KI - Künstliche Intelligenz (2019) 33:131–140
https://doi.org/10.1007/s13218-019-00579-0

TECHNICAL CONTRIBUTION

A Jumpstart Framework for Semantically Enhanced OPC-UA

Badarinath Katti[1] · Christiane Plociennik[2] · Michael Schweitzer[1]

Received: 15 August 2018 / Accepted: 23 March 2019 / Published online: 3 April 2019

Abstract
Decentralization is the norm of future smart production as it assists in contextual dynamic decision-making and thereby increases the flexibility required to produce highly customized products. When manufacturing business software is operated as a cloud based solution, it experiences network latency and connectivity issues. To overcome these problems, the production control should be delegated to the manufacturing edge layer and hence, the argument of decentralization is even more applicable to this narrative. In order to accomplish the assigned manufacturing task effectively, the edge layer is required to possess contextual awareness to make run-time decisions in production. Semantic technologies, on the other hand, assist in discerning the meaning, reasoning and drawing inferences from the data. There are several specifications and frameworks to automate the discovery, orchestration and invocation of web services; the prominent are OWL-S, SAWSDL and WSMO. This paper derives a hybrid approach that integrates OWL-S and SAWSDL specifications to overcome the downsides, yet retain the benefits of both approaches to the OPC-UA application methods. Consequently, the proposed semantically enriched OPC-UA concept enables the edge layer to create flexible production orchestration plans in a manufacturing scenario controlled by cloud MES. Furthermore, the derived hybrid approach is applied to a real use case to demonstrate its feasibility in industrial environments.

Keywords Cloud MES · GeSCo · Edge Manufacturing · OPC-UA · Semantic Web Services · SAWSDL · OWL-S

1 Introduction

This research work continues in the direction of enriching the OPC-UA with formal semantics. Semantic OPC-UA builds upon the architecture that was proposed in [9]. The architecture integrated the Edge component Generic Shop-Floor Connector (*GeSCo*) in the context of Cloud MES (CMES) controlled manufacturing. GeSCo tackles the challenges of connectivity and network latency by caching the production control data (see Fig. 1). The GeSCo, which is a production control delegate from CMES, coordinates with various entities of the shop-floor. The objective of GeSCo is to reduce the communication with CMES by taking runtime decisions based on the production order (PO) requirements.

In order to make runtime decisions, the control layer in the shop-floor should possess reasoning capabilites. However, the OPC-UA, which is a standard intersection of IT and automation, does not possess reasoning capabilities that allow to make numerical and logical calculations that consequently assist in design of consistency-check rules, and derive logical inferencing. This *intelligence* is required to reason on the services provided by manufacturing resources, assess the resource conditions for better coordination in the production and evaluate the pre- and post-conditions of an OPC-UA method execution in the PO orchestration. Figure 2 provides an example of above explanation where the OPC-UA with its current capabilities can only answer questions with green tick bullets, but not crossed in red. Such conditions that not only involve connecting the mere references, but also involve logical and reasoning expressions that need to be represented in the information model at design time,

This research work is supported by a doctoral grant from SAP SE. A non-provisional patent, with ID 15/926,856, on the research of CMES controlled edge manufacturing has been filed in USA on 20-Mar-2018 with the authors as the inventors.

✉ Badarinath Katti
katti@rhrk.uni-kl.de; badarinath.katti@sap.com

Christiane Plociennik
christiane.plociennik@dfki.de

Michael Schweitzer
michael.schweitzer@sap.com

[1] SAP SE, Walldorf, Germany

[2] DFKI GmbH, Kaiserslautern, Germany

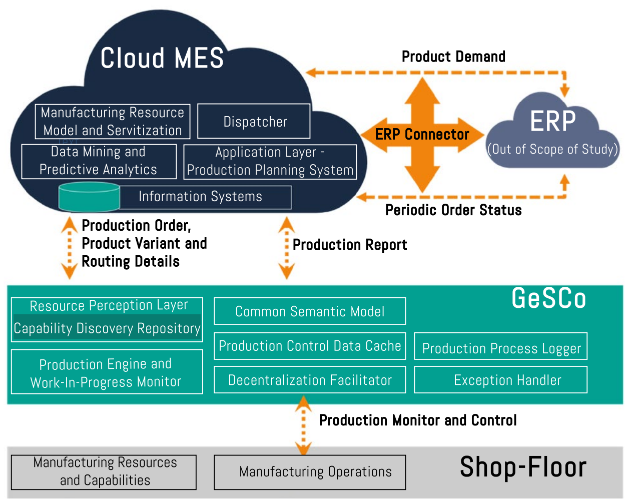
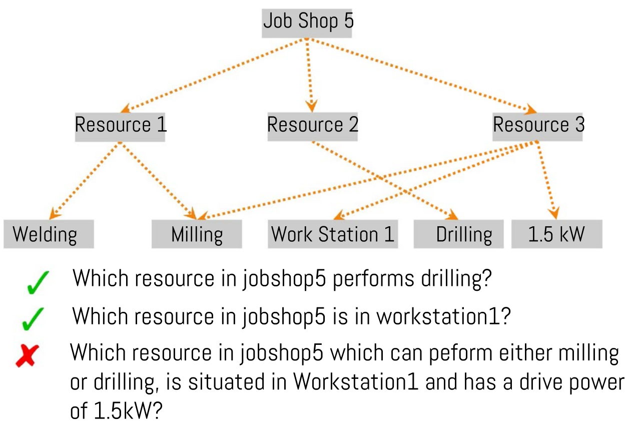

Cloud MES
Manufacturing Resource Model and Servitization
Dispatcher
Data Mining and Predictive Analytics
Application Layer - Production Planning System
Information Systems
Product Demand
ERP Connector
ERP
(Out of Scope of Study)
Periodic Order Status
Production Order, Product Variant and Routing Details
Production Report
GeSCo
Resource Perception Layer
Capability Discovery Repository
Common Semantic Model
Production Control Data Cache
Production Process Logger
Production Engine and Work-In-Progress Monitor
Decentralization Facilitator
Exception Handler
Production Monitor and Control
Manufacturing Resources and Capabilities
Manufacturing Operations
Shop-Floor

Fig. 1 Integration of GeSCo with CMES [9]



✓ Which resource in jobshop5 performs drilling?

✓ Which resource in jobshop5 is in workstation1?

✗ Which resource in jobshop5 which can peform either milling or drilling, is situated in Workstation1 and has a drive power of 1.5kW?

Fig. 2 Illustration of logical reasoning incompetency of OPC-UA

and processed at runtime to assess contextual information at factory shop-floor, and it is not possible with the current OPC-UA specification.

The authors attempt to overcome this insufficiency of OPC-UA by augmenting it with a reasoning engine based on description logics. Additionally, Service Oriented Architecture (SOA) provided by OPC-UA is decoupled from actual implementation details in order that the vendors can choose a communication protocol of their preference. Hence, an effort is made to develop a semantic solution that is independent of the underlying communication protocol details.

The outline of the paper is as follows: Sect. 2 points out the related work, and Sect. 3 provides cursory information regarding OWL-S and SAWSDL specifications. A hybrid of the OWL-S and the SAWSDL specifications is introduced in Sect. 4, which is followed by design and implementation details of an industrial use case in Sects. 5 and 6. Section 7 lists the lessons learned in practice and Sect. 8 concludes.

2 Related Work

With an intention to share the data across the applications and enterprise boundaries, the concept of the semantic web was coined in [3] and consequently, the idea of semantic web was extended to Semantic Web Services (SWS) in order to enhance the utility of the whole concept. SWS combines different technological concepts such as web services, semantic web, and automated logical reasoning. Enriched by semantics that are capable of being processed by machines, SWS gained instant traction since they could efficiently exploit the services on the web without human intervention. To date, a huge volume of literature has been published on the subject of applying SWS in the manufacturing domain. The concept of introducing ontologies in manufacturing as the state of the art was reviewed in [4]. The facilitation of dynamic orchestration of operational processes in the shop floor by SWS is the reason for its widespread adoption in factory automation. There are also several research papers, for example, [8] that focus on purely syntactical level orchestration which are suitable only for static workflow requirements. However, these techniques fail where adaptive process planning is the key requirement of the production. The idea of replacing the low-level programming of sensors and actuators with the high-level programming of the manufacturing resources with the application of SWS was recognized in [13]. [5] shows that a high degree of customization and reconfiguration of the system is possible through ontology-based web services. There are many SWS standards such as OWL-S [1], WSMO [6], SAWSDL [12], SWSF [2] et cetera that aim to describe the semantics of web services based on standard ontologies. These semantics are referred during intelligent service discovery, orchestration, and invocation.

However, as per the current standards, the communication in factory automation should take place via OPC-UA. The OPC-UA standard allows server methods that enable Service Oriented Architecture (SOA) at machine level. [17] enhanced abstract services covered in part 4 of the OPC-UA specification with semantics for discovery of servers based on location. However, they constrain themselves to service discovery and do not cover the decision-making process. Moreover, they used the WSMO framework and applied it in the field of smart energy grids.

Following the introduction of the prominent and the industry-neutral OWL-S and SAWSDL frameworks to OPC-UA specification, known as SA-OPC-UA [11] and SemOPC-UA [10] respectively, this paper introduces a composition of these schemes to reap the advantages of both the approaches to enable true factory automation. It also provides rationale behind the conception of such a hybrid approach. The simplicity and easy implementation of the SAWSDL concepts, coupled with the general-purpose representation framework of OWL-S make this technique irreproachable and attractive to industry adoption.

3 Technical Background

With the objective to address the difficulties that stem from differences in the meaning and usage of manufacturing vocabulary, and subsequent inability to express semantic information about manufacturing capabilities offered by the resources, a scientific methodology is formulated that serves the following purposes:

– Consistent representation of domain knowledge.
– Description of each capability, resource, and state of the resource, PO and manufacturing operation in the shop-floor by exploiting this domain knowledge to design a well-formed ontology.
– Exploitation of description logics in the ontology to formulate the complex semantic rulesets using off-the-shelf semantic language axioms and custom conditions using the semantic web rule languages in order to capture the contextual information at each step of the production.
– Creation of the semantically augmented OPC-UA framework that fosters the application of knowledge in production which in turn assists in the dynamic decision-making process.

The semantically augmented OPC-UA framework which is the focus of this paper requires the knowledge of SAWSDL and OWL-S semantic service frameworks. The following sub-sections discuss briefly the concepts of these frameworks. Later, these concepts are used as basis for introducing the research topic of hybrid of OWL-S and SAWSDL specifications.

3.1 SAWSDL

The web service description language (WSDL) [18] is a W3C recommendation which provides a formal, syntactic and machine readable description of SOAP based web services. However, there is a semantic gap between the syntactic description of a web service and its underlying meaning. Semantic Annotations for WSDL (SAWSDL) [12] is an incremental bottom-up mechanism of modeling SWS and is a W3C recommendation. It is a mechanism where elements of WSDL are decorated with extensible attributes to attach semantic annotations (see Fig. 3 for an example).

```
<wsdl:operation name="Welding"
  sawsdl:modelReference="http://opcua-sawsdl.poc.de/#WeldingMethod">
  <wsdl:input element="CoOrdinates"
  sawsdl:modelReference="http://opcua-sawsdl.poc.de/#PlanarWeldingParams"
  sawsdl:loweringSchemaMapping="http://WDFN32202381A/CoOrdOnt2CoOrd.xslt"
  sawsdl:liftingSchemaMapping="http://WDFN32202381A/Ack2StatusOnt.xml"/>
  <wsdl:output element="Acknowledgement"
  sawsdl:modelReference="http://opcua-sawsdl.poc.de/#operationStatus"/>
</wsdl:operation>
```

Fig. 3 SAWSDL annotations on a web service operation and its parameters

3.2 OWL-S

OWL-S [1], a W3C recommendation, is an OWL-based ontology framework of the semantic web to describe SWS. It enables an agent-based framework to discover, orchestrate and invoke the SWS. It consists of three main sub-ontologies, namely service profile, process model and service grounding. The service profile advertises the service functionality, the process model provides a detailed description of the service and the service grounding provides concrete details to communicate with the service instance.

4 Hybrid of OWL-S and SAWSDL

Section 4.1 puts forward the arguments for and against both the OWL-S and the SAWSDL specifications with regards to the software quality characteristics such as usability, efficiency and maintainability. Consequently, it draws the conclusion that a hybrid approach is more feasible that derives the benefits and at the same time, precludes the shortcomings of both approaches. Section 4.2 describes the hybrid methodology.

4.1 Motivation

The following Table 1 lists the attributes with regard to both functional and non-functional properties and compares how the OWL-S framework and SAWSDL specification fare against these attributes.

Considering the above critique of both OWL-S and SAWSDL specifications consisting of benefits (+), neutral arguments (±) and shortcomings (–), the best practice is to preserve the general-purpose ontology representation framework, and yet retain maximal information in the server i.e., the manufacturing resource, and publish only nominal information to the GeSCo to assist in method discovery.

Table 1 Empirical comparison of OWL-S and SAWSDL specifications

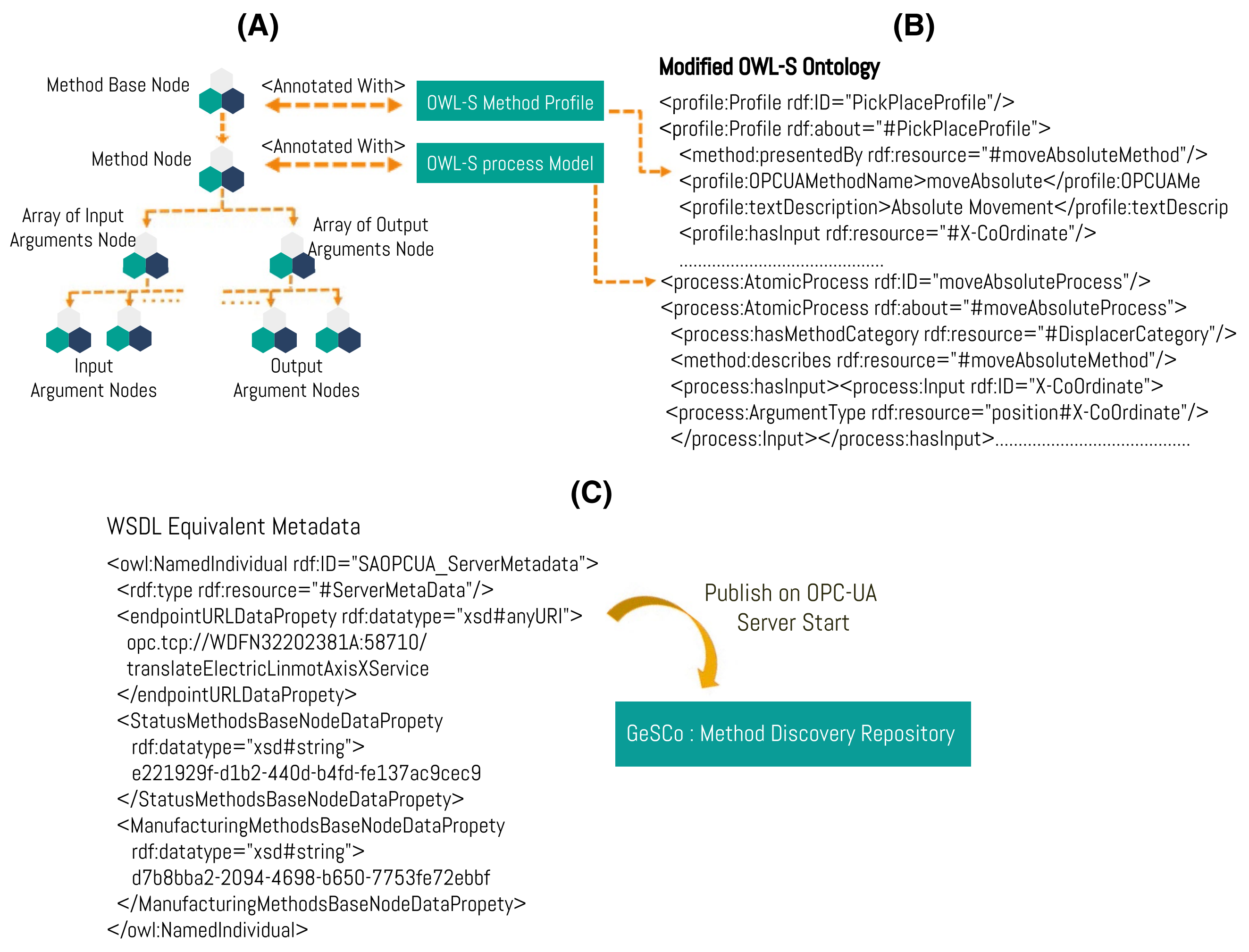
	OWL-S		SAWSDL	
	Remarks	Rating	Remarks	Rating
Simplicity and implementation	No, more effort	–	Yes, less effort	+
Industry adoption	Relatively difficult to pursue	–	Smooth	+
Framework prescription	Yes	+	No	–
Restriction of ontology language	Yes	±	No	±
Strict guidelines for programming	Yes	±	No	±
Standardized communication interfaces and protocols	Supports standardization	+	Possibility of creation of ad-hoc semantics structures	–
Homogeneous ontology	Yes	+	Multiple ontology languages can be used for the same server annotations	–
Arbitrary usage	No	+	Subjected to arbitrary usage	–
Volume of published server metadata	More data	–	Less data	+
	Entire OWL-S ontology is published for each of the server method	–	Constant size metadata is sent on per server basis	+
	Storing of huge data in clients	–	Storing of comparatively less data in clients	+
	More burden on production network	–	Less burden on production network	+
	If server cannot preserve address space, the updated ontology has to be published after every reboot	–	Does not apply	±
	Security risk	–	Relatively less security risk	+
Built in support for orchestration	Yes	+	No	–
Sophisticated discovery mechanism	Yes	±	No	±
Matchmaking	Efficient	+	Convoluted mechanism due to absence of a framework	–
Support for OWL	Yes	+	Yes	+

4.2 Methodology

To find a right balance, we propose a procedure which is a heterogeneous composition of SAWSDL concepts and the OWL-S framework. In the OWL-S approach, the complete ontology is developed and the corresponding OWL-S sub-ontology is built parallel to the OPC-UA server. Hence, the OPC-UA server and OWL-S ontology are completely decoupled entities. On start of the OPC-UA server, the accompanying OWL-S ontology is published to the Generic Method Discovery Repository (*GMDR*). The client in pursuit of method discovery only interacts with the published OWL-S file and comes into the contact with the OPC-UA server only during method invocation time. In the SAWSDL approach, the developed custom ontology and OPC-UA are tightly coupled to each other. The OPC-UA server provides compact grounding information that is utilized by the client to locate the application specific methods of the server. In contrast to the OWL-S scheme, the client directly interacts with the semantically annotated OPC-UA server over its complete *discovery—orchestration—invocation* cycle. In case of the hybrid approach, it employs the OWL-S framework to develop the sub-ontologies in order to model the methods. These sub-ontologies are annotated to the corresponding nodes in the OPC-UA server and compact grounding information is published to the RPL of the GeSCo layer (see Fig. 4).

The profile and the process model ontologies of OWL-S semantically describe the capability of a specific OPC-UA method for the purpose of method discovery and the OWL-S grounding ontology semantically describes accessing the method and its argument nodes for the purpose of dynamic method invocation. This paper recommends that the semantic description layer and the execution layer that specify the method capability and the grounding description respectively should be separated for the purpose of achieving the design principle of separation of concerns.

The profile ontology of OWL-S is attached to the method (both status and manufacturing) base nodes. These base nodes provide a high-level description of the capabilities provided by the OPC-UA server as illustrated in Fig. 4a and b. The method base node is a parent node for a collection of OPC-UA methods. Therefore, it has to be ensured that the profile ontology that is attached to the manufacturing (or status) base method node is generically designed to encompass



Modified OWL-S Ontology

```
<profile:Profile rdf:ID="PickPlaceProfile"/>
<profile:Profile rdf:about="#PickPlaceProfile">
  <method:presentedBy rdf:resource="#moveAbsoluteMethod"/>
  <profile:OPCUAMethodName>moveAbsolute</profile:OPCUAMe
  <profile:textDescription>Absolute Movement</profile:textDescrip
  <profile:hasInput rdf:resource="#X-CoOrdinate"/>
  ...........................................
<process:AtomicProcess rdf:ID="moveAbsoluteProcess"/>
<process:AtomicProcess rdf:about="#moveAbsoluteProcess">
  <process:hasMethodCategory rdf:resource="#DisplacerCategory"/>
  <method:describes rdf:resource="#moveAbsoluteMethod"/>
  <process:hasInput><process:Input rdf:ID="X-CoOrdinate">
 <process:ArgumentType rdf:resource="position#X-CoOrdinate"/>
  </process:Input></process:hasInput>.........................................
```

WSDL Equivalent Metadata

```
<owl:NamedIndividual rdf:ID="SAOPCUA_ServerMetadata">
  <rdf:type rdf:resource="#ServerMetaData"/>
  <endpointURLDataPropety rdf:datatype="xsd#anyURI">
   opc.tcp://WDFN32202381A:58710/
   translateElectricLinmotAxisXService
  </endpointURLDataPropety>
  <StatusMethodsBaseNodeDataPropety
    rdf:datatype="xsd#string">
    e221929f-d1b2-440d-b4fd-fe137ac9cec9
  </StatusMethodsBaseNodeDataPropety>
  <ManufacturingMethodsBaseNodeDataPropety
    rdf:datatype="xsd#string">
    d7b8bba2-2094-4698-b650-7753fe72ebbf
  </ManufacturingMethodsBaseNodeDataPropety>
</owl:NamedIndividual>
```

Fig. 4 Hybrid of OWL-S and SAWSDL specifications applied to a method of a pick and place robot for the purpose of Illustration: **a** annotated OPC-UA node structure. **b** Rough approximation of corresponding OWL-S ontology for node annotation. **c** Grounding information to be published to OPC-UA clients

the functionality provided by all the manufacturing (or status) methods.

For further fine grained discovery, the next task is the annotation of the process model ontology to the OPC-UA nodes. There are two possible ways of semantic annotation of the process model ontology. In the first scheme, each of the semantic references of the OWL-S process model ontology corresponding to the method capability, and its input and output arguments should be annotated to the respective OPC-UA nodes. In the second scheme, each of the status and manufacturing method nodes are directly annotated with the entire process model ontologies of the OWL-S framework. The OWL-S ontology does not support the concept of collection of method arguments. Consequently, OPC-UA node that corresponds to the array of input/output arguments cannot be annotated with the semantic concept. Therefore, the latter scheme is preferred over the former. As the whole process model ontology is attached to the application method node of OPC-UA server, the server and client can attach and draw the entire semantic information that describes capability of server method at one place as illustrated in Fig. 4a and b. From OPC-UA server context, the latter scheme allows to do away with the semantic annotation on the input and output arguments of the OPC-UA method. At the same time, it also simplifies the browsing of an OPC-UA server for a client where the browse path reduces by 1 level corresponding to the input and output arguments. This results in significantly less effort from both the OPC-UA client and server viewpoints and hence, is a preferred scheme for semantic annotation.

The organization of OPC-UA application methods as described in [11] provides a standard mechanism to browse the manufacturing and status methods. The browse path from the server root node to a specific method node becomes irrelevant in such a scenario. Instead, the node ID of the Application Methods Base Nodes that act as a starting point to browse the methods down the hierarchy can be supplied as a substitute. The only other requirement is the provision of endpoint URL of the OPC-UA. Hence, the endpoint URL along with the application methods base nodes is published to the GeSCo GMDR as illustrated in Fig. 4c. Such an arrangement allows to do away with the standard grounding ontology of the OWL-S framework.

In a broader sense, OWL-S is the ontological framework and SAWSDL specification is the method of integration of developed ontology to the server for the benefit of intelligent automation. This reasoning justifies the natural consolidation of both the schemes where modified OWL-S constructs are appropriately employed as annotations for OPC-UA server nodes.

5 Method Discovery and Orchestration

The demonstrator system which was originally developed in the RES-COM project [16] (see Fig. 5) that produces the smart key finders was enhanced in order to evaluate the applicability of the proposed hybrid approach. The demonstrator setup contains industrial equipments from varied vendors to constitute a production cell where three individual parts of the key finder, namely, housing cover, housing base and the circuit board, are assembled. The work station has multiple key finder assembly units and a generic purpose pick-and-place robot which makes it convenient for experimenting with adaptability and reusability features of the manufacturing resources (see demonstrator layout in Fig. 6). Besides the provision of rich process variants, the demonstrator also consists of infrastructure for the material flow, raw material warehouses and quality control identification systems totaling nearly 50 field devices.

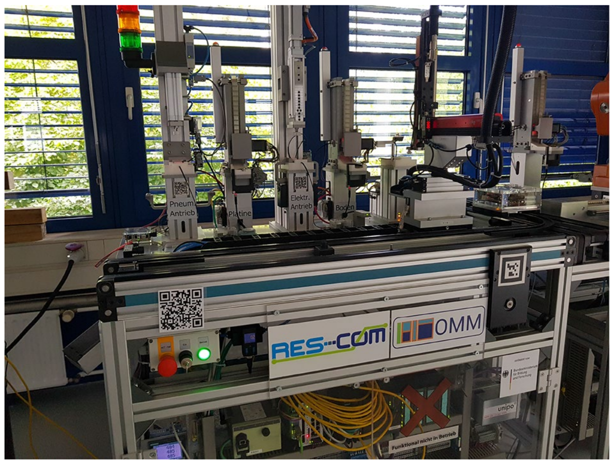


Fig. 5 Automated key finder assembly demonstrator

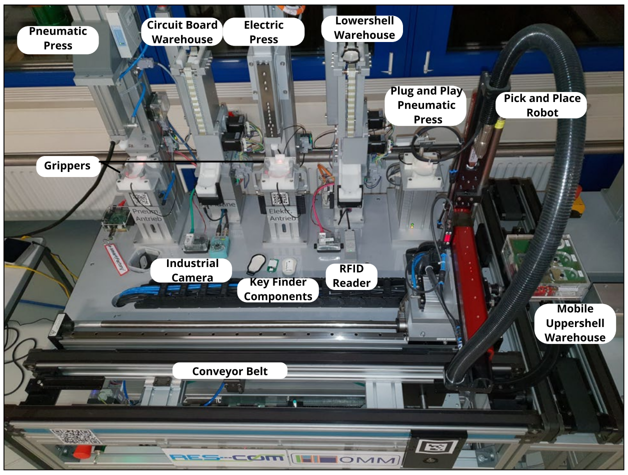


Fig. 6 Layout of the key finder demonstrator

The GeSCo acts as method discovery repository, OPC-UA client and OPC-UA server for the shop-floor communication. The UI of OPC-UA servers of manufacturing resources and GeSCo was designed such that the OWL-S profile and process model sub-ontologies can be annotated to the *Description* fields of the manufacturing and status method base node, and the corresponding method nodes respectively. In order to automate this annotation process, a configuration file was created that mapped the method base nodes and the application methods to their corresponding OWL-S ontologies. During start of the OPC-UA servers, this configuration file was referred in order to hook the semantic annotations to the respective nodes.

The OPC-UA servers of manufacturing resources including GeSCo register themselves by publishing server metadata to the GeSCo RPL when they go online. With the help of published server metadata, the GeSCo, which now acts as OPC-UA client, semantically queries all the application specific methods of registered manufacturing resource in order to discover the necessary methods and stores this information regarding the method functionalities in GMDR. The GMDR stores these functionalities in a key-value pair

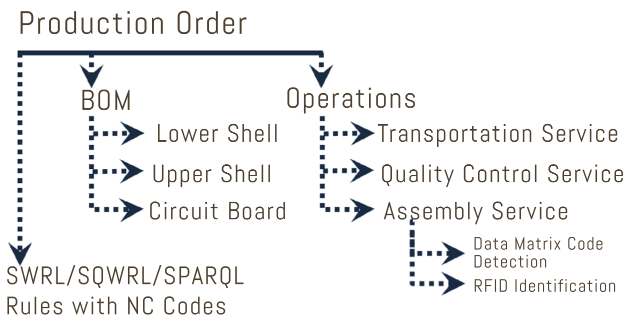


Fig. 7 Contents of PO

collection for the purpose of fast retrieval. The Manufacturing resources also refer to the GMDR in order to find suitable methods to communicate with GeSCo.

The PO for the production of intelligent key finder is generated in the CMES. It is basically an ordered list of abstract requirements to produce the product. Figure 7 briefly describes the PO of the use case at hand. The resources of the key finder demonstrator offer various manufacturing and status methods. The result of the status methods along with the other contextual conditions is taken into account during formulation of the preconditions for manufacturing methods. For the purpose of illustration, Table 2 lists the methods offered by the *press* and *pick and place robot*. PO also contains possible non-conformance codes that might arise in production and a corresponding list of abstract services to resolve the non-conformance issue. When the dispatcher in CMES dispatches the PO to the GeSCo cache, the production engine in GeSCo requests GeSCo decentralization facilitator to find the most suitable manufacturing resource to each of its abstract PO requirements. The decentralization facilitator parses the semantic concepts related to resource methods from its GMDR, and matches the required services of PO to the capabilities offered by resources on the shopfloor with the aid of the centrally accessible ontology service. In our implementation, the decentralization facilitator analyzes GMDR for the manufacturing services that offer the sub-components of the BoM, sub-components transportation service, and quality control and assembly services. When the production engine receives the chosen manufacturing resources, it creates an adaptive orchestration plan. GeSCo also takes into the account the necessary pre- and post-conditions of a method to hold true for continuation of the planned orchestration. In the key-finder use case, the pre-condition for placing the circuit board onto the assembly unit is the detection of its data matrix code. Similarly, the upper shell has to be recognized by the RFID reader before it is transported to the assembly unit. As the OWLAPI [14] is available only for Java stack, a custom Java application is implemented that loads, queries, creates, updates, and saves the ontology. It is also capable of reasoning and adding restriction on the entities of the ontology. At the end of each step, the production engine inspects for possible nonconformance logs against the PO and also invokes the java application to reason about the context of method execution by asserting the class axioms and custom rulesets. The decoupling of manufacturing resources from manufacturing operations during planning in combination with the proposed semantically enriched OPC-UA communication protocol effectively addresses the problems of quality nonconformance and resource breakdowns.

Table 2 Illustration of methods offered by resources

Resource	Manufacturing methods	Status methods
All three	Prepare	isInUse
Press units	Assembly	isInUseEvent
Pick and Place	Home	readActualPosition
Robot for all three degrees of freedom	jogMove	
	jogMoveStop	
	moveAbsolute	
	Stop	

When quality issues are logged against the subassembly at a certain step of production, the production engine again finds appropriate resources to resolve the non-conformance and adapts the orchestration plan accordingly. In principle, the production engine makes no distinction between normal and exceptional situations of the production.

5.1 Use Case 1: Quality Control

A study was carried out where a quality issue was logged with regard to the *assembly* operation with the corresponding non-conformance code. During orchestration plan creation step, GeSCo chooses the electric press as first choice for the final assembly operation of PO because the pick-place-robot can transport all the sub-components of the key finder to the electric press mounting area by covering the smallest distance in relation to other presses (refer to the Fig. 6). GeSCo arrives at this decision based on the evaluation of a SWRL rule for selecting mounting assembly. The algorithm also takes into consideration the distance covered by the pick-and-place robot to perform the quality control operations of the sub-components.

In this simulation, the electric assembly unit is issued a command to *press* after the verification of pre-conditions. The electric assembly does not move down to press the subassembly although it does not generate HTTP 5XX server error. Instead, it returns status code *HTTP 200 OK* which corresponds to successful execution of *press* operation. A manual worker who oversees this production step observes the defective electric assembly unit and logs a corresponding Non-Conformance (NC) code against the PO. However, the production work-flow does not have the option for manual logging of defects. Hence, the production flow was intercepted through an external application and the defect is injected into the production execution. At this point, the production engine realizes the defect in the *assemble* step and it retrieves the abstract services related to the non-conformance code from PO cache required to resolve the quality issue and sends it to the decentralization facilitator to find the relevant manufacturing resources. The GeSCo production engine creates a new orchestration plan containing only those operations relevant to the logged defect. It searches in the GMDR for the equivalent method providers. It reasons that pneumatic press also presents a set of methods that provide the equivalent manufacturing services. Finally, the production step is executed via the pneumatic press.

5.2 Use Case 2: Resource Breakdown

The resource breakdown scenario was also simulated in another PO where the *Pneumatic Press* resource which is part of PO orchestration plan was rendered *unresponsive* with the HTTP 500 internal server error status code. The specific PO was put on hold until a new resource that provided the same manufacturing service was plugged in to the production landscape and its capability metadata is published to the GeSCo RPL. A new pneumatic press resource is plugged in (rightmost press resource in Fig. 5). GeSCo checks in its production engine queue for the pending status/new PO that relies on the newly installed resource, and accordingly changes the orchestration plan.

Thus, it proves that with the aid of this hybrid approach, the edge component can make adaptive orchestration plans at runtime at the shop-floor even in the event of exceptional scenarios.

6 Implementation

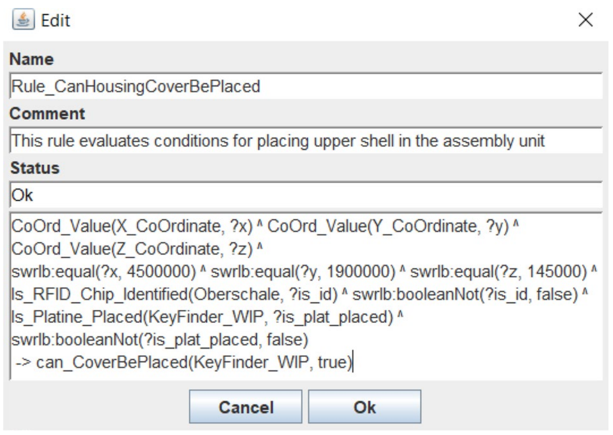
The intelligent key finder assembly industrial use case was developed to realize the Proof-of-Concept where hybrid concepts are integrated into OPC-UA application specific methods. The resources of the key finder work station are augmented with the embedded systems to transform them into cyber physical production systems. The key finder demonstrator modules are controlled via SOAP based web services. OPC-UA server wrappers were created around each of these SOAP based services to make them viable for experimentation of our hybrid approach. The manufacturing and status methods of OPC-UA servers of manufacturing resources were implemented under their respective base nodes as illustrated in [11]. Both GeSCo and manufacturing resources act as both OPC-UA client and server. A mock CMES was implemented that created a PO.

The ontology was modeled in Protégé 5.2.0 [15]. The modeled ontology can be broadly categorized into three submodels, namely domain model, manufacturing resources and capabilities model and model related to service concepts. The ontology was accessible as a static service to all the entities involved in the production. A utility software was implemented that generates metadata file as per the OWL-S specification for a given method of an OPC-UA server. As the research revolved around a single common ontology, the service capabilities have a unique and explicit representation and therefore, no equivalent ontology definition is required. The capabilities of these resources were modeled using the designed common ontology. A full capability OWL API based OWL/XML parser/reasoner engine was tailored to our needs and deployed on all the shop-floor entities to dereference the semantic annotations attached to the server method and its arguments. HermiT reasoner [7] which supports the interfaces and factory classes of OWL API was employed in both external java application and inside Protégé to determine the consistency of the designed ontology and evaluate the derived ruleset. The pre- and post-conditions of a server method were modeled in the OWL/XML syntax in the form of class axioms in OWL and additional rulesets on named individuals of OWL classes in Semantic Web Rule Language (SWRL). The decision making process is entirely localized to the shop-floor as GeSCo draws its conclusions based on the evaluation results of the SWRL ruleset. For example, before the pick-and-place robot picks the upper shell of the key finder, the GeSCo confirms this pre-decided step of the orchestration if the following ruleset shown in Fig. 8 holds true.

GeSCo analyzes the PO requirements, evaluates the method metadata in GMDR and the contextual conditions of the shop-floor with the aid of OWL/XML parser/reasoner and CSM, and dispatches the manufacturing tasks of the orchestration plan to manufacturing resources.

7 Lessons Learned

The extraction of knowledge and its representation is already a difficult proposition. In the manufacturing plant, this challenge is amplified owing to the complexity of the varied nature of resources, work stations, raw materials, part assemblies and products. To add to this complexity, the manufacturing plant evolves continuously during its lifecycle due to installation and operation of new resource, removal of a dilapidated resource and introduction of new product variant. This calls for continuous update of the



Name
Rule_CanHousingCoverBePlaced

Comment
This rule evaluates conditions for placing upper shell in the assembly unit

Status
Ok

```
CoOrd_Value(X_CoOrdinate, ?x) ^ CoOrd_Value(Y_CoOrdinate, ?y) ^
CoOrd_Value(Z_CoOrdinate, ?z) ^
swrlb:equal(?x, 4500000) ^ swrlb:equal(?y, 1900000) ^ swrlb:equal(?z, 145000) ^
Is_RFID_Chip_Identified(Oberschale, ?is_id) ^ swrlb:booleanNot(?is_id, false) ^
Is_Platine_Placed(KeyFinder_WIP, ?is_plat_placed) ^
swrlb:booleanNot(?is_plat_placed, false)
 -> can_CoverBePlaced(KeyFinder_WIP, true)
```

Fig. 8 An instance of SWRL rule to gain contextual knowledge

Table 3 Comparison of UTF-8 encoded data transfer between OPC-UA server and GMDR in SemOPC-UA and hybrid methodologies

	SemOPC-UA	Hybrid
Average size of the published data/method	5250 Bytes	510 Bytes
Assumed number of methods/resource	10	
Average size of the published data/resource	52500 Bytes	510 Bytes

knowledge in the shop-floor. Any knowledge extraction solutions should also be resilient to unscheduled changes of the plant.

The SOAP based web services and subsequently, SWS are deemed to be heavy-weight service descriptions after the advent of REST based web services in the last decade. Hence, SWS could not be widely adopted and consequently, no large scale experimental results are available to ascertain on their scalability on internet level. However, for bounded usage, the semantic description of communication protocols for the purpose of intelligent discovery and invocation is scalable for new requirements, for example inter-enterprise operations. This argument also justifies enriching the OPC-UA with semantics for communication within the manufacturing plant.

Taking into account that this work is a first attempt at enriching OPC-UA with semantics, there is no tool to automatically create a modified OWL-S sub-ontologies template that corresponds to a method in OPC-UA. Nor there is a tool to attach the modeled ontology to the OPC-UA nodes in case of SA-OPC-UA and hybrid schemes. The author only developed utility software that serves the purpose. For industrial adoption, a sophisticated solution is required that is infalliable, mature, and hides the complexity involved in developing semantic descriptions.

Another point of contention for switching to the hybrid approach was the reduction of data traffic that results from publishing the methods metadata by the resources. Table 3 provides an estimate of UTF-8 encoded data transferred from OPC-UA server to the GMDR for both the approaches. The SemOPC-UA server sends data that varies directly with deeper node browse paths and number of supported methods. The SemOPC-UA approach adds the additional burden on the network bandwidth considering the metadata publishing of all the OPC-UA servers of the shop-floor. The hybrid approach on the other hand publishes constant size data to the GMDR irrespective of the number of supported methods, and hence, trumps the SemOPC-UA approach in this regard.

8 Conclusion and Future Work

The paper in pursuit of empowering the OPC-UA with reasoning and inferencing capabilities, introduces a novel concept of integrating OWL-S and SAWSDL specifications into an hybrid, thereby extending the truly semantic communication down to the shop-floor which is the last layer of the automation pyramid. With this approach, it is possible to generate a flexible orchestration plan of manufacturing operations which can be changed in case of unforeseen events in the production with well-founded semantic reasoning of the OPC-UA methods. Additionally, the *intelligence* of semantic OPC-UA facilitates the implementation of loosely coupled production systems and thereby provides *plug and produce* and smooth reconfigurability features. Further research on automating production ontology, and determining the appropriate extent of caching of the production control data in GeSCo is on the roadmap.

References

1. Ankolekar A, Others (2001) Daml-s: semantic markup for web services. In: First international conference on semantic web working, pp 411–430
2. Battle S et al (2005) Semantic Web Services Framework (SWSF) Overview, W3C member submission
3. Berners-Lee T, Hendler J, Lassila O (2001) The semantic web. Sci Am 284:28–37
4. Borgo S, Leitão P (2004) The role of foundational ontologies in manufacturing domain applications. In: OTM 2004, pp 670–688, Springer
5. Cheng H et al (2017) Ontology-based web service integration for flexible manufacturing systems. In: INDIN 2017, pp 351–356
6. Fensel D, Bussler C (2002) The web service modeling framework wsmf. Electronic commerce research and applications, pp 113–137
7. Hermit reasoner (2018) http://www.hermit-reasoner.com. Accessed 15 Aug 2018
8. Izaguirre MJAG et al (2011) Opc-ua and dpws interoperability for factory floor monitoring using complex event processing. In: INDIN, 2011, pp 205–211. IEEE
9. Katti B, Plociennik C, Schweitzer M (2018) GeSCo: exploring the edge beneath the cloud in decentralized manufacturing. Int J Adv Syst Meas v11,1&2:183–195
10. Katti B, Plociennik C, Schweitzer M (2018) SemOPC-UA: introducing semantics to OPC-UA application specific methods. IFAC Papers Online 51(11):1230–1236
11. Katti B, Plociennik C, Schweitzer M, Ruskowski M (2018) SA-OPC-UA: introducing semantics to OPC-UA application methods. In: 14th IEEE CASE Full Proceedings p. To appear
12. Kopeckỳ J (2007) Sawsdl: semantic annotations for wsdl and xml schema. IEEE Internet Comput 11(6):60–67
13. Lobov A et al (2009) Semantic web services framework for manufacturing industries. In: ROBIO 2008, pp 2104–2108
14. Owl api (2018) http://owlcs.github.io/owlapi. Accessed 15 Aug 2018

15. Protégé (2017) https://protege.stanford.edu. Accessed 15 Aug 2018
16. Res-com project (2018) http://www.res-com-projekt.de. Accessed 15 Aug 2018
17. Rohjans S et al (2011) Opc ua goes semantics: integrated communications in smart grids. In: Emerging technologies and factory automation, pp 1–4
18. Wsdl (2001) https://www.w3.org/TR/wsdl. Accessed 15 Aug 2018