An Industry 4.0 Self Description Information Model for Software Components contained in the Administration Shell

Luís Neto*†§, Gil Gonçalves*†§, Pedro Torres‡¶, Rogério Dionísio‡¶, Sérgio Malhão‡‖
*SYSTEC (Research Center for Systems & Technologies)
†FEUP, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal
‡Escola Superior de Tecnologia, Instituto Politécnico de Castelo Branco
§Email: {lcneto,gil}@fe.up.pt
¶Email: {pedrotorres,rdionisio}@ipcb.pt
‖Email: smalhao@ipcbcampus.pt

***Abstract*—Industry 4.0 is the movement towards a fourth industrial revolution that will consist in the digitization and integration of all value chain. In Europe, this movement is led by the German *RAMI 4.0* (Reference Architecture for Industry 4.0) proposal, which is attracting a lot of attention from industry, academia and other practitioners. Under the RAMI 4.0 scope there is an Administration Shell proposal to abstract physical and logical assets in a standardized way. Once abstracted, assets become Industry 4.0 Components and can be fully integrated in the Cyber Physical Production System or value chain. This work focuses on the utilization of software components within the Administration Shell. There is a necessity to represent software components and their relation to industrial asset. Therefore, control and monitoring applications involving software components and other assets can be represented in compliance with the I4.0 Component Model. To address this necessity the Smart Object Self Description information model is proposed and applied to a real case study scenario.**

Keywords–Information Model; Component Based Software Engineering; Smart Component; Administration Shell; Industry 4.0.

I. INTRODUCTION

Industry 4.0 (I4.0) is the movement that aims to transform the traditional factory into a smart factory. There is a hype around this movement fueled by great expectations in the way industry will transform the value chain, business models and economy [1].

In terms of path to effectively create the smart factory, there are several of models with different specificities proposed by different countries [2]. In Europe, RAMI 4.0 [3], proposed by several German organizations gathered under Platform Industrie 4.0, seems to join the bigger consensus [4]. RAMI 4.0 most widespread concept is a tree dimensional map that combines: 1) the hierarchy of Industry 4.0 components, according to ISA 95 (International Society of Automation 95); 2) the product life cycle and it value chain, from conception to disposal; 3) the factory architecture perspectives, from assets to the whole organization and business processes. The main objective of this model is to create a clear understatement of all participants within the Industry 4.0 and across the value chain [1].

One of the core technological aspects for Industry 4.0 is *The Industrie 4.0 Component model* [5]. The components model for Industry 4.0 was developed by the participants of Platform Industrie 4.0 to help equipment producers and system integrators to create standardized I4.0 compatible hardware and software components [1]. This component model specifies that each asset (logical or physical) must be encapsulated by a standardized digital container – the *Adminsitration Shell* – that will enable description, collaboration and communication among all I4.0 Components.

A. Administration Shell

The *Administration Shell* (AS), acts as an interface connecting all physical and logical assets to the I4.0 compliant network, therefore creating an I4.0 compatible Cyber Physical Production System (CPPS). [6] is a proposal for the general structure of the AS, as proposed by Plattform Industrie 4.0, and therefore it was a reference to this work. The things abstracted by an AS are diverse and some are manufacturer dependent, so the AS maintains an internal interface specific for each asset, as in Figure 1. The AS has also an external interface, which is responsible for communication with the I4.0 network. Another peculiarity of the AS is that it can represent passive and active assets. One example of a passive asset is a purely analogical machine or tool which might be important to represent digitally. On the other hand, an example of an active asset is a complex machine incorporating digital control units capable of processing and communication. The asset itself can be composed of other assets as is the case of a machine whose sub-systems can be represented individually, it is also the case of a production line or even the entire factory. The representation of an asset, once abstracted by an AS, is also commonly called the *digital twin*.

Once an asset is encapsulated by an AS it becomes an I4.0 Component (Figure 1) and can participate in the I4.0 compatible CPPS, which is formed by other I4.0 Components. A descriptive diagram explaining the process of encapsulation and the main advantages of the I4.0 Component Model are described in detail in [1] [5].

B. PRODUTECH-SIF Project

The work described in this paper was performed in the scope of PRODUTECH-SIF (Solutions for the Industry of the Future) project [7]. This is a Portuguese National initiative with a research agenda that comprises a set of R&D activities in key domains with the objective of enabling the digital transformation of the Portuguese industry. One of the base activities is the study and implementation of base technologies to create CPPS, whose first results are described in this paper. As already discussed, the AS is a base technology to

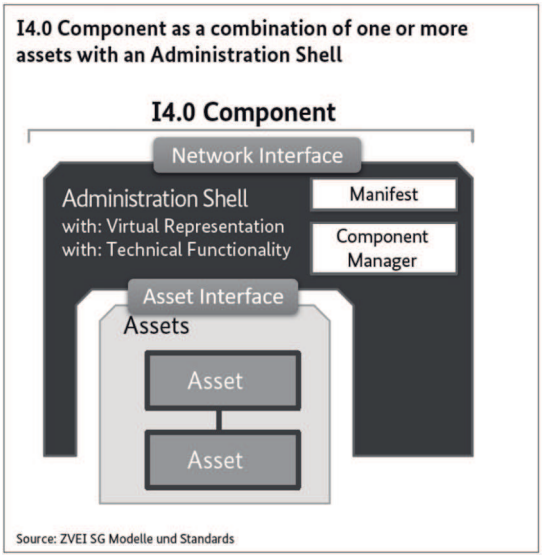

Figure 1. Representation of the Administration Shell application to an Asset to form an I4.0 Component [6].

create CPPS. As this concept is proposed under the scope of RAMI 4.0, the AS was chosen to convert the participating companies assets in I4.0 Components. Therefore, preparing the Portuguese industry for the upcoming establishment of I4.0 based technologies.

The methodology followed started with requirements assessed by means of inquiries to all companies participating in the project. Thereafter, a company responsible to host the pilot demonstration of the project was visited to prepare a complete case study, upon which all developments presented in this paper were based. All hardware requirements are tackled by a SmartBox described in Section II. The AS of the project is realized by the SmartObject concept, which is described in Section III. The case study, based on the pilot demonstration of the project, is presented in Section IV. A series of problems and the motivation for this work are presented in Section V. The main contribution of this work is a proposal for an information model used for self-description and (re)configuration of SmartObject's, presented in Section VI. The paper finishes with some conclusions and future work in Section VII.

II. SmartBox

The *SmartBox* is a smart hardware developed over the National Instruments (NI) cRIO-9040 platform that runs the *SmartObject* and enables the remote connectivity with machines in the shop floor. NI cRIO is a Programmable Automation Controller (PAC) that allows extremely high speed measurements and also allows to perform software-defined hardware through an internal FPGA. One important component is the modular Real-Time PAC platform that enables flexible data acquisition and actuation based on hardware modules that can be incorporated, including machine vision systems. In particular, the cRIO-9040 has a dual-core 1.3 GHz processor, 2 GB DRAM and 4 GB Storage with 4 slots for different I/O modules.

In the scope of the PRODUTECH-SIF project, the *SmartBox* was programmed for Device-to-Device (D2D) connections supported by the OPC-UA protocol and Device-to-Server (D2S) connections with the MQTT protocol. A *SmartBox* can be installed to manage one machine or a groups of machines. In both cases, the *SmartBox* acts as an OPC-UA server or a MQTT publisher of the IIoT system architecture, as an Edge-Node between sensors and actuators and the information systems. Figure 2 illustrates the *SmartBox* that communicates over EtherCAT with Remote I/Os for shop floor scenarios where data or command information flows between machine physically distant from each other. Currently installed on the *SmartBox*, there is multifunctional module (NI-9381) with digital and analogue I/O to receive data from digital sensors or analogue voltage values between 0V - 5V with medium to low resolutions needs. The same module is used for actuation. The *SmartBox* has also an AC differential input module (NI-9215) installed, with 4 channel of $\pm$ 10 V, 16 bits for acquisition of analogue signals such as acoustic or vibrations. Another available module (NI-9239) has a 24 bits ADC, and 4 channels. It is used for vibrations and small signals measurements produced by magneto-resistive sensors.

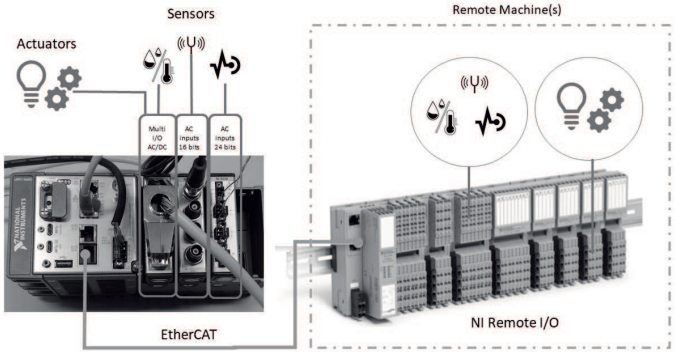

Figure 2. *SmartBox* with Remote I/Os for scenarios with multiple machines installation.

What distinguishes the *SmartBox* from other conventional controllers is the capacity to implement machine learning techniques. For example, it can be used to measure vibrations in induction motors and run classification methods that automatically detects and alerts for anomalies like possible misalignments of the rotors and worn out bearings and gears. Another possibility is the forecasting of temperature in specific machine areas based on the history and based on this, act in the machine or simply monitor the current state. These are only 2 examples already tested with the *SmartBox*, but another supervised, unsupervised or reinforced machine learning techniques can be implemented. The *SmartBox* can be reconfigured by adding or removing specific hardware modules, and each can be also reconfigured by software. This versatility is an asset for the implementation in different scenarios presented by the industrial partners of PRODUTECH-SIF project.

III. SmartObject

The SmartObject (SO) is an implementation of the *SmartComponent* concept [8], defined by Smart Component's community [9] [10]. In the scope of the PRODUTECH-SIF project, the SO implementation is being extended to consider the AS requirements defined in [6]. Therefore, the SO will act as the AS for encapsulating and converting the case study assets in *I4.0 Component's*. To make assets transparent to each other,

i.e., capable of mutual cooperation and understatement, each SO will maintain and make available to the CPPS a *manifest* describing each asset. This paper focuses on the common data model created for that purpose, the *Smart Object Self Description (SOSD)*.

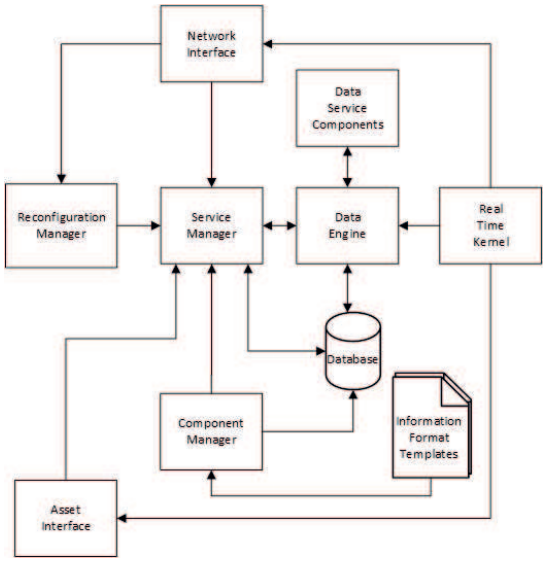

Figure 3. SmartObject Block Diagram.

In software terms, the SO is a component framework [11] constituted by a composition tool and a runtime environment. The diagram of Figure 3 shows the main components of the runtime environment, which will be deployed in each SmartBox. The composition tool allows to develop, maintain and deploy software components compatible with the runtime environment. During deployment an user can create a new composition or modify any one running in a certain SO. The deployment environment embeds a canvas that displays the composition and allows new components to be dragged in and interconnected by means of it's interfaces. This way, a control engineer doesn't need to know any specific details of software. All he sees is a set of black boxes whose functionality and interfaces are well documented and that can be used to build control or monitoring applications. Neto and Gonçalves [12] explain and survey component frameworks applied to industrial environments. A comprehensive state of the art of software engineering in industrial automation is presented by Vyatkin [13].

IV. Case Study

The pilot demonstration of the PRODUTECH-SIF project defines the case study for this work. A Portuguese company produces labels and technical narrow fabrics for clothing and other applications is the application target. A part of the production line consists in a variety of looms that produce the labels. Due to its business nature, the company has been acquiring new looms across the years, having now a diversified set of machines, from older to new ones. A set of intermediate aged looms were chosen to constitute the pilot and therefore the application requirements. To support the pilot description one can refer to Figure 4, which illustrates the physical architecture of the case study. These machines have some control electronics and are capable of signaling some errors related to severe failures and others related to small production issues, like rupture of the threads and fabrics. Despite that, these machines are not capable of communication and the only way to acknowledge production problems is through a warning light tower connected to the loom. An operator regularly checks the light towers for faults. A small screen used to upload the label design can also be used to check error codes. The company identified a set of problems that should be tackled by the proposed combination of SmartBox and SmartObject:

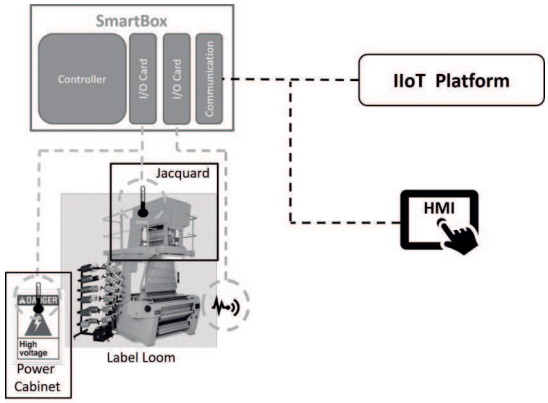

Figure 4. Case Study Physical Architecture.

- **P1**: The power cabinet of the machines is cooled by a fan. Due to mechanical wear, or due to the amount of dust in the air, generated by the threads being processed, the fan fails compromising the cooling. The machine is not prepared to react to this problem and sometimes happens that the power components heat up too much leading to severe failure.
- **P2**: The loom has a cutting system based on electrical resistance elements which can be positioned to cut the fabric according with the required label dimensions. These elements sometimes fail and the machine has no sensing to detect the issue.
- **P3**: The part of the loom which controls the needles is a complex electric-mechanical system called *Jacquard*. This part can sometimes overheat disrupting the mechanical elements of the system and leading to unexpected failures that the machine cannot predict or warn due to the lack on sensing.
- **P4**: There are several issues related with the threads and fabric processed as these break very often. The warning is given by the light tower and there is no automatic way to warn the loom is stopped. Also, related to this, there is no way to know for how long the machine has stopped and what is the rate of production.

V. Problem, Requirements and Motivation

Table I shows the solution proposed for each problem of the case study. A set of requirements result from these proposals. All the hardware aspects are solved by the SmartBox modular capacity. All the software aspects are dealt by the SO as follows. The SO will act as a runtime environment and AS for the assets outlined in the case study. People from the

company in charge of the production line will configure the SO through the composition tool discussed in Section III. The tool will have a list off all assets in the shop floor, so that the operator can drag and drop the assets in the canvas to create the desired application. New associations can be made between machines, components, sensors, actuators and software components to create control or monitoring applications defined by the composition workflow. Figure 7 illustrates the vision that the control engineer will have of the monitoring application for the described case study.

TABLE I. CASE STUDY PROPOSED SOLUTIONS.

Problem	Solution
P1	Install a temperature sensor in the power cabinet. Use a software component programmable alarm to alert for excessive temperature.
P2	Install a temperature sensor in the cutting element. Use a software component programmable alarm to alert for excessive temperature.
P3	Install a temperature sensor in the Jacquard. Use a software component programmable alarm to alert for excessive temperature.
P4	Install an inductive/encoder sensor in the loom. Use a software component to convert the sensor impulses into meters.

For all the functionalities discussed so far to be possible, there was a necessity to use an information model that would be capable of tackling the following requirements:

- The SO must abstract all machines and its components (loom and respective power cabinet and Jacquard), sensors and actuators.
- The SO must abstract all software components used and the respective instances.
- The SO must allow for any external actor to subscribe data produced by any device or software component instance.
- The SO must allow to define hierarchy and subscription relations among all devices, software component instances and external actors.

Although the RAMI 4.0 defines ontologies and information models to support communication and representation of several industrial assets throughout its respective life cycles [14] [15], it seems to lack in specification for the composition of logic assets – such as software components [16] – in workflows. A set of applications that are of great importance for a smart factory, such as: condition monitoring, predictive maintenance, self-reconfiguration, quality control and fault detection; depends of software components. Therefore, there should be an information model which could represent all software component peculiarities and its relation with industrial assets to form applications. In this paper, we propose such an information model.

VI. Smart Object Self Description

The SOSD defines classes and properties for all assets specified in the use case, taking in consideration the following requirements:

- **Physical Assets:** The model is capable of representing: machines, machine components, sensors and actuators. It also must be capable of representing dependencies and connections between these. This is of major importance if we want to contextualize information of a given sensor, or if components need to be represented as parts of some machine. As an example, in Figure 7, it can be seen that there are: 1) relations between a machine and it's constituent parts, the hierarchical references between the *Loom* and it's *Power Cabinet* and *Jacquard* components; 2) relations of contextualization between machines or machine parts and sensors, as between the *Loom* and *Inductive Sensor*, or between the *Jacquard*/*Power Cabinet* and the *Temperature Sensors*. **For representing physical assets the class *SOSD:DeviceType* (left in Figure 7) is used.** This class has sub-types: 1) *Device*, to represent machines; 2) *Component*, to represent machine components; 3) *Sensor*, to represent sensors; 4) *Actuator*, to represent actuators.
- **Services and Service Instances:** A service, in the SOSD context, represent some algorithm or computational process that is available in the CPPS network. A SO or any other node in the CPPS can announce its services and respective capabilities, e.g. data processing services like a Fast Fourier Transform (FFT) or a simple alarm. A service corresponds to a software component that can be instantiated by including it in the composition design. Each new instance created can be interfaced with providers and subscribers, e.g. the *Alarm* and *Impulses to Meters* instances in the view of Figure 7 are fed by sensors and feed external nodes in the CPPS. **Class *SOSD:ServiceDescriptionType*, represented in Figure 8 under the folder *ServiceDescriptionSet*, is used to represent software components maintained by the SO composition tool. Class *SOSD:ServiceInstanceType*, represented on the left side of Figure 7, is used to represent instances of software components running in a SO.**
- **Points:** A Point represent a node in the CPPS network, e.g. an Human Machine Interface (HMI) device used by the operators to check production variables or an Industrial Internet of Thing's (IIoT) Platform to maintain production telemetry data. In the composition of Figure 7, the *HMI* and *IIoT Platform* are notified each time the *Alarm* or *Impulses to Meters* service instances produce a new output value. **The class *SOSD:PointDescriptionType*, on the left side of Figure 7, is used to represent other nodes in the CPPS with which the SO can communicate.**
- **Variables, Parameters and Methods:** All static and dynamic variables and parameters associated to assets, services or endpoints must be represented. Static variables represent information about some entity, e.g. in case of machine, the manufacturer, model and serial number. Dynamic variables represent information generated during production, e.g. a sensor value or a service output. Parameters represent values that can be changed to modify or tune some process, e.g. a welding machine laser power or an alarm minimum and maximum thresholds. A Method represents an simple routine that can be invoked, e.g. a calibration method for a sensor or a stop routine for a machine. For simplicity and compatibility reasons, each Variable and Parameter defined under the classes proposed

by SOSD, are represented using *BaseDataTypes*, *VariableTypes* and *Method semantic* from the OPC UA Information Model (OPC UA Specification: Part 5) [17]. In Figure 8, under the main classes representation, examples of properties and variables are illustrated.

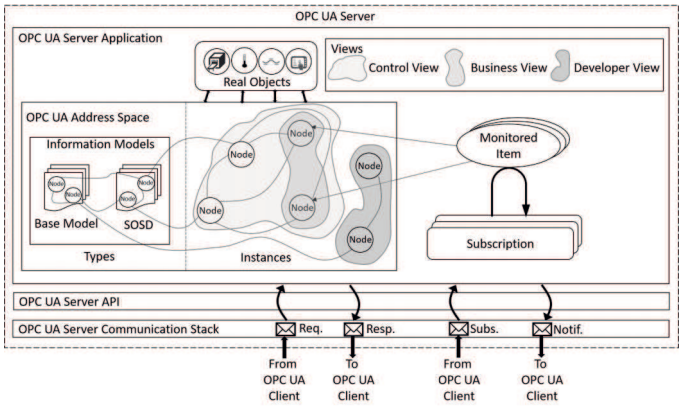

Figure 5. OPC UA Server with SOSD (Adapted from [18]).

OPC UA is the *de facto* standard communication protocol under the RAMI 4.0 proposal [19]. For that reason, the SO network interface embeds an OPC UA Server. The SOSD model was entirely mapped in the OPC-UA native and Data Access types and information model. Once an asset is physically or wireless connected to the SmartBox, its information will be mapped in the SOSD model (Figure 5), and make available to the CPPS by a local OPC-UA server instance.

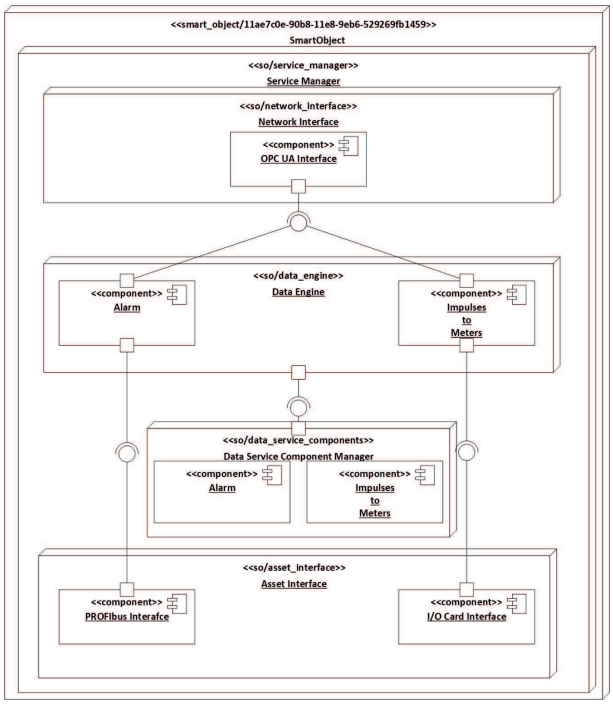

Figure 6. SmartObject Component Developer View.

Figure 5 illustrates the functionality details of an OPC UA server embedded in the SO. Each node represented by a SOSD class will have a direct dependency to the SOSD model. Each property, variable or method of an SOSD class will reference the OPC UA Base Model.

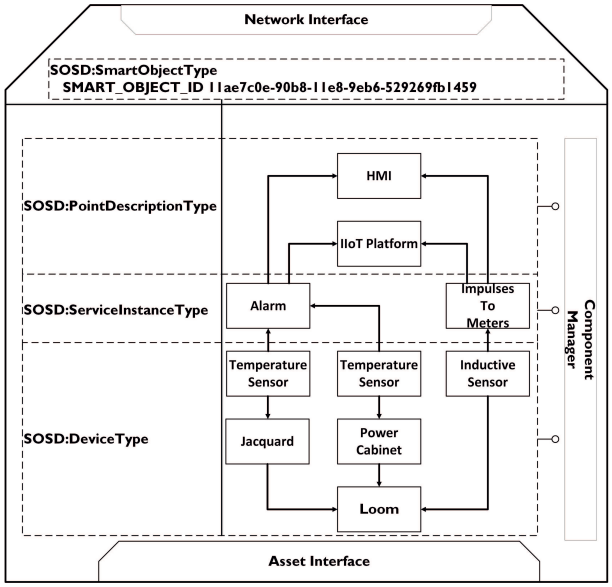

Figure 7. SmartObject Control Engineer View.

The combination of the SOSD with the OPC UA views allows to separate the visions of the software developer and the industrial domain expert. Figure 6 relates to how the software components developer sees the composition running in the SO. In the bottom there are the components that implement the drivers for the temperature and impulse sensors. In the middle there are the components that implement the alarms and impulse to meters algorithms. In the top there is a component with an embedded OPC UA driver. The software architect can grasp the SO composition and develop, modify and reuse components as needed by the domain experts. Figure 7 relates to how the industrial domain expert sees the composition running in the SO. Each block in the figure corresponds either: 1) to a physical asset as discussed in the case study (Section IV); 2) to a software component as discussed previously. The figure shows a monitoring application that the control engineer could assemble by dragging and dropping the blocks using the SO composition tool.

Another important concept provided by OPC UA is the views. Figure 5 gives a general idea of views: *Control View*, *Business View* and *Development View*. These views define which nodes are presented to different users groups. A control engineer, who has the expertise to build control and monitoring applications, only cares about nodes relevant to build or watch technical compositions, as in Figure 7. A software component developer only cares about nodes relevant to the SO architecture, as in Figure 6. The business view, although not represented, can be used to specify users only interested in see components related to production performance and other Key Performance Indicators. By creating only the essential classes and structures, and combining these with the OPC UA features, the SOSD was demonstrated to tackle all requirements of the Case Study, constituting a solid start point for software component models assembled under the scope of AS and SO. Figure 8 shows a tree view of the SOSD model embedded in a working SO OPC UA server. Due to constraints in size and

Figure 8. SmartObject OPC UA Server SOSD Model View.

simplification this representation cannot include all details, but it gives a general idea of how the model works in practice.

VII. Conclusions and Future Work

This work proposes the SOSD information model, whose objective is to establish a standard for the representation of software components, assets and compositions between them. This model was successfully embedded in a OPC UA server and used to model a real case study.

Future work will involve the modeling of more complex scenarios and the extension to support real-time quality of service requirements in the composition. The capability of defining specific views for different components also needs to be explored to support a business perspective.

Acknowledgment

This research was supported by the project *PRODUTECH-SIF - Solutions for the Industry of the Future*, financed by the Portuguese National program COMPETE 2020 and Portugal 2020.

References

[1] F. Zezulka, P. Marcon, I. Vesely, and O. Sajdl, "Industry 4.0–an introduction in the phenomenon," IFAC-PapersOnLine, vol. 49, no. 25, 2016, pp. 8–12.

[2] G.-G. et al., "Towards a semantic administrative shell for industry 4.0 components," in 2016 IEEE Tenth International Conference on Semantic Computing (ICSC). IEEE, 2016, pp. 230–237.

[3] M. Hankel and B. Rexroth, "The reference architectural model industrie 4.0 (rami 4.0)," ZVEI, vol. 2, 2015, p. 2.

[4] M. Hermann, T. Pentek, and B. Otto, "Design principles for industrie 4.0 scenarios," in 2016 49th Hawaii international conference on system sciences (HICSS). IEEE, 2016, pp. 3928–3937.

[5] M. A. . C. K. Hoffmeister Festo. Industrie 4.0: The industrie 4.0 component. [retrieved: 6, 2019]. [Online]. Available: \url{https://www.zvei.org/fileadmin/user_upload/Themen/Industrie_4.0/Das_Referenzarchitekturmodell_RAMI_4.0_und_die_Industrie_4.0-Komponente/pdf/ZVEI-Industrie-40-Component-English.pdf}

[6] P. e. a. Adolphs, "Structure of the administration shell. continuation of the development of the reference model for the industrie 4.0 component," ZVEI and VDI, Status Report, 2016.

[7] "PRODUTECH-SIF - solutions for the industry of the future," http://mobilizadores.produtech.org/en/produtech-sif, [retrieved: 6, 2019].

[8] L. Neto, J. Reis, R. Silva, and G. Gonçalves, "Sensor selcomp, a smart component for the industrial sensor cloud of the future," in 2017 IEEE International Conference on Industrial Technology (ICIT). IEEE, 2017, pp. 1256–1261.

[9] Pinto, Rui; Reis, João; Silva, Ricardo; Pesch, Michael, and Gonçalves, Gil, "Smart sensing components in advanced manufacturing systems," International Journal On Advances in Intelligent Systems, vol. 9, no. 1 & 2, [retrieved: 6, 2019]. [Online]. Available: {https://repositorio-aberto.up.pt/bitstream/10216/109876/2/239571.pdf}

[10] S. Micheler, Y. M. Goh, and N. Lohse, "Green paper: R&d priorities for smart manufacturing components and systems," 05 2017.

[11] Neto, Luis; Madsen, Anders; Jeppesen, Nicolaj Søndberg; Silva, Ricardo; Reis, João; McIntyre, Peter, and Gonçalves, Gil, "A Component Framework as an Enabler for Industrial Cyber Physical Systems," in IEEE International Conference on Industrial Cyber-Physical Systems (ICPS 2018). [Online]. Available: {https://ieeexplore.ieee.org/abstract/document/8387682/}

[12] Neto, Luis and Gonçalves, Gil, "Component Models for Embedded Systems in Industrial Cyber-Physical Systems," in The International Symposium on Intelligent Manufacturing Environments InManEnv 2018, [retrieved: 6, 2019]. [Online]. Available: {https://www.researchgate.net/profile/Luis_Neto8/publication/326693373_Component_Models_for_Embedded_Systems_in_Industrial_Cyber-Physical_Systems.pdf}

[13] V. Vyatkin, "Software engineering in industrial automation: State-of-the-art review," IEEE Transactions on Industrial Informatics, vol. 9, no. 3, 2013, pp. 1234–1249.

[14] F. e. a. Zezulka, "Communication systems for industry 4.0 and the iiot," IFAC-PapersOnLine, vol. 51, no. 6, 2018, pp. 150–155.

[15] X. Ye, T. Y. Park, S. H. Hong, Y. Ding, and A. Xu, "Implementation of a production-control system using integrated automation ml and opc ua," in 2018 Workshop on Metrology for Industry 4.0 and IoT. IEEE, 2018, pp. 1–6.

[16] H. Bedenbender, A. Bentkus, U. Epple, and T. Hadlich, "Industrie 4.0 plug-and-produce for adaptable factories: Example use case definition," Models, and Implementation, 2017, pp. 56–62.

[17] S.-H. Leitner and W. Mahnke, "Opc ua–service-oriented architecture for industrial applications," ABB Corporate Research Center, vol. 48, 2006, pp. 61–66.

[18] J. Imtiaz and J. Jasperneite, "Common automation protocol architecture and real-time interface (capri)," in Kommunikation unter Echtzeitbedingungen. Springer, 2013, pp. 79–88.

[19] L. Rauchhaupt, G. Kadel et al., "Network-based communication for industrie 4.0-proposal for an administration shell," Federal Ministry for Economic Affairs and Energy (BMWi), 2013.