iMMAS an Industrial Meta-Model for Automation System Using OPC UA

Jose Miguel Gutierrez-Guerrero[1], Juan Antonio Holgado-Terriza[1]
[1]Software Engineering Department, University of Granada, C/ Periodista Daniel Saucedo Aranda s/n. 18071 Granada, Spain
jmgutierrez@ugr.es

***Abstract*—Industry 4.0 promotes the integration of IT software technology with the industrial devices of a factory for the supervision and control of Industrial Automation Systems (IAS). In this paper, a novel metamodel called iMMAS (Industrial Meta-Model for Automation System) is proposed to simplify the development and deployment of an IAS. The metamodel provides a language for modelling the industrial process of a factory through industrial devices (including PLC, Soft-PLC or RTU) using new abstractions to conceptualize the IAS. This work presents how the models are performed and executed on an OPC UA environment, and how they can be used to generate a program for a PLC. A case study is also shown to test the applicability of the proposal presented.**

***Index Terms*—OPC unified architecture; industrial metamodel; software engineering; human machine interface (HMI); mobile device; supervision control and data acquisition (SCADA); methodology.**

I. INTRODUCTION

Software development is an extensive area generally related with information technology and computing science, focused on the methodologies, software infrastructures, techniques and tools that can assist the development of complex and large software projects. In industrial environments, the traditional techniques employed for the management of software development and deployment are not being systematically used for several reasons. Firstly, industrial systems, in practice, may not include software or its use might be limited, due to the investment in terms of money and time, especially in very simple productive processes where it does not need precise control by software; e.g., although a simple pneumatic system can have just one button to start and another one to stop, it can include also a software module for monitoring the operation of the compressor in a smart display. Secondly, many software tools have to be managed individually with a limited integration among them. And lastly, the traditional programming tools for industry do not have support for software engineering techniques [1].

In an Industrial Automation System (IAS), the main part of the developed software is largely based on Programmable Logic Controllers (PLC) [2]. The software for these devices is based on the development of sequential programs and, in some cases, structured programming using structures such as function blocks. In this scenario, some standards were developed to standardize PLCs in industrial systems as well as the programming languages that could be used, such as IEC 61131-3 [3].

But this standard is only focused on PLC programming and it does not contemplate how industrial systems should be organized regarding other levels. Many well-established paradigms of IT such as the Object-Oriented Programming (OOP) and the Development of Model Driven Software (MDD) are not sufficiently exploited in industrial systems. For instance, the IEC 61499 standard provides support for OOP [4], although, despite being 10 years in the market now, it has not been fully adopted by the industry [5]. The application of MDD and SOA paradigms is more limited and, therefore, it is not yet clear how they must be adopted [6].

On the other hand, software requirements in current industrial systems are becoming a major need. In fact, in the last decade, the ratio of software development with respect to the costs of machinery has doubled from 20 % to 40 % and this trend continues to increase progressively [6]. Moreover, the industrial companies have to adapt quickly to the market developing flexible, scalable systems easy to maintain for reducing the costs of the production process and time-to-market, improving the product quality, reducing energy consumption and contributing to environment sustainability.

These challenges are very difficult to confront with traditional approaches. The use of software engineering techniques is necessary in order to provide systematic methods to design and develop industrial processes, and, furthermore, to determine how they are deployed on the different computing devices involved, such as the PLC, the Machine Interface (HMI) systems (or Supervisory Control And Data Acquisition (SCADA)), and the intermediate systems between PLC and HMI/SCADA.

Model Driven Engineering (MDE) is a software engineering paradigm that promotes the creation of conceptual models procuring an abstract representation of the requirements, functionalities and any other topic related to a specific problem to be solved by a software system. Then, from these models and the application of a systematic methodology, the engineer can generate an implementation of the system that is correct-by-construction.

The supervision of an industrial process requires the

Manuscript received 2 November, 2016; accepted 23 March, 2017.

monitoring of thousands of signals, each one in charge of some specific aspects. Thus, a modelling of industrial processes from the requirements can help engineers to organize the software from both structural and behavioural views. MDE can give an additional abstraction layer, generalizing the common patterns and elements that govern a specific industrial process into a meta-model. As a result, the meta-model provides a common language to express the industrial elements participating in the industrial process, including the behaviour of these elements.

Accordingly, MDE contributes to the definition of new concepts to the Industry 4.0 [7], [8], where a smart factory can help to achieve integrated and easily reconfigurable industrial systems covering the production demands, according to the needs of the increasingly changing market. In this sense, all the elements of our IAS are based on a model that can be improved by the introduction of behaviour modelling and transformation rules [9].

In this paper, a novel meta-model called iMMAS (Industrial Meta-Model for Automation System) is proposed to simplify the development and deployment of an IAS. The meta-model provides a language with a concrete syntax and specific semantic for modelling those industrial devices (including PLC, Soft-PLC or RTU) using new abstractions to conceptualize the IAS. Thereby, the meta-model gives a set of rules and a method to assist the engineer/developer in the modelling process.

Besides, models carried out with iMAAS can be run in an executable environment at runtime inside the scope of OPC UA technology. That is, an OPC UA client from a third-party can access to the OPC UA server and can manage directly the iMMAS models, disengaging the industrial processes and hiding the industrial devices of the factory plant.

Finally, iMMAS can address the software development process of industrial devices from iMMAS models according to a model-driven approach. In this way, the development of an IAS is performed by using high-level abstractions instead of using low-level abstractions at PLC level.

In section II, a contextualization of different standard and software engineering techniques used within the industrial system is summarized and the current situation of relevant standards related with industrial systems is revised. Section III presents iMMAS focusing on the two first levels of the meta-model, and the general rules to deploy a model on OPC-UA server and to generate a PLC program. Section IV explains how to create a specific model based on iMMAS, and how to deploy it in a software control system to manage a modelled climate room as an example. Finally, the last section covers the conclusions and future works.

II. BACKGROUND

The industrial community has developed several standards during the last years to regulate the process of software development for specific industrial devices, integrating the software of different machines placed in a factory plant.

For example, IEC 61131-3 standard is oriented to the programming of PLC devices [4], and its last version (third edition) includes a specific extension to support object-oriented programming [10]. However, ISA-88 standard [6] is focused on issues related with system configuration and system integration, specifically for process control systems on batch, although it can also be applied to discrete, continuous, hybrid, and storage process control [11]. In this sense, the ISA-95 standard (standardized internationally as IEC 62264 [12]) delves into how the integration between IAS systems and business systems like ERP (Enterprise Resource Planning) and MES (Manufacturing Executing System) can be done, defining the interface between control functions/systems and other enterprise functions/systems.

A software developer on IAS will face a laborious work related with integration and deployment issues between systems [13], horizontal (e.g., communication HMI to HMI) and vertical (e.g. communication BATCH systems and MES system). In addition, the software developer has to develop the IAS system; that is, programming PLC devices and HMI systems (usually SCADA systems), and establishing the communication between both systems [14].

Unfortunately, the aforementioned standards provide only general rules for guiding and they do not include any software development methodology [9]. Conversely, MDE has been successfully applied to the development of IT systems [15]. It provides a methodology driven by models, structuring the development of a complex system in separate abstraction levels, from conceptual high-level models to implementation low-level models. Then, the software development process implies a transformation between models until a low level model, the program, is constructed. The separation of the software development at two levels, the design of high-level conceptual models to conceptualize abstractions, and the development based on the transformation between models on different domains, give the essential pillars of MDE [16].

The MDE application to IAS can be similar to IT systems [10]. The industrial systems can be seen as models at different levels: at PLC programming domain, where the signals are captured to reflect the physical environment; at OPC domain, where the signals are captured as data to be exposed to the rest of industrial systems; and, finally, at a MES domain, where the industrial process is governed. Then, it is potentially possible to perform models at different domain levels, and the definition of rules to perform transformations between domains.

The Object Management Group (OMG) [17] is a non-profit organization especially oriented towards business information technology, who defines and guides the application of MDE paradigm to IT systems. In fact, they defined a number of standards for MDE, in particular model-driven architecture (MDA), based on MetaObject Facility (MOF) [18] and Unified Modelling Language (UML) [19]. UML provides a number of diagram representations to capture the requirements, structure, behaviour and even the refinements to executable code, although it does not define the methodology to be applied to develop the software. System Modeling Language (SysML) [20] is a UML derivative for system engineering applications that is getting increasingly popular; e.g. in Manufacturing Machinery [21]. In particular, SysML supports specific diagram representations such design phases for capturing and formalizing the requirements.

There are several works related with the using of UML in software engineering [6]. On one hand, Papakonstantinou *et al*. proposed a way to generate IEC 61499 function blocks from UML diagrams [22]. On the other hand, Thramboulidis described a UML-FB architecture for generating UML diagrams to build Function Blocks [5]. Moreover, Fan *et al*. developed an extension to UML for industrial system called UML-PA (UML for Process Automation) [23].

The use of software methodology to IAS domain could be very beneficial and could reduce the time of design and developing industrial system. Unfortunately, they are not currently specific tools adapted to IAS [6].

Although the application of MDE paradigm and software engineering were limited, the manufacturers of industrial devices can develop some specific software tools to resolve specific issues such as the PLC programming, the definition of simple data models of PLCs, or the configuration and integration of industrial devices in SCADA. In industrial systems, OPC (Ole for process control) [24] systems provide a common way to decouple the control domain from the instrument domain, providing a communication interface to standardize the data exchange with industrial devices using a data access model. Recently, the OPC Foundation [25] has released OPC UA [26], a standard that has native support for modelling and deployment of models directly, opening the possibility of applying MDE paradigm for IAS in order to develop software driven by models. This paper shows a proposal of how OPC UA [27] can exploit MDE principles to develop industrial systems.

III. INDUSTRIAL META MODEL FOR AUTOMATION SYSTEM (iMMAS)

iMMAS (Industrial meta model for automation systems) is a specific meta-model conceived for the development and deployment of an industrial automation system based on a model-driven approach. The meta-model of iMMAS provides a common language with a concrete syntax and specific semantic, which is adapted from the information model of OPC UA, in order to model any IAS.

The modelling process defined in iMMAS provides a systematic method for characterizing the industrial elements that are involved in an industrial process. The models can be executed directly on OPC UA environments using client-server architecture, facilitating in this way its integration with other industrial systems. The models are stored persistently in an OPC UA server using the address model of OPC UA and data are collected from industrial hardware devices (e.g., PLC). All models and data are kept on an OPC UA server, which is accessible for any integrated system that acts as an OPC UA client. Furthermore, models created from iMMAS metamodel can help to generate a program for industrial devices transparently to the developer in order to simplify the signal capturing and storing further on the OPC-UA server. For instance, the program loads on a PLC device can be obtained directly by transformation of iMMAS models. Figure 1 shows the relevant features of an iMMAS infrastructure.

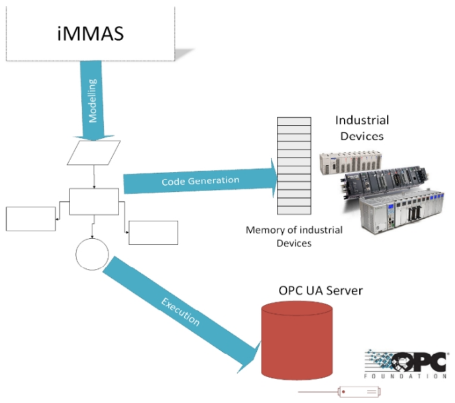


Fig. 1. The meta-model of iMMAS provides a language for designing models of IAS that can be executed on OPC-UA servers and can generate a program for PLC devices.

A. iMMAS Meta-Model

The meta-model establishes the bases of the modelling language defined by iMMAS to model any industrial process and the structure of the industrial devices at different abstraction layers. Moreover, developed models are independent of the software and hardware platform, since they are managed as abstractions that can be implemented in any programming language supporting abstract objects or data definition and can be further deployed on specific hardware or software platform.

The models conforming to iMMAS are also based on the information model of OPC UA, which it allows executing directly the models on an OPC UA environment as executable models. Figure 2 shows the main abstractions employed by the information model of OPC UA, while Fig. 3 shows an example of how the new abstractions of iMMAS are built over the model of OPC UA.

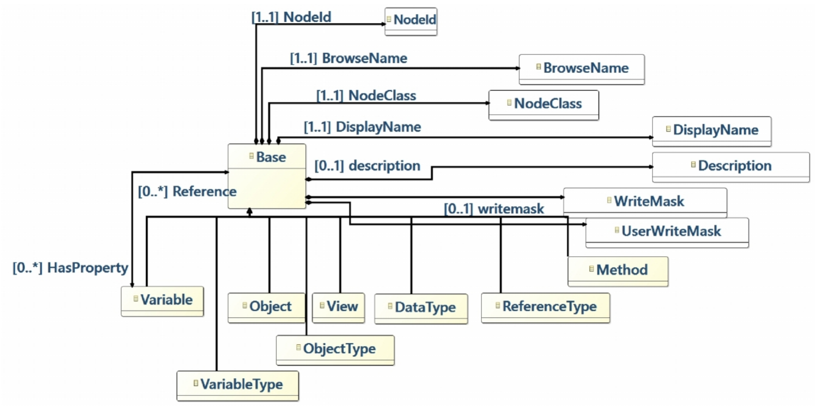


Fig. 2. Basic and Main concepts of OPC UA data Metamodel.

The executable models of iMMAS include not only the models, abstractions and their relationships, close to data managed by industrial devices. In addition, models and data are also accessible to any other OPC UA client interested in the supervision, monitoring or management of the industrial processes.

The architecture of iMMAS meta-model is composed of three hierarchical layers in a similar way as other standards as ISA 88 [28]:

– *Layer 0*: This layer is built over OPC UA. A selection of the abstract elements managed by the OPC UA meta-model is necessary to define the basic elements of iMMAS: PlcData and SimpleObject. PlcData encapsulates the minimum data unit managed in an industrial device, while a SimpleObject specifies the minimum abstraction unit in iMMAS. A simpleObject characterizes a valued data managed by the industrial system, such as a temperature signal, while PlcData indicates the datatype in use to represent the signal.

– *Layer 1*: It includes the elements related with the input and output signals used by industrial devices to control a concrete process. In addition, it includes the definition of specialized objects to manage data from these inputs and outputs.

– *Layer 2*: In this layer we can define the structure of complex abstractions as a composition of elements of a lower layer. In addition, we can determine the behaviour of these abstractions or the industrial process driven by an algorithm, a rule or a configuration set. Some examples are regulators, bombs, motors, etc.

The models in iMMAS are created applying a structured strategy in three levels based on the requirements of the process. For example, a model at layer 2 can include the definition of a pump including a simple analogue device of layer 1. However, at layer 1 the model can only include elements of the same level. Figure 3–Fig. 5 show the basic elements of iMMAS in each layer. In this paper, only a definition of layer 0 and 1 is presented in order to show how the traditional signals conveying from/to a PLC can be organized in OPC-UA server with iMMAS models.

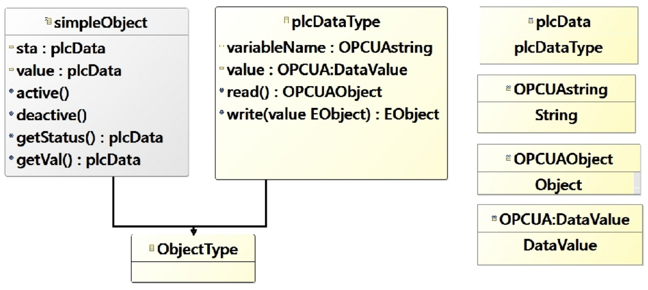


Fig. 3. iMMAS meta-model of level 0 built over OPC UA meta-model.

The meta-model of iMMAS at layer 0 defines a new type of object plcDataObject as the minimum data. This new type of object provides a new data type that allows working with different types of PLCs. It has a method for writing and reading values on a PLC memory address and two properties to store and name. A property variable Name is defined in this case. The next object is the main object type of this level SimpleObject. A SimpleObject has defined two properties, status and value of plcDataObject type and it contains methods for activating or deactivating the object and reading the value of the object.

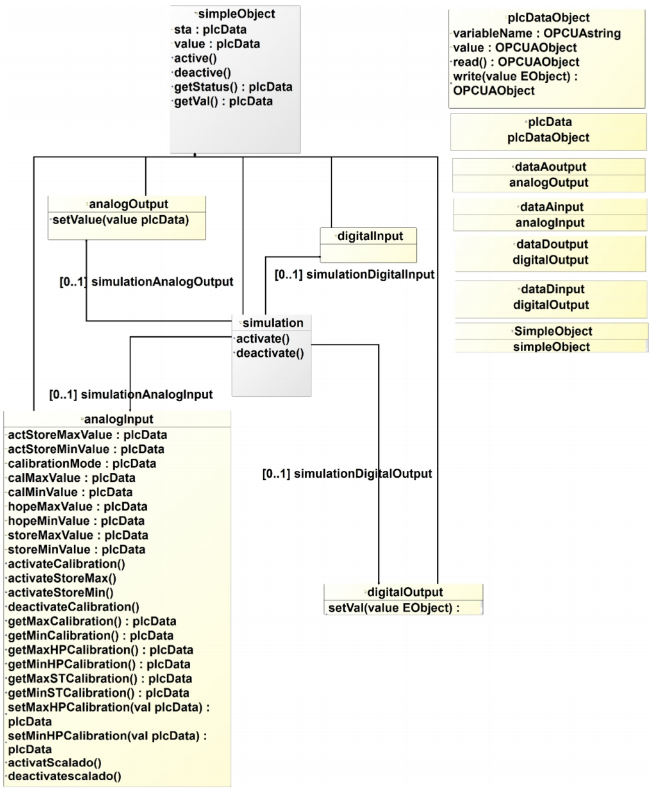


Fig. 4. Concepts (I/O) of iMMAS that form the level 1.

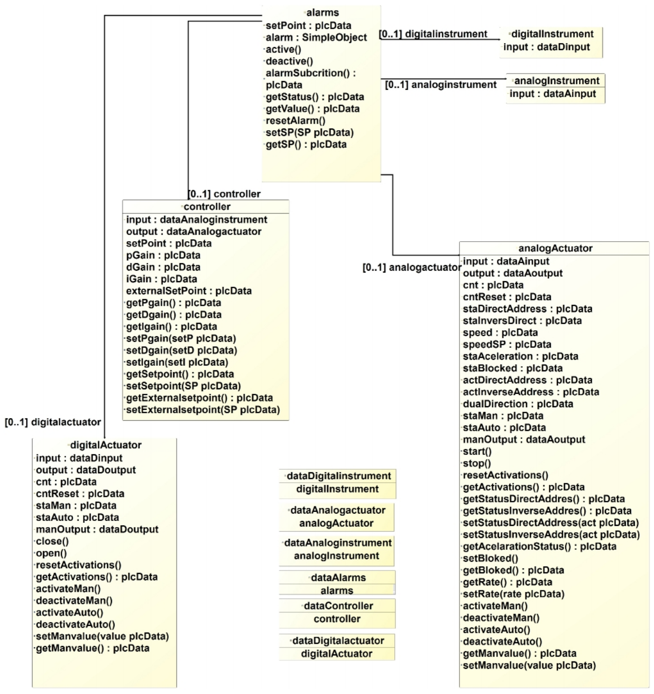


Fig. 5. Concepts (actuator/instrumentation), based in the level 1, for level 2.

The next sections show how the iMMAS models of layer 1 have been implemented in the OPC UA standard and how iMMAS models can be used to generate the programs to be loaded on a PLCs. In this way we can see how we can manage iMMAS in a real industrial system.

B. Deployment on OPC UA Server

OPC UA provides a more flexible approach for controlling and supervising an industrial process than classic OPC. The inclusion of iMMAS over OPC UA gives an additional flexibility to manage data of control processes. Commercial OPC UA clients have usually a lack of support for accessing abstract models of OPC UA servers. For this reason, we have developed a specific HMI application as an OPC UA client, which invokes the methods of iMMAS objects for controlling and monitoring the industrial process.

The definition of iMMAS models over an OPC UA information model guarantees that models are executable on an OPC UA environment. To test the applicability of our proposal, we have developed an OPC UA server using the C# stack of the OPC Foundation [25] and an implementation of iMMAS meta-model on OPC UA. The injection of the meta-model in OPC UA server is achieved by a description of all iMMAS elements in a XML file. Besides, XML files are compiled and imported to OPC UA server by using a tool of OPC Foundation named OpcUaModelCompiler.exe. An example of the definition of *plcData* and *SimpleObject* is shown below. Both objects are defined from the concepts that the information model of OPC UA provides [26]. In this case, *BaseObjectType* is used as a basic type to define PLC Data and SimpleObject.

This procedure guarantees that the models created in iMMAS are addressable by any OPC UA client. The following code shows a XML description of the iMMAS models.

```
<ObjectType SymbolicName="coreiMMASL1:plcData"
   BaseType="OpcUa:BaseObjectType">
      <Children>
         <Property SymbolicName="coreiMMASL1:deviceDirection"
           DataType="OpcUa:String" />
         <Property SymbolicName="coreiMMASL1:dataType"
           DataType="OpcUa:String" />
         <Variable SymbolicName="coreiMMASL1:value"
           DataType="OpcUa:DataValue" />
         <Method SymbolicName="coreiMMASL1:read"
           TypeDefinition="coreiMMASL1:TypeRead" />
         <Method SymbolicName="coreiMMASL1:write"
           TypeDefinition="coreiMMASL1:TypeWrite" />
      </Children>
 </ObjectType>
 <ObjectType  SymbolicName="coreiMMASL1:simpleObject"
    BaseType="OpcUa:BaseObjectType">
      <Children>
         <Object SymbolicName="coreiMMASL1:value"
           TypeDefinition="coreiMMASL1:plcData" />
         <Object SymbolicName="coreiMMASL1:sta"
           TypeDefinition="coreiMMASL1:plcData" />
         <Method SymbolicName="coreiMMASL1:getValue"
           TypeDefinition="coreiMMASL1:TypeRead" />
         <Method SymbolicName="coreiMMASL1:getStatus"
           TypeDefinition="coreiMMASL1:TypeRead" />
         <Method SymbolicName="coreiMMASL1:active"
           TypeDefinition="coreiMMASL1:TypeMethod" />
         <Method SymbolicName="coreiMMASL1:deactive"
           TypeDefinition="coreiMMASL1:TypeMethod" />
      </Children>
 </ObjectType>
```

C. Generation of a PLC Program.

The flexibility of the abstraction models in iMMAS can be used to implement code in industrial devices based on a MDE paradigm. Therefore, a program can be generated from abstract models specifically to a concrete PLC device, independently of its manufacturer.

In this section, an implementation of iMMAS model is performed for two PLC devices from different manufacturer (Siemens and Beckoff) based on the definition of user data types (UDT). The definition of UDT in any industrial software application provides a convenient way to organize the data model supported by the PLC device that can be propagated furthermore to other industrial systems such as OPC or SCADA. Then, UDTs are responsible to capture the data structure that can be recorded on the PLC memory.

The programming of a data model on a PLC can be achieved by transforming the iMMAS meta-model to the definition of the same types using UDT. Thus, we have created specific UDTs to group several primitive data type (real, int and bool) for each object of iMMAS meta-model (SimpleObject or PLCData). Table I shows how the simpleObject of iMMAS is defined by UDTs in Siemens, whereas Table II shows the same objects in Beckhoff.

TABLE I. DEFINITION OF SIMPLEOBJECT ACCORDING TO SIEMENS PLC USER DATA STRUCTURES (UDT).

TYPE "SimpleObjectReal" VERSION : 0.1 STRUCT value : REAL ; sta : BOOL ; END_STRUCT ; END_TYPE	TYPE "SimpleObjectBool" VERSION : 0.1 STRUCT value : BOOL; sta : BOOL ; END_STRUCT ; END_TYPE
TYPE "SimpleObjectInt" VERSION : 0.1 STRUCT value : DINT; sta : BOOL ; END_STRUCT ; END_TYPE	TYPE "SimpleObjectChar" VERSION : 0.1 STRUCT value : CHAR; sta : BOOL ; END_STRUCT ; END_TYPE

TABLE II. DEFINITION OF SIMPLEOBJECT ACCORDING TO SIEMENS PLC USER DATA STRUCTURES (UDT).

TYPE SimpleObjectReal :
STRUCT
value : REAL ;
sta : BOOL ;
END_STRUCT
END_TYPE
	TYPE SimpleObjectBool:
	STRUCT
	value : BOOL;
	sta : BOOL ;
	END_STRUCT
	END_TYPE

TYPE SimpleObjectInt:
STRUCT
value : DINT;
sta : BOOL ;
END_STRUCT;
END_TYPE
	TYPE SimpleObjectChar:
	STRUCT
	value : STRING;
	sta : BOOL ;
	END_STRUCT
	END_TYPE

IV. THE INDUSTRIAL PROCESS

For testing our solution, a small control system for regulating the temperature of a scale modelled-room is used. This modelled-room includes the following elements:

– *Fan.* It has an actuator controlled by an analog output to specify the speed of the fan from 0 to 100, and a sensor through an analog input to know the current speed of the fan in order to check the operation of the fan.

– *Heater.* It includes an actuator with an analog output to set the intensity of the heater from 0 to 100.

– *Temperature Sensor*. It includes an analog input to measure the temperature level of the room.

– *Slicing wall.* It contains a motor controlled by two digital outputs to move the wall to the right or to the left, and two switches connected to digital inputs to determine if the wall achieved the most right position or the most left position. This element can increment or reduce the volume of the room to be controlled, producing perturbation to the system.

– *Openable Roof.* The roof is controlled by two digital outputs to open or close it, and two digital inputs to know if the roof is open or closed.

For monitoring the system, we connected all the actuators and sensors to a Siemens PLC system with an external board containing digital inputs/outputs and analog inputs/outputs. All the devices were connected to a Profibus DP fieldbus. The inputs read analog signals of 0-10v, while the outputs were established by analog signals of 0-10v. Figure 6 shows the scale-modelled room and the industrial devices.

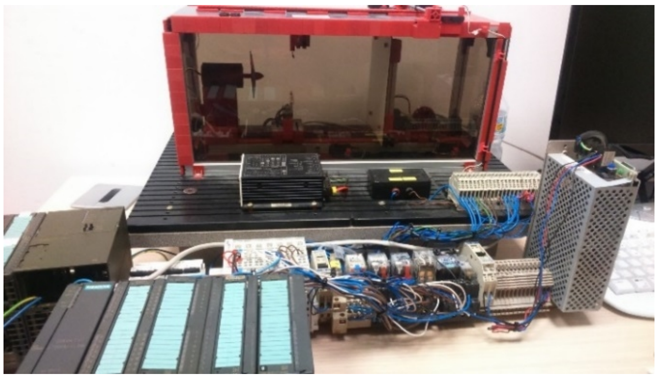


Fig. 6. Climate room model where the sensors and actuators are connected to the PLC device.

A. Model of the Plant

iMMAS enhances the building of the structure of the IAS as the first stage of modelling, because the abstractions contain the behaviour of the industrial process. Then, we can view an industrial process as a composition of a set of plant elements, which hides the abstractions.

Figure 7 provides an abstract model of the climate room managing abstractions defined in the level one and two of the meta-model of iMMAS. The model is structured in two subsystems: a *TemperatureControlSubsystem* subsystem to regulate the temperature of the climate room and a *PerturbationSubsystem* subsystem to modify the conditions of the climate room. Both subsystems have a structure based on the physical elements of the plant, starting from the abstractions available in the meta-model. In fact, there is a mapping between the physical elements of the plant and the objects of the model. This simplifies the implementation of the model, and gives a structured way to face the system.

The attributes of the classes define the properties of each object and their values identify the current status of the object. The object methods encapsulate the behaviour through actions, which include the invocation of methods in related objects. *TemperatureControlSubsystem* includes the following classes:

– *Fan:* A Fan object contains 2 objects, *SpeedOrder* based in an *analogOutPut* and a *realSpeed* object based in an *analogInput*.

– *Temperature:* it is a subclass of *analogOutput* class.

– *Heater:* A Heater object has only one analogOutput.

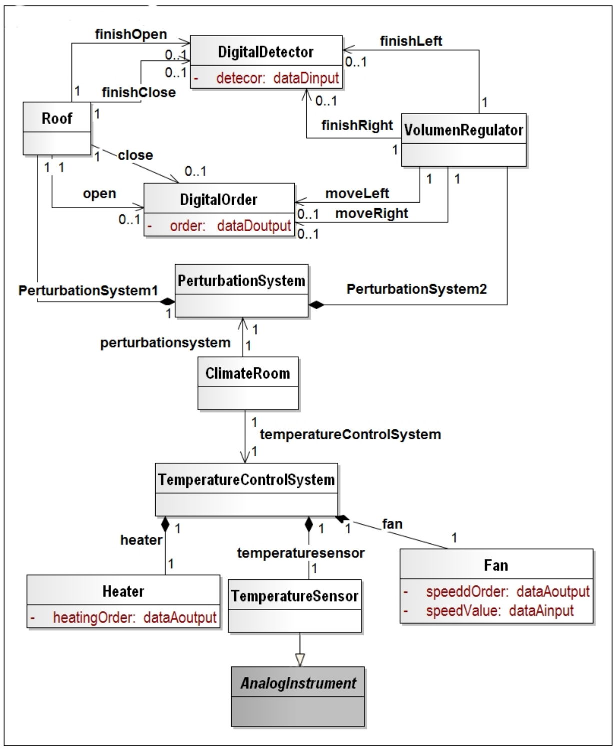


Fig. 7. Model based in iMMAS to collect the elements of climate room Model.

The main classes of PerturbationSubsystem are the following ones:

– *Wall:* This class models the behaviour of the wall. It has 2 *digitalOutput* to move the wall to the right or to the left and 2 *digitalInput* to know if the wall is in the left position or in the right position.

– *Roof:* This class abstracts the roof with 2 *digitalOutput* to open or close the roof and 2 *digitalInput* to know if the roof is open or closed.

B. Deployment on an OPC-UA Server

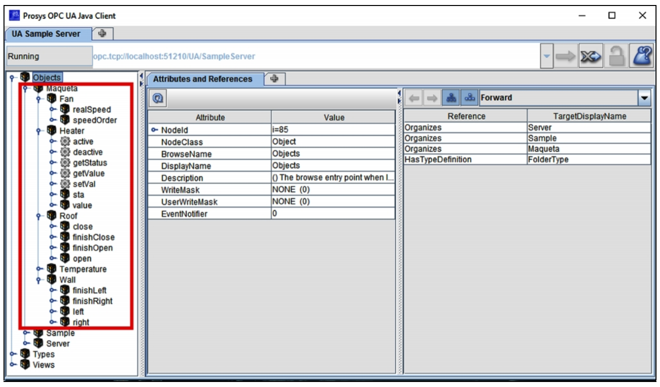
The model of Fig. 6 can be uploaded to any OPC-UA server, since the objects of this model are derived from the abstractions defined on the meta-model of iMMAS, which is based on OPC UA information model. Thus, each object of our model can be exported as an OPC-UA object and hosted to the OPC UA server, which can be accessible in a name space by any OPC UA client.

In order to upload the objects of the model in an OPC UA server, we need to obtain an XML file according to the specification of OPC UA. This file contains the objects of the model and their relationship, and they can be imported by the OPC UA server. As an example, we can see below the XML definition for OPC UA server of the Wall object.

The OPC UA specification of a Wall object based on iMMAS written in XML:

The Wall object includes the objects that determine its behaviour. Since OPC UA objects are included in nodes, it is possible to access any object by the OPC UA client. Any OPC UA client can read or write the attributes of any OPC UA object, when it is connected to the OPC UA server. Besides, an OPC-UA client can invoke object methods whenever the OPC-UA client has a full support of OPC-UA information model. Figure 8 shows how the objects can be seen by a commercial OPC UA Client.

```
<Object SymbolicName="coreiMMASL1:Wall"
TypeDefinition="OpcUa:FolderType">
      <Children>
         <Object SymbolicName="coreiMMASL1:right"
           TypeDefinition="coreiMMASL1:digitalOutput" />
         <Object SymbolicName="coreiMMASL1:left"
           TypeDefinition="coreiMMASL1:digitalOutput" />
         <Object SymbolicName="coreiMMASL1:finishRight"
           TypeDefinition="coreiMMASL1:digitalInput" />
         <Object SymbolicName="coreiMMASL1:finishLeft"
           TypeDefinition="coreiMMASL1:digitalInput" />
      </Children>
  </Object>
```



Attribute	Value
NodeId	i=85
NodeClass	Object
BrowseName	Objects
DisplayName	Objects
Description	() The browse entry point when l...
WriteMask	NONE (0)
UserWriteMask	NONE (0)
EventNotifier	0

Reference	TargetDisplayName
Organizes	Server
Organizes	Sample
Organizes	Maqueta
HasTypeDefinition	FolderType

Fig. 8. An OPC UA client can browse the objects of the model hosted on the OPC UA server.

C. Generation and Deployment of a PLC Program

Once the mapping between iMMAS meta-model and UDT types are established, a specific data model for a PLC can be generated from the specific model in iMMAS. Siemens provides convenient tools for programming PLC and its configuration. The data model structured with UDTs is stored in the PLC memory into data blocks (DBs) that can be addressed by OPC UA server. In our case we have used Step7 5.5 and not the last version of Siemens software (TIA Portal) because with this version of software we can work with old systems and devices such as a Siemens 300 PLC.

D. Generation and Deployment of a Program for Siemens PLC

Following the example shown in the former section, now we can see how the wall object has been defined by UDTs in a Siemens device in Table III.

TABLE III. UDT AND DB SPECIFICATIONS OF THE WALL OBJECT FOR SIEMENS PLC.

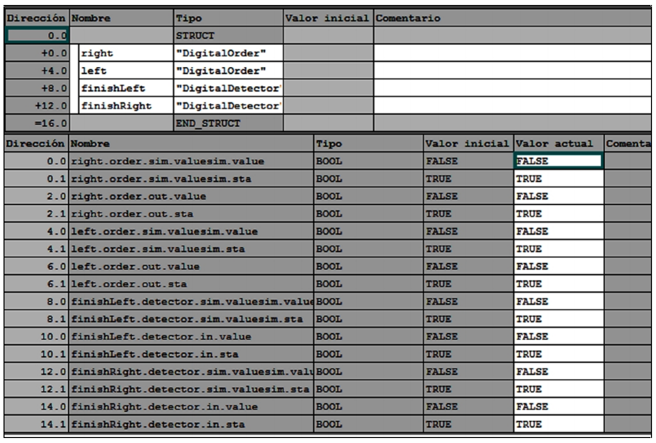
```
TYPE "DigitalDetector"
VERSION : 0.1
  STRUCT
   detector : "digitalInput";
  END_STRUCT ;
END_TYPE
```

```
TYPE "DigitalOrder"
VERSION : 0.1
  STRUCT
   order : "digitalOutput";
  END_STRUCT ;
END_TYPE
```

```
DATA_BLOCK "Wall"
TITLE =Element Climate Room
VERSION : 0.1
  STRUCT
   right : "DigitalOrder";
   left : "DigitalOrder";
   finishLeft : "DigitalDetector";
   finishRight : "DigitalDetector";
  END_STRUCT ;
BEGIN
   right.order.sim.valuesim.value := FALSE;
   right.order.sim.valuesim.sta := TRUE;
   right.order.out.value := FALSE;
   right.order.out.sta := TRUE;
   left.order.sim.valuesim.value := FALSE;
   left.order.sim.valuesim.sta := TRUE;
   left.order.out.value := FALSE;
   left.order.out.sta := TRUE;
   finishLeft.detector.sim.valuesim.value := FALSE;
   finishLeft.detector.sim.valuesim.sta := TRUE;
   finishLeft.detector.in.value := FALSE;
   finishLeft.detector.in.sta := TRUE;
   finishRight.detector.sim.valuesim.value := FALSE;
   finishRight.detector.sim.valuesim.sta := TRUE;
   finishRight.detector.in.value := FALSE;
   finishRight.detector.in.sta := TRUE;
END_DATA_BLOCK
```

Two new UDTs have been created for DigitalDetector and DigitalOrder objects to define the wall object. Then, DBs have been created with these model objects in Fig. 9.

From a model we can create the Siemens DB based in UDT by applying a transformation between both models. The result of the transformation is shown in the Fig. 9. Table IV provides an example of how the mapping is carried out described in XML. Basically, we associated one iMMAS object with one Siemens DB direction.



Dirección	Nombre	Tipo	Valor inicial	Comentario
0.0		STRUCT		
+0.0	right	"DigitalOrder"		
+4.0	left	"DigitalOrder"		
+8.0	finishLeft	"DigitalDetector'		
+12.0	finishRight	"DigitalDetector'		
=16.0		END_STRUCT		

Dirección	Nombre	Tipo	Valor inicial	Valor actual	Comenta
0.0	right.order.sim.valuesim.value	BOOL	FALSE	FALSE	
0.1	right.order.sim.valuesim.sta	BOOL	TRUE	TRUE	
2.0	right.order.out.value	BOOL	FALSE	FALSE	
2.1	right.order.out.sta	BOOL	TRUE	TRUE	
4.0	left.order.sim.valuesim.value	BOOL	FALSE	FALSE	
4.1	left.order.sim.valuesim.sta	BOOL	TRUE	TRUE	
6.0	left.order.out.value	BOOL	FALSE	FALSE	
6.1	left.order.out.sta	BOOL	TRUE	TRUE	
8.0	finishLeft.detector.sim.valuesim.value	BOOL	FALSE	FALSE	
8.1	finishLeft.detector.sim.valuesim.sta	BOOL	TRUE	TRUE	
10.0	finishLeft.detector.in.value	BOOL	FALSE	FALSE	
10.1	finishLeft.detector.in.sta	BOOL	TRUE	TRUE	
12.0	finishRight.detector.sim.valuesim.valu	BOOL	FALSE	FALSE	
12.1	finishRight.detector.sim.valuesim.sta	BOOL	TRUE	TRUE	
14.0	finishRight.detector.in.value	BOOL	FALSE	FALSE	
14.1	finishRight.detector.in.sta	BOOL	TRUE	TRUE	

Fig. 9. Siemens STEP 7 representation as result of the instantiation of the UDT/DB specification of the Wall.

TABLE IV. XML FILE WHICH INCLUDES THE MAPPINGS BETWEEN SIEMENS OBJECTS AND OPC UA OBJECTS.

```
<left>
 <object ID="1106" name="sta" >
       <dataType>
          BOOL
       </dataType>
       <deviceDirection>
          DB1.DBX 10.1
       </deviceDirection>
   </object>
   <object ID="1098" name="value" >
       <dataType>
          BOOL
       </dataType>
       <deviceDirection>
          DB1.DBX 10.0
       </deviceDirection>
   </object>
 </left>
```

```
<finishleft>
<object ID="1116" name="sta" >
      <dataType>
         BOOL
      </dataType>
   <deviceDirection>
         DB1.DBX 12.1
      </deviceDirection>
 </object>
 <object ID="1198" name="value" >
      <dataType>
         BOOL
      </dataType>
      <deviceDirection>
         DB1.DBX 12.0
      </deviceDirection>
 </object>
</finishleft>
```

```
<right>
   <object ID="917" name="sta" >
       <dataType>
          BOOL
       </dataType>
```

```
<finishright>
 <object ID="916" name="sta" >
      <dataType>
         BOOL
      </dataType>
```

<deviceDirection>	<deviceDirection>
DB1.DBX 14.1	DB1.DBX 16.1
</deviceDirection>	</deviceDirection>
</object>	</object>
<object ID="909" name="value" >	<object ID="998" name="value" >
<dataType>	<dataType>
BOOL	BOOL
</dataType>	</dataType>
<deviceDirection>	<deviceDirection>
DB1.DBX 14.0	DB1.DBX 16.0
</deviceDirection>	</deviceDirection>
</object>	</object>
</right>	</finishright>

E. Mobile HMI Client

The tools and frameworks based on SCADA or HMI do not support the full information model of OPC-UA model. We developed a specific HMI application that can be deployed on smartphones and tablets with Android ecosystem [29]. The tactile native application shows the value of the signals of each object in a graphic fashion as Fig. 10 shows.

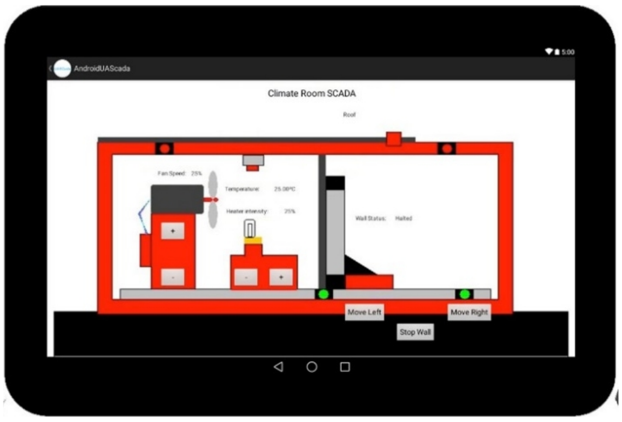


Fig. 10. A screen capture of the HMI on which a client OPC UA is running in a tablet device.

Conversely, the user can settle the set-points or provoke perturbations into the system. Internally, the OPC UA client invokes methods on iMMAS models hosted by OPC UA server.

V. CONCLUSIONS

OPC-UA have supposed a change of paradigm in the way that industrial systems and industrial processes can be organized and can be accessed by other industrial systems in an IAS. In this paper, we have presented a novel approach based on OPC-UA, named iMMAS, which provides a simple way to describe industrial processes as well as the organization of the IAS, managing new basic abstractions and concepts close to the language of industrial engineers supported by metamodels in three layers.

On the other hand, the application of MDE principles in iMMAS opens the possibility to generate software at different levels by means of the same conceptual models. These models can be executed directly facilitating its integration with the rest of industrial systems and providing at the same time a common data access that can be exploited by other systems such MES. In this work we explored how a program for a PLC device can be built applying a transformation process to data models hosted on an OPC-UA server. This transformation process is completely interoperable among different manufacturers of PLC devices as long as UDTs are known in a specific PLC solution.

Furthermore, the models designed with iMMAS can be also extended to HMI or SCADA scope, simplifying the visualization of the models to users. As a future work we want to improve the transformation process between different metamodels defining transformation rules that facilitates this process automatically.

Other important challenge to be addressed in a future work is the behavior modeling in iMMAS, based for example on some industrial standard for this purpose, such as ISA 88 [28].

REFERENCES

[1] M. Hollender, *Collaborative Process Automation Systems*. *International Society of Automation*, 2010, p. 420.

[2] P. Chiacchio F. Basile, D. Gerbasio, “On the implementation of industrial automation systems based on PLC”, *IEEE Trans. on Automation Science and Engineering*, vol. 10, no. 4, pp. 990–1003, 2013. [Online]. Available: https://doi.org/10.1109/TASE.2012.2226578

[3] Programmable controllers Part 3: Programming languages. *IEC International Standard IEC 61131-3:2013*. [Online]. Available: https://webstore.iec.ch/publication/4552

[4] J. Fuchs, S. Feldmann, C. Legat, B. Vogel-Heuser, “Identification of design patterns for IEC 61131-3 in machine and plant manufacturing”, in *IFAC Proc. Volumes,* vol. 47, no. 3, 2014, pp. 6092–6097. [Online]. Available: http://dx.doi.org/10.3182/20140824-6-ZA-1003.01595

[5] K. Thramboulidis, “A framework for the implementation of industrial automation systems based on PLCS”, *Cornell University Library*, 2014. [Online]. Available: https://arxiv.org/abs/1402.3920

[6] V. Vyatkin, “Software engineering in industrial automation: State-of-the-art review”, *IEEE Trans. Industrial Informatics*, vol. 9, no. 3, pp. 1234–1249, 2013. [Online]. Available: https://doi.org/10.1109/TII.2013.2258165

[7] S. Yang, J. Lee, H. Kao, “Service innovation and smart analytics for industry 4.0 and big data environment”, *Procedia CIRP*, vol. 16, pp. 3–8, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.procir.2014.02.001

[8] M. Keller, M. Rosenberg, M. Brettel, N. Friederichsen, “How virtualization, decentralization and network building change the manufacturing landscape: An industry 4.0 perspective”, *International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering*, vol. 8, no. 1, pp. 37–44, 2014. [Online]. Available: http://www.waset.org/publications/9997144

[9] I. Grobelna, M. Grobelny, “Logic controller design system supporting uml activity diagrams”, in *22nd Int. Conf. Mixed Design of Integrated Circuits & Systems (MIXDES)*, pp. 624–627, 2015. [Online]. Available: https://doi.org/10.1109/MIXDES.2015.7208599

[10] M. Obermeier, F. Jobst, B. Vogel-Heuser, S. Braun, K. Schweizer, “Usability evaluation on teaching and applying model-driven object oriented approaches for PLC software”, in *American Control Conf.*, pp. 4463–4469, 2012.

[11] Batch control-part 1: Models and terminology. *International standard IEC 61512-1*:1997. [Online]. Available: https://webstore.iec.ch/publication/5528

[12] Enterprise - control system integration part 1: Models and terminology. *International standard IEC 62264-1*:2013. [Online]. Available: https://www.iso.org/standard/57308.html

[13] Z. X. Guo, X. Q. Xie, Z. G. Ni, “The application of OPC da in factory data acquisition”, *IEEE Int. Conf. Computer Science and Automation Engineering (CSAE)*, pp. 209–212, 2012. [Online]. Available: https://doi.org/10.1109/CSAE.2012.6272760

[14] M. S. Mahmoud, M. Sabih, M. Elshafei, “Using OPC technology to support the study of advanced process control”, *ISA Trans.*, vol. 55, pp. 155–167, 2105. [Online]. Available: http://dx.doi.org/10.1016/j.isatra.2014.07.013

[15] J. Cabo, M. Wimmer, M. Brambilla, “Model-driven software engineering in practice”, *Synthesis Lectures on Software Engineering, Morgan & Claypool*, vol. 1, pp. 1–182, 2012. [Online]. Available: https://doi.org/10.2200/S00441ED1V01Y201208SWE001

[16] A. Rodrigues da Silva, “Model-driven engineering: A survey supported by the unified conceptual model”, *Computer Languages, Systems & Structures*, vol. 43, no. 1, pp. 139–155, 2015. [Online].

Available: http://dx.doi.org/10.1016/j.cl.2015.06.001

[17] MDA (Model Driven Architecture) guide version 1.0.1. *OMG (Object Management Group)*. 2001.

[18] *Meta object facility (MOF) core specification, v2.4.2*. 2014. [Online]. Available: http://www.omg.org/spec/MOF/2.4.2/

[19] I. Jacobson, G. Booch, J. Rumbaugh, *The unified modeling language user guide*". Addison Wesley, 1999.

[20] Omg systems Modeling language. *Technology Standards Consortium.* [Online]. Available: http://www.omgsysml.org/

[21] L. Bassi, C. Secchi, M. Bonfe, C. Fantuzzi, "A SysML-Based methodology for manufacturing machinery modeling and design", *IEEE/ASME Trans. Mechatronics*, vol. 16, no. 6, pp. 1049–1062, 2011. [Online]. Available: https://doi.org/10.1109/tmech.2010.2073480

[22] N. Papakonstantinou, S. Sierla, "Generating an Object Oriented IEC 61131-3 software product line architecture from SysML Emerging", *IEEE 18th Conf. Technologies & Factory Automation (ETFA)*, 2013. [Online]. Available: https://doi.org/10.1109/ETFA.2013.6648057

[23] C. Fan, Y. Chang, S. Yih, "An editing system converting a UML state diagram to a PLC program", *Advances in Intelligent Systems & Applications*, vol. 2, pp. 647–655, 2013. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-35473-1_64

[24] OPC DA, HDA and E&A specifications. *OPC Foundation*. [Online]. Available: https://opcfoundation.org/

[25] OPC UA Specifications. *OPC Foundation*. Available: [Online]. https://opcfoundation.org/

[26] W. Mahnke, L. Stefan-Helmut, M. Damm, "OPC Unfied Architecture", *Springer*, 2009. [Online]. Available: https://doi.org/10.1007/978-3-540-68899-0

[27] Opc unified architecture. Part 3: Address space model. *OPC Foundation*. [Online]. Available: https://opcfoundation.org/

[28] ISA88 Batch Control*.* *Instrument Society of America ISA*. [Online]. Available: https://www.isa.org/isa88/

[29] J. Gutierrez-Guerrero, J. A. Holgado-Terriza, "Mobile human machine interface based in OPC UA for the control of industrial processes", *Actas de las XXXVI Jornadas de Automatica*, pp. 1073–1080, 2015. [Online]. Available: http://www.ehu.eus/documents/3444171/4484750/ 154.pdf