Semantic Web of Things for Industry 4.0

Aparna Saisree Thuluva[1,2], Darko Anicic[1], Sebastian Rudolph[2]

[1]Siemens AG - Corporate Technology, [2]TU Dresden, Germany.
aparna.thuluva@siemens.com, darko.anicic@siemens.com,
sebastian.rudolph@tu-dresden.de

Abstract. Industry 4.0 which is also referred to as the fourth industrial revolution aims at mass customized production with low-cost and shorter production life-cycle, by digitalizing and automating the manufacturing process. The Automation Systems (AS) used in the manufacturing process should be flexible in order to achieve this goal. But, lack of interoperability between AS devices from different domains makes it harder to achieve this goal. In this study, we employ Web of Things and Semantic Web Technologies to address the cross-domain interoperability problems in AS. Then, we propose an approach to engineer and configure an AS with minimum effort; and show the preliminary results as a demonstration on a process automation workstation to show the feasibility of our approach.

1 Introduction

Mass customized production faces the challenges of higher production costs and longer production life-cycle. The fourth industrial revolution also known as Industry 4.0 [1] aims at reducing the cost and production time for mass customized production. Manufacturing processes are being digitalized and automated by employing Automation Systems (AS) for manufacturing processes. A goal in a manufacturing process is achieved by letting the AS devices communicate, co-operate and exchange information in a certain manner.[1] Mass customized production requires ever-changing settings such as dynamic configuration of interactions between AS devices and constant update of functionalities on an AS, in order to manufacture products to fulfill the demands of every customer. Therefore, to meet the ever-changing settings the AS should be flexible. But, if the AS devices used in manufacturing belong to different domains and they use different information models and protocols such as OPC UA[2], BACnet[3] and so on, that restrains the interoperability between the devices which in turn makes it harder to achieve flexibility in an AS. Therefore, interoperability problems between the AS belonging to different domains should be addressed.

[1] https://www.plattform-i40.de/I40/Redaktion/EN/Downloads/Publikation/interaction-model-I40-components.pdf
[2] https://opcfoundation.org/about/opc-technologies/opc-ua/
[3] http://www.bacnet.org/

Today, the AS used in manufacturing are increasingly becoming part of Internet of Things (IoT). IoT is the system where the physical devices are embedded into electronic systems which can connect to the internet and can be discovered, monitored, controlled and interacted with each other over various network interfaces. But, IoT lacks an universal application protocol which hinders the integration of devices from various manufacturers into a single application [2]. Web of Things (WoT) [3] addresses this limitation by leveraging the Web standards to the embedded devices, which by addressing the interoperability problem, enables the devices from various manufacturers to be integrated with the Web applications with minimal effort. Employing Semantic Web Technologies (SWT) also enables cross-domain interoperability. Semantic modeling of the WoT-enabled devices, their services and applications provide un-ambiguous and machine-readable device descriptions and also creates interoperability between the devices and their services across domains. Therefore, in this project we propose an approach to engineer and configure an AS with low-effort and minimum human intervention by employing SWT on WoT-enabled devices.

The main contribution of this project is to: create transparency of AS infrastructure, develop a low-effort approach to engineer and configure the interactions between AS devices in order to fulfill a goal specified by a manufacturing task. We also develop a low-effort approach to re-engineer the AS devices to update their functionality, whenever it is required by a manufacturing process. We will make this semantic-based approach easily usable for an AS engineer with little knowledge of the SWT.

2 State of the Art

Web of Things: Initial standards are being developed for the Web of Things by the W3C WoT working group[4] to address the concerns regarding interoperability between IoT platforms. The standard provides an interface for a physical thing called Thing Description[5] (TD). TD describes a Thing in terms of its interactions and meta-data.

Industry 4.0: The Reference Architectural Model [4] (RAMI 4.0) is an initial standard developed for Industry 4.0. It proposes that all the Industry 4.0 components need common standards for communication and a standardized language for exchange of information. RAMI 4.0 proposes that every Thing should have its own Administrative Shell [4] which stores all the important data of a physical thing or a software in digital format in a standardized language. TD can be used as a basis to model the Administrative Shell of a Thing, as it provides common standard for communication and enhances inter-operability between the things in Industry 4.0.

[4] https://www.w3.org/WoT/WG/
[5] https://w3c.github.io/wot-thing-description/

Semantic Web Technologies (SWT): provides standardized knowledge representation formalisms such as RDF[6], RDF Schema[7] and OWL[8] and query language over the semantic data, called SPARQL[9]. There exists a number of semantic models for WoT which provide domain-dependent and domain - independent vocabularies. Few of them are: (1) Semantic Sensor Networks ontology (SSN) [5] is an OWL 2 ontology which models the sensors, actuators and their contextual information. (2) eCl@assOWL [6] is an OWL ontology that models industrial products and services. It is created based on the eCl@ss[10] cross-industry data standard. Adoption of SWT has been a recent development in the industrial domain [7, 8]. An important application of these technologies has been the formalization of information models and physical devices using ontologies providing interoperability, un-ambiguous and machine-readable descriptions.

The **state-of-the-art engineering** of AS devices is typically done using Software Engineering approach based on model-driven design. There are five phases in the engineering of an embedded AS device. They are: 1. Design phase 2. Development phase 3. Engineering phase 4. Commissioning phase and 5. Operating phase. According to the model-driven design, an engineer specifies a field function or a data point in a model in the Design phase. The code generation is run to produce a skeleton of a service that is supposed to implement the function or data point in the Development phase. Finally, in the Engineering phase, the engineer implements the service skeleton, deploys it in a service run-time (that is embedded in a device) in the Commissioning phase and starts it. The current way of engineering of embedded AS devices is time consuming and demands a lot of human effort for implementation of the application and deployment. This method is not suitable for ever-changing nature of mass customized production. In contrast to this method, we propose a low-effort engineering method which involves minimum human intervention in Section 4.

3 Research Challenges and Goal

We have set the following **research goals** to achieve flexibility in an AS for mass customized production:

- Facilitate an AS infrastructure to be transparent in order to enable rapid application development.
- Develop an end-to-end engineering approach to **establish and configure** the interactions between the AS devices and **re-engineer** an AS to install new functionality on it with **low-effort and minimum human intervention**.
- Make the proposed semantic-based engineering approach easily usable for an AS engineer with little knowledge of SWT.

[6] https://www.w3.org/RDF/
[7] https://www.w3.org/TR/rdf-schema/
[8] https://www.w3.org/OWL/
[9] https://www.w3.org/TR/rdf-sparql-query/
[10] https://www.eclass.eu/

We identified the following **research challenges** that should be addressed in order to achieve the goals mentioned above:

– How to make an AS infrastructure transparent for rapid application development?
– The AS are complex, there exists large number of functionalities and interconnections on an AS which should be taken into consideration during the engineering process. How to **discover the functionalities** on a complex AS?
– How to **engineer**, **configure** and **re-engineer** an AS with **low-effort, minimum human intervention**?
– We believe that SWT can tackle the above challenges. But, to make the proposed semantic-based approach usable for AS engineers with little knowledge of SWT is a challenge by itself.

The next section presents our approach to achieve the above mentioned research goals by addressing these challenges.

4 Methodology

In this project, we develop a semantic-based approach for engineering and configuring a WoT-enabled AS. Employing SWT in Web of Things is called SWoT [9]. In this approach, an AS device is embedded into an electornic system which can connect to the Web and interact with other devices using existing Web standards[11] [10–12]. The following paragraphs present our approach ib WoT-enabled AS:

Traperency of the AS infrasturture: We model the WoT-enabled AS, their services and applications semantically using W3C WoT Thing Description (TD). A TD is semantically annotated by re-using the existing domain-independent and domain-dependent models to enable cross-domain inter- operability [13] between the Things and their services. The TD of a device is stored on the device itself . Enriching the device with its semantic description, rules [14, 15] and semantic reasoning techniques makes an AS infrastructure trasperent for rapid application development and equips the device with **decision-making support**. This is a key feature for **engineering with minimum human intervention** in our approach. As it enables local discovery [7, 16, 17] of functionalities on an AS protecting its data ownership and also enables compatibility check between the AS devices to be done in an automated fashion, during the commissioning phase of the engineering process.

Low-effort Engineering: In a manufacturing process a task is usually performed by letting the AS devices interact in a certain manner. It would be beneficial to store this composition of interactions as a template for a later use; In such a way that, a template can be effectively discovered and extended with other templates. In order to achieve this, we have come up with a concept of

[11] http://mqtt.org/documentation

"Template based engineering" where interactions between certain devices are stored as an Engineering Template (ET). We develop a light-weight semantic model to describe an ET, which can be stored and discovered from a semantic repository as shown in Figure 1. In order to engineer an AS, an engineer discovers an ET from the repository and instantiates it with the matching devices on the AS, then the interactions between the devices is established by implementing and configuring the interactions using scripts and existing Web standard methods.

Low-effort Re-engineering: In the mass customized production, the devices need to constantly update their functionality to meet the customer demands. In order to update new functionality easily, we have come up with a concept called Apps. An App is a semantic model which defines an added-value functionality. Apps are stored in the semantic repository as shown in Figure 1, which enables discovery and re-use of the Apps. An App model describes the functionality of the App; and requirements that should be fulfilled by a device to install the App. The App and the device semantic models enable automated compatibility check in the commissioning phase, to ensure that the device has the capability to run the App. Therefore, it minimizes the human intervention during re-engineering. Moreover, the App model is directly run-able on the device using device run-time, which minimizes the effort during the engineering process.

Easy-to-use Engineering Tool: Our semantic-based approach for end-to-end engineering could be complex to use for an AS engineer with little knowledge of SWT. Therefore, in oder to overcome this limitation, we develop an graphical tool for our engineering approach as shown in Figure 1. The tool should support an AS engineer in all phases of engineering. It should have the ability to guide an engineer to model TDs, ETs and Apps; and do **semantic-based discovery** on the semantic repository to discover the Apps, ETs and TDs and download them to the tool (see Figure 1). The tool should have the ability to communicate with an AS through a RESTful API to discover the functionalities on AS, install Apps and scripts on a device; or instantiate, implement and configure an ET to engineer an AS.

4.1 Basic Building Blocks

The devices under our consideration are resource-constrained devices which have limited memory and processing power, PC-based semantic reasoning techniques are not feasible on these devices [18]. Thus, in this project we use **an embedded micro reasoner** which can run on the resource-constrained devices (see Figure 1) with Unix / Linux platforms. The micro reasoner consists of two parts: **a micro event processing engine** implemented in C, which is based on the work from [19]. Event rules are used to do Complex Event Processing (CEP) of the events from the AS devices. The rules can be added/deleted easily to the engine over a RESTful API. The event rules and scripts can be directly deployed on the devices from the engineering tool as shown in Figure 1. The second part is **a datalog reasoner**[12] which provides datalog reasoning on the device. The micro reasoner is embedded in the edge device which in our case is SIMATIC IOT2040[13]. An edge device is embedded on an AS which acts as a gateway between the AS and the cloud. In some cases, the micro-reasoner can also run on the Class 2 resource-constrained devices [20] as shown in Figure 1.

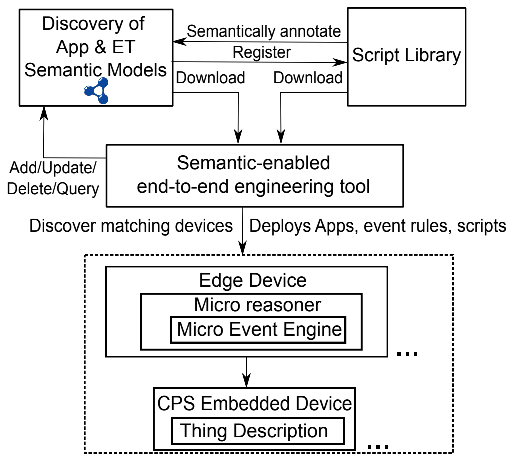

Fig. 1: The System Architecture with basic building block components

5 Preliminary Results

We performed a feasibility test of our methodology presented above for the low-effort re-engineering of an AS device to install new Apps on an AS with minimum effort.

Demo setup: The implementation is demonstrated on the FESTO[14] process automation workstation shown in Figure 2. The workstation is equipped with a few edge devices[15] with micro reasoner running on them. All the sensors and the actuators on the workstation are embedded with NodeMCUs[16] and connected to the edge devices. There is a binary float sensor on Tank 1, which detects overflow in the tank. There exists a pneumatic valve which takes Boolean value as input and turns on the valve to pump out the liquid from Tank 1. The workstation is

[12] http://www.ccs.neu.edu/home/ramsdell/tools/datalog/datalog.html
[13] http://docs-europe.electrocomponents.com/webdocs/1536/0900766b815365c3.pdf
[14] http://www.festo-didactic.com/int-en/learning-systems/process-automation/compact-workstation/mps-pa-compact-workstation-with-level,flow-rate,pressure-and-temperature-controlled-systems.htm
[15] http://docs-europe.electrocomponents.com/webdocs/1536/0900766b815365c3.pdf
[16] https://nodemcu.readthedocs.io/en/master/

engineered such that when overflow occurs in Tank 1 then the pneumatic valve will turn on to ensure overflow protection on Tank 1.

Use case: Our use-case was to ensure overflow protection on Tank 1. Imagine a situation where the float sensor is malfunctioning then, the whole process will be disturbed. In order to avoid the machine downtime, we re-engineered the AS by installing an App on it which uses the liquid level values from the ultrasonic sensor (which measures the level of liquid in the tank) on Tank 1 as shown in Figure 2 to detect the overflow of the tank.

Demo steps: We implemented the proposed semantic-based re-engineering approach and corresponding features in the engineering tool. The tool provides an interface to the semantic repository to discover Apps, TDs and download them to the tool. The tool connects to the edge devices on the AS (through a RESTful API), and does distributed discovery for the devices and automated compatibility check between the App and the device, locally on the edge device. We installed an App on the discovered edge device which fulfills the App requirements. The App reads the liquid level values from the ultrasonic sensor and detects the tank overflow. Thus, the AS is re-engineered with low-effort.

Result: We measured the time taken for distributed discovery and automated compatibility check. We tested the integration of engineering tool with semantic repository and its interaction with the AS. It proved that our approach is feasible to be applied on real-world use cases.

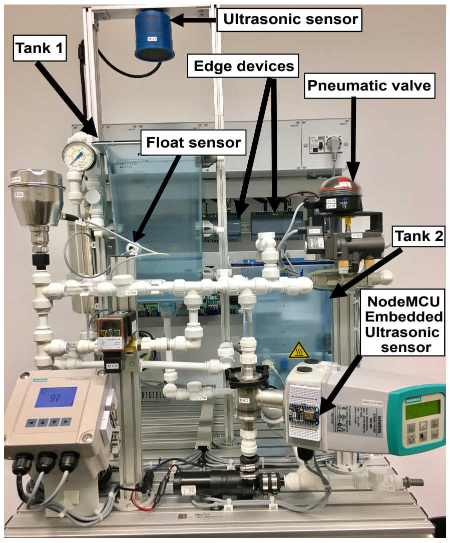

Fig. 2: Festo MPS Process Automation Workstation

6 Use cases and Evaluation Plan

We demonstrate the engineering of an AS using ETs and re-engineering of an AS device on the following use cases:

- Plug and Play use case: When a new device is introduced into the workstation, we will demonstrate how our approach can be used to re-configure the an AS easily using engineering ETs with minimum human effort, to adopt the new device into the system. After the feasibility test, we will also evaluate our approach on a large-scale industrial manufacturing unit.
- Re-engineering use case: In addition to the demo on the FESTO workstation, we will demonstrate our approach for low-effort re-engineering on a large-scale industrial manufacturing unit.

We will perform quantitative analysis to check the (1) time taken for distributed discovery and automated compatibility check to find matching devices, (2) time taken to implement and configure an ET. We will perform a qualitative analysis as follows: we will invite the engineers of the AS with little knowledge of SWT to test the engineering tool: We will give the engineers a task to engineer an AS. (1) measure the time taken by them to do the engineering, (2) check the quality of their engineering using our approach, (3) give a questionnaire to the AS engineers and get their feedback regarding the pros and cons of the tool.

7 Conclusion and Future Work

In this project, we proposed a semantic-based approach for low-effort engineering and configuration and re-engineering of a WoT-enabled AS with minimum human intervention. This approach makes an AS flexible for mass customized production. We demonstrated our implementation of re-engineering approach on a FESTO process automation workstation to test the feasibility of our approach. The following are the **future steps** in this project: (1) as a first step, we develop an approach of engineering the AS devices with ETs, do distributed automated compatibility checks between the AS devices, instantiate an ET, implement and configure the interactions between the AS devices with minimum effort. (2) second, we work further on the engineering tool to design, engineer and implement ETs. (3) at last we do quantitative and qualitative analysis of our approach according the evaluation plan mentioned in Section 6.

Bibliography

[1] Norbury, A.: Industry 4.0- vision to reality

[2] Dominique, G., Vlad, T., Friedemann, M., Erik, W.: 5. In: From the Internet of Things to the Web of Things: Resource Oriented Architecture and Best Practices. Springer, New York Dordrecht Heidelberg London (2011) 97–129

[3] Dominique, G., Vlad, T.: Towards the web of things: Web mashups for embedded devices. In: Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences), Madrid, Spain (April 2009)

[4] Adolphs, P., Bedenbender, H., Dirzus, D., Ehlich, M., Epple, U., Hankel, M., Heidel, R., Hoffmeister, M., Huhle, H., Krächer, B., Koziolek, H., Pichler, R., Pollmeier, S., Schewe, F., Walter, A., Waser, B., Wollschlaeger, M.: Reference architecture model industrie 4.0 (rami4.0). 32 pp

[5] Compton, M., Barnaghi, P., Bermudez, L., Garca-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., Huang, V., Janowicz, K., Kelsey, W.D., Phuoc, D.L., Lefort, L., Leggieri, M., Neuhaus, H., Nikolov, A., Page, K., Passant, A., Sheth, A., Taylor, K.: The ssn ontology of the w3c semantic sensor network incubator group. Web Semantics: Science, Services and Agents on the World Wide Web **17**(0) (2012)

[6] Hepp, M.: eclassowl: A fully-fledged products and services ontology in owl. In: In: Poster Proceedings of ISWC, Galway. (2005)

[7] Grangel-González, I., Halilaj, L., Coskun, G., Auer, S., Collarana, D., Hoffmeister, M.: Towards a semantic administrative shell for industry 4.0 components. CoRR **abs/1601.01556** (2016)

[8] Kharlamov, E., Grau, B.C., Jiménez-Ruiz, E., Lamparter, S., Mehdi, G., Ringsquandl, M., Nenov, Y., Grimm, S., Roshchin, M., Horrocks, I.: Capturing industrial information models with ontologies and constraints. In: The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part II. (2016) 325–343

[9] Gyrard, A., Bonnet, C., Boudaoud, K.: Domain knowledge interoperability to build the semantic web of things. In: In W3C Workshop on the Web of Things June 25. (2014)

[10] Fette, I., Melnikov, A.: The websocket protocol, RFC 6455. (2011)

[11] Belshe, M., Peon, R., Thomson, M., Melnikov, A.: Hypertext transfer protocol version 2.0. internet draft,. (2013)

[12] Kovatsch, M., Duquennoy, S., Dunkels, A.: A low-power coap for contiki. In: In Proceedings of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems. (2011)

[13] Maarala, A.I., Su, X., Riekki, J.: Semantic reasoning for context-aware internet of things applications

[14] Motik, B., Sattler, U., Studer, R.: Query answering for OWL DL with rules. Journal of Web Semantics **3** (2005) 41–60

[15] Krötzsch, M., Rudolph, S., Hitzler, P.: Description Logic Rules. In: ECAI. (2008)

[16] Wahlster, W.: Semantic technologies for mass customization. In: In Towards the Internet of Services: The THESEUS Research Program. Springer (2014) 3–13

[17] Thoma, M., Braun, T., Magerkurth, C., Antonescu, A.: Managing things and services with semantics: A survey. In: In Network Operations and Management Symposium (NOMS). (2014) 1–5

[18] Christian, S., René, S.: Rule-based OWL reasoning for specific embedded devices. In: The Semantic Web – ISWC 2011. Springer Berlin Heidelberg (2011) 237–252

[19] Anicic, D., Rudolph, S., Fodor, P., Stojanovic, N.: Stream reasoning and complex event processing in etalis. Semantic Web **3** (2012) 397–407

[20] Bormann, C., Ersue, Mand Keranen, A.: Terminology for constrained-node networks. RFC 7228 (2014)