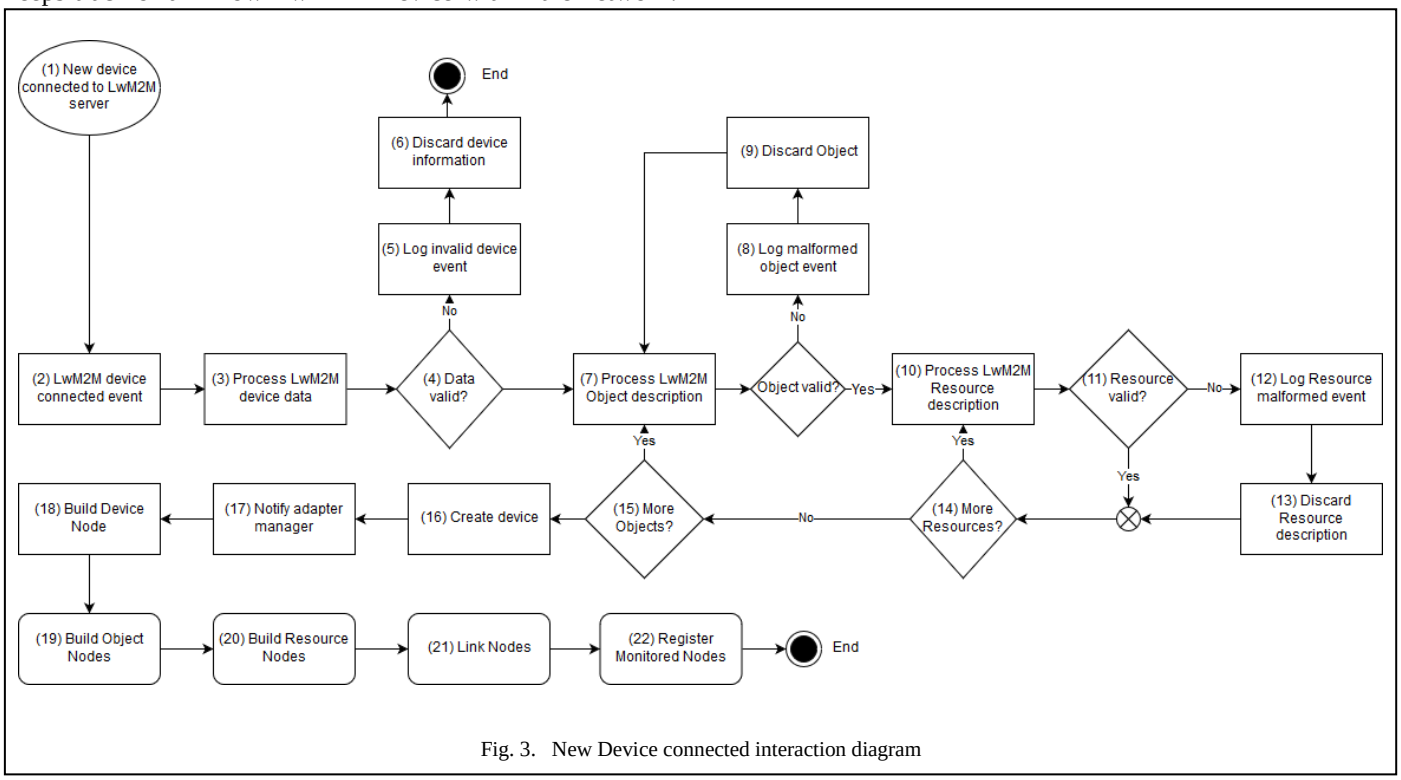

Fig. 3. New Device connected interaction diagram

Typically, *Resource Nodes* (20) are the only *Node* types that have real data. However as described in the following section, we allow specific *Object Nodes* (19) to contain it as well. This property is identified by the IPSO description, and specifies whether the *Node* can be monitored for value changes. Once all of the *Children Nodes* are linked (21) to the *Device Node*,the system checks, which of *Nodes* have the *Observable* property and be registers them for monitoring (22).

### B. Dynamic IPSO Information Model.

Before LwM2M Clients can register, our LwM2M Server implementation requires the IPSO descriptions library to be loaded and processed into the Server. A specialized parser converts the description files into description objects, which will create an internal IPSO descriptions dictionary. As previously described, this dictionary will be used during the LwM2M processing (3). The IPSO descriptions library consists of multiple *.xml* files, each describing LwM2M *Object* and its *Resources*, these files are dynamically loaded at LwM2M sever startup.

### C. Static and Dynamic Fields.

During our development phase, we identified that certain LwM2M *Resources* never change throughout the lifetime of the Client. These *Resources* typically describe static information of the *Object* like the IPSO-Alliance's "Units" *Resource*. This *Resource* describes what measurement units, this object uses. In majority of cases this resource will never change, hence we do not need to monitor, nor save it within our historical database. So in turn, we call this *Resource Static* and our OPC UA Server does not need to provide any monitoring or historization services to it. To simplify the identification of such *Resources* we extended the IPSO Alliance's *Resource* definition with a new field called "*Dynamic*". This allows the OPC UA Server to only assign *Read*/*Write* services to *Static Resources* and pay closer attention to *Dynamic* ones.

Similar to the "*Dynamic*" field for *Resource* definitions, we found that it is quite convenient to package all of the *Dynamic Resources* of the Device and pack it into a *Dynamic Object Instance*. This *Object* can be identified by another new "*Dynamic*" field within the IPSO *Object* definition. Since we know, that all of the *Resources* within this *Object* are *Dynamic*, we can skip the "*Dynamic*" field definitions for each of the *Resources* and assign a single monitoring to the entire *Object*. This approach further optimizes IPSO Description processing and Value observation on the OPC UA Server.

### D. Event Mechanism.

As illustrated in "Fig .1." our OPC UA Server uses events to facilitate various requests from the OPC UA and the LwM2M Clients alike. To identify events, we separate them into two types: LwM2M events and OPC UA events. LwM2M events are events coming from the LwM2M device and to OPC UA Server, this would include "New device connected" event, previously discussed and shown in "Fig. 3.", as well as "Device disconnected" and "Device value changed" events. OPC UA events are events coming from the OPC UA Client to the OPC UA Server, this would include such events as "Read Node Value", "Write Node Value", "Execute Node Command" and "Read Node's Historical Value".

## IV. Implementation

As shown in "Fig. 4." our implementation consists of five technology stacks that correspond to OPC UA Server packages depicted in "Fig. 1.". These technology stacks are: the Automation Service Next Generation (ASNeG) [13] OPC UA Stack (1), ASNeG OPC UA Database (2), IPSO Parser (3), LwM2M interface (4) and the Eclipse Wakaama [14] library (5).

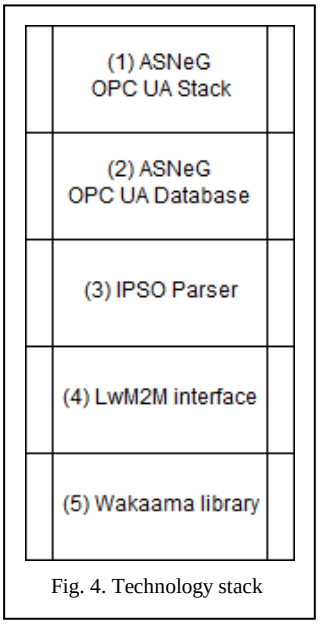

Fig. 4. Technology stack

We use the ASNeG OPC UA Stack open source framework for our *OPC UA Server implementation* (2.10 from "Fig. 1.". This framework provides a Software Development Kit (SDK) for Client\Server applications, which can be run as dynamic libraries and a Server information model with configuration settings in an XML format, thus allowing the user to configure the OPC UA Server to suit the needs of their applications.

Our *Historizer* (2.4) implementation from "Fig. 1" is based on an additional ASNeG's open source project, the OPC UA Database (2) that provides the interface to Open Database Connectivity (ODBC) and MySQL database.

Our *Technology adapter manage*r (2.2) from "Fig. 1.", is implemented within the IPSO Parser (3), which we developed as our initial adapter manager. As discussed when presenting our approach, it controls the dynamic loading of IPSO description files, LwM2M device conversion to OPC UA *Address Space* and the *Event Hander*.

A part of our Technology adapter implementation (2.3) from "Fig 1." (mainly the interface between the Technology adapter manager and the adapter itself), is implemented in the LwM2M interface (4), which contains LwM2M device descriptions. Finally, the rest of our Technology adapter implementation (2.3) from "Fig. 1." is based on an open source implementation of OMA's LwM2M protocol from Eclipse called Wakaama [14]. This library allows us to build our LwM2M Server as a dynamic library. The full implementation of our approach can be found in our open source project the "Non-disruptive Kit for the Evaluation of Industry 4.0" (NIKI 4.0) [15].

## V. Conclusions

In conclusion, our approach allows the user to work with LwM2M devices as OPC UA *Nodes* within the OPC UA Server, by mapping the LwM2M *Objects* and *Resources* to the OPC UA *Nodes* with the help of dynamically loaded IPSO *Information Model*. This *Information Model* can be adjusted to suit the needs of the application as well as allow the user to expand it with new

sensor systems based on the LwM2M technology. As seen from our approach, if the sensor confirms to LwM2M *Information Model*, it will be processed and converted into appropriate OPC UA *Nodes*, which can be accessed and controlled via OPC UA Client applications.

## VI. Acknowledgment

This work has been partially funded by the Baden-Württemberg Stiftung as part of the NIKI 4.0 project and the Bundesministerium für Bildung und Forschung – BMBF as part of the projects Parsifal 4.0 and Ameli 4.0.

## VII. References

[1] U. Steinkrauss, „Overview: OPC Unified Architecture Technical overview and short description,“ 2010.

[2] OPC FOUNDATION®, „OPC Unified Architecture, Part 1: Overview and Concepts“.

[3] OPC FOUNDATION®, „OPC Unified Architecture, Part 8: Data Access,“ 2017.

[4] OPC FOUNDATION®, „OPC Unified Architecture, Part 9: Alarms and Conditions,“ 2017.

[5] OPC FOUNDATION®, „OPC Unified Architecture, Part 10: Programs,“ 2017.

[6] OPC FOUNDATION®, „OPC Unified Architecture, Part 11: Historical Access,“ 2017.

[7] OPC FOUNDATION®, "OPC Unified Architecture, Part 3: Address Space Model," 2017.

[8] OPC FOUNDATION®, „OPC Unified Architecture, Part 4: Services,“ 2017.

[9] Open Mobile Aliance, "Lightweight Machine to Machine Technical Specification," 2017.

[10] OPA IPSO WorkGroup, „https://github.com/IPSO-Alliance/pub,“ [Online]. Available: https://github.com/IPSO-Alliance/pub. [Zugriff am 4 December 2018].

[11] M. Nottingham, „Web Linking,“ 2010.

[12] Z. Shelby, „Constrained RESTful Environments (CoRE) Link Format,“ 2012.

[13] ASNeG, „github.cim,“ github, [Online]. Available: https://github.com/ASNeG/OpcUaStack. [Zugriff am 14 Januar 2019].

[14] Eclipse wakaama, „github.com,“ github, [Online]. Available: https://github.com/eclipse/wakaama. [Zugriff am 14 Januar 2019].

[15] C. B. A. B. D. D. A. G. N. H. K. A. N. C. R. M. S. Sascha Alpers, „Nicht-disruptives Kit für die Evaluation von Industrie 4.0,“ 2017.