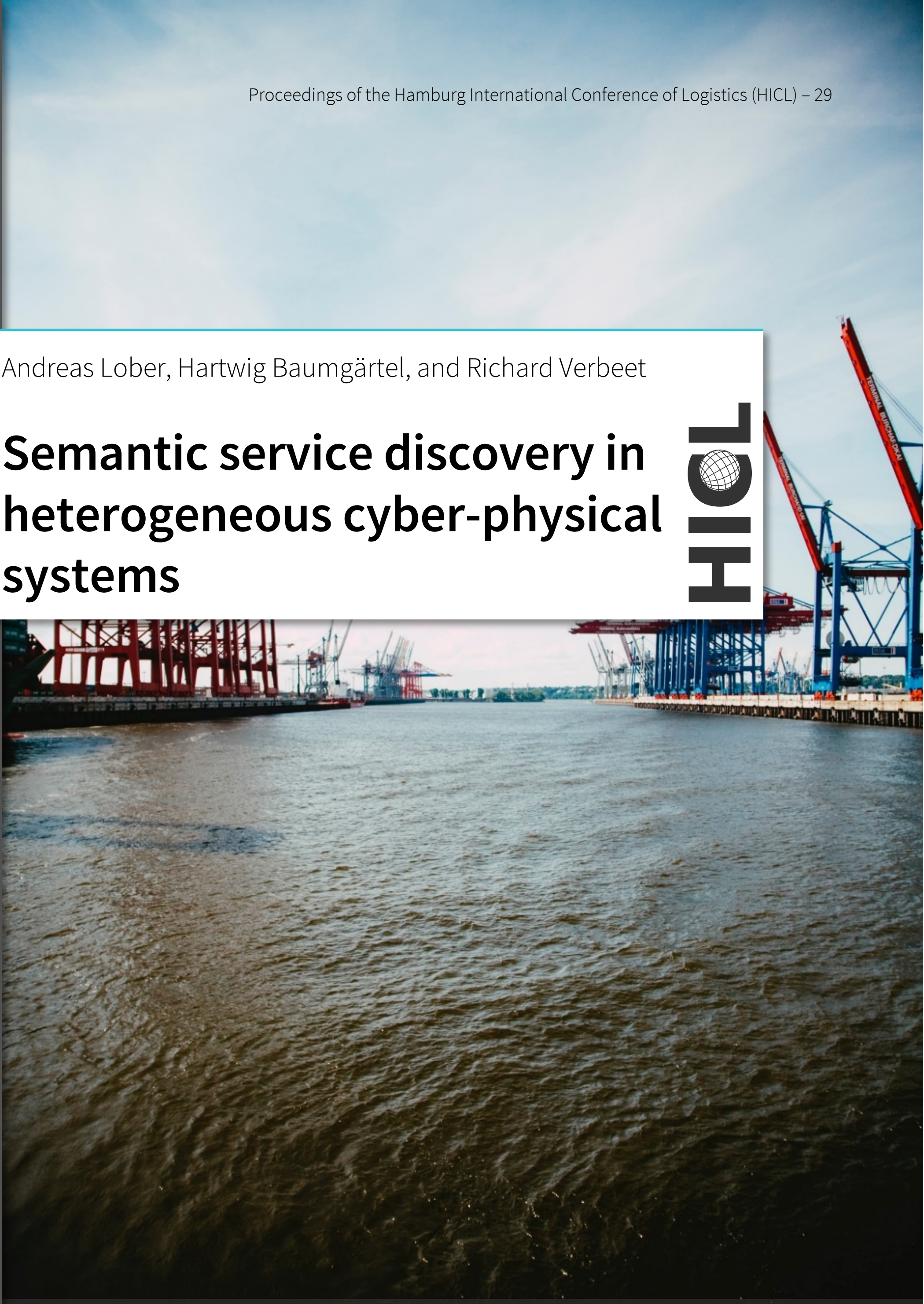


Proceedings of the Hamburg International Conference of Logistics (HICL) – 29

Andreas Lober, Hartwig Baumgärtel, and Richard Verbeet

Semantic service discovery in heterogeneous cyber-physical systems

Published in: Data science and innovation in supply chain management
Wolfgang Kersten, Thorsten Blecker and Christian M. Ringle (Eds.)
ISBN: 978-3-753123-46-2 , September 2020, epubli

Semantic service discovery in heterogeneous cyber-physical systems

Andreas Lober [1], Hartwig Baumgärtel [1], and Richard Verbeet [1]

1 – Ulm University of Applied Sciences

Purpose: A primary requirement of Industry 4.0 and realization of Cyber-Physical Systems in production and logistics is the dynamic connection of physical and digital components. Service-oriented Architectures are a well established approach to meet this requirement. However, a service discovery using syntactic descriptions of services limits efficient application of a Service-oriented Architecture concerning the complexity and variability of existing processes.
Methodology: A semantics based mechanism for service discovery can solve this limitation. It uses an ontology management system containing a domain specific ontology and modelling specific Cyber-Physical Systems as individuals. SPARQL Protocol And RDF Query Language (SPARQL) queries searching this ontology with context-related parameters. A use case demonstrates the mechanism by realizing an in-house transport request.
Findings: A syntax based service discovery requires a definition and publishing of unique service names. However, complex Cyber-Physical Systems using multiple parameters during service calls require disproportionate effort to implement and maintain these names. A semantics based service discovery considers various parameters by using a specific ontology calling services by their properties without knowing the service name.
Originality: A semantics based is decoupled from specific service implementations of components in a Cyber-Physical Systems. Therefore, an explicit specification of parameter configurations in service descriptions is not necessary. A Service-oriented Architecture can be implemented in complex systems without extensive adjustments or coordination mechanisms.

First received: 11. Mar 2020 Revised: 2. Jun 2020 Accepted: 12 Aug 2020

1 Introduction

The fourth industrial revolution is currently a widely debated topic in industry, politics, and science. Two essential aspects of Industry 4.0 (I4.0) are horizontal and vertical integration, which means networking of production and logistics systems and their components across technical system hierarchies as well as across company borders. A key problem in linking heterogeneous systems is the control of their interfaces (Baumgärtel and Verbeet, 2020). A common approach to realize this integration are Service-oriented Architectures (SoA), which enable the design of distributed systems and the implementation of dynamic access to capabilities and information of components during runtime by other components by the concept of services. Common SoA are Enterprise Service Bus, OPC UA, and Arrowhead.
In production and logistics systems, most components that must cooperate according to horizontal and vertical integration, are cyber-physical systems (CPS). They consist of physical elements, like machines, tools, robots, and conveyors, as well as of control units that are IT elements, like microcontrollers or Industrial PCs. These IT elements are parts of and connected to IT networks of their companies, e.g. to enable access to service clouds via the Internet. On the IT side, CPS can use technologies like SoA as normal computers. For example, they can offer services to others and use services of different CPS.
In the context of I4.0, CPS in production and logistics are called "I4.0 components" (BMWi, 2017). Services that are offered by I4.0 components are defined as "I4.0 services" in DIN SPEC 16593-1 (DIN, 2018), as they implement capabilities of their I4.0 component (Verbeet and Baumgärtel, 2020).

This connection of the digital and physical world leads to an increased difficulty in describing services, due to the complexity and variety of physical processes capturing by services.
Mechanisms for providing, requesting, and consuming services are a central requirement of a SoA. An essential component to realize these mechanisms is the service discovery, which means a targeted or general search for services based on a service registry and service descriptions (see Section 2). A general problem of syntax based service discovery is the complexity for installing and maintaining them to ensure the interoperability and flexibility required by I4.0. These systems have similar structures but differ in detail and use different syntax. Examples of this diversity are machines, manufacturing technologies, and handling operations, which exceed the variance of pure IT approaches that are used in the web service area. Due to globally networked processes, semiconductor manufacturing is tremendously complex manufacturing methods of all. Despite manufacturing technologies and equipment are similar for all semiconductor plants, this industry comprises a huge variety of designations even for same or similar processing technologies, steps, and equipment (Ehm, et al., 2019; Moder, et al., 2020) An alternative is the semantics based description of services with the help of ontologies. This paper discusses syntax based and semantics based approaches for service discovery and presents a concept for a semantics based using an Ontology Management System (OMS) for service registry and discovery without any syntax based searching mechanism. This concept is illustrated by an in-house transport request of the digital twin of a stored product within a production system whose components are interlinked by a SoA. The paper presents an exemplary ontology for a semantic description of services and a SPARQL query for the implementation

of a corresponding discovery. SPARQL stands for "SPARQL Protocol And RDF Query Language" and is a common query language and protocol for semantic data sources.
This paper is organized as follows: Section 2 introduces SoA and clarifies central terms and concepts. Section 3 presents the related work of semantic service discovery. Syntax and semantics based approaches for discovery are analysed in Section 4 illustrated by an in-house transport use case using a semantics based discovery in Section 5. Section 6 discusses the presented approaches and the final section 7 concludes this paper and gives an outlook on future work.

2 Service-oriented Architectures

A SoA is a style of software design modularizing previously monolithic IT systems. It is based on the concept of a service, a mechanism providing access to a capability of a software or hardware system through a predefined

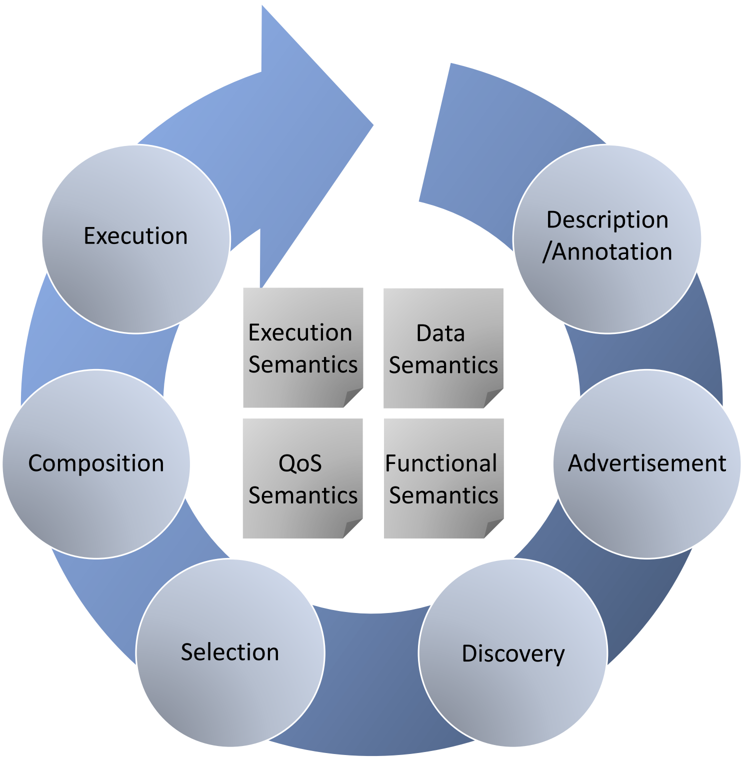


Figure 1: Service lifecycle and semantics, adopted from Cardoso and Sheth (2005)

interface by restrictions and policies defined in a service description (OA-SIS, 2006). Other systems can consume these services, creating a local or global communication network containing various hosts running software systems providing and consuming services. A service provider is a system offering services and a system using it service consumer. Service discovery is the process of finding services. During this search and selection process, i.e. before establishing a connection (service binding), a potential consumer is called service requester.

SoA base on a technical and conceptual hierarchy that consists of hosts, software systems, services, and methods. Hosts are physical resources like computers or CPS that can execute software systems. SoA software systems are installed and executed on a host and offer one or several services. Services consist of one or more methods. Methods are the basic building blocks that can be called to produce concrete results. Service consumption consists of the call and parametrization of service methods and the receipt of the produced results.

According to the OASIS reference model (OASIS, 2006), a SoA is defined by the following principal concepts: Service Description (definition of information needed to use a service), Visibility (descriptions of technical requirements, constraints, policies, and mechanisms for access or response), Interaction (definition of messages and sequence of their exchange), Real World Effect (actual result of using a service), Execution Context (technical or business conditions for interaction), and Contract and Policy (represents a constraint, condition, or agreement on use or deployment of an entity).

Cardoso and Sheth (2005) present a model for a web process lifecycle using services and semantics (Figure 1). Even if it is designed for web services it can be used to define a general lifecycle model for services. It contains the stages Description, Advertisement, Discovery, Selection, Composition, and Execution, whereby every stage might profit from using semantics.

Description: A service description contains information about a service to enable other systems to use it. Standardized languages such as Web Service Description Language (WSDL) or Web Application Description Language (WADL) can be used to syntactically describe a service. Textual service description documents contain textual descriptions of the semantics of the services, e.g. data semantics, functional semantics, and Quality of Service (QoS) semantics. If the use of a specific service is planned at the design time of a software system, the developers of this system can download these description documents, read and understand it and program their code accordingly. Machine-readable and processable descriptions of service semantics can be provided by an ontology, e.g. through a Web Ontology Language (OWL) document.

Advertisement: To use a service, potential consumer must know this service or be able to find it by service discovery. Typically, only the service name or the service name added by some key-value pairs expressing characteristic properties of services is used to register services. This information is by far not all information from the service description. Meta services can be used for targeted dissemination, dynamic dissemination can be realized via global platforms or central service registers. Universal Description, Discovery, and Integration (UDDI) and Domain Name Service for Service Discovery (DNS-SD) were approaches for a global service register (Bellwood, 2002).

Discovery: The intention of service discovery is to seek for an appropriate service in a communication network according to requester's requirements and descriptions of provided services (Dong, Hussain and Chang, 2013).

Selection: After discovering services whose descriptions match the requester’s requirements, the most suitable service must be selected. Each service can have different quality aspects and authorization policies. Therefore, service selection involves evaluating these aspects for each service leading to a filtered by authorization rating called QoS. The selection can either be based on the information given in the service registry record, or on the whole service description documents. In the latter case, the service description documents must be downloaded, processed ("understood"), and evaluated by the requester.
Composition: Service Composition is the combination of a set of services for achieving a certain purpose by developing customized services by integration and execution of existing services. According to (Liegener, 2012) different service composition types can be distinguished: Orchestration, Choreography, Coordination, and Wiring. These variants differ in their technical implementation of a service call and the place of decision making.
Execution: Execution of a service, which may include physical or digital processes, e.g. a mechanical assembly or a complex calculation procedure. A service provider can also initiate complex interactions with the service consumer by message exchange during execution of a service. Data exchange during use can be performed via application protocols like http, OPC UA binary protocol, CoAP etc.
In a distributed system of CPS, service discovery is particularly important among the phases of the life cycle due to the challenge of evaluating heterogeneous service descriptions. These descriptions can be realized syntax based or semantics based (Zorgati, Djemaa and Amor, 2019). Syntax based descriptions are implemented as a list of predefined attributes, identified

by keywords and values, whereby discovery is performed by a matching of these keywords and values to evaluate the QoS. provider and consumer each must know these keywords for a successful service binding, even a slight mismatch prevents a successful discovery. A syntax based can also be realized by matching the service name with the request. Semantics based descriptions are defined by ontologies, which contain classes for resources and services, adopt new resources and services as instances to the appropriate classes, link them by object properties and express their parameters by object or data properties. Services that meet requester's requirements in the ontology can be identified by specific search mechanisms, i.e. a semantics based is realized by a query on an ontology, e.g. by SPARQL.

3 Related Work

Many technologies and concepts appeared the first time by the mapping of software services in the web area. However, they have been adapted for other domains in recent years (Baumgärtel and Verbeet, 2020). DIN SPEC 16593-1 (DIN, 2018) defines a service as a mechanism to enable access to one or more capabilities, which must be provided by an Application Programming Interface (API). Tihomirovs and Grabis (2016) discuss the general approaches SOAP and REST for realizing a SoA and to design services. Providing services based on SOAP means to describe the service interface based on an interface language, such as WSDL (DIN, 2018). REST represents a programming paradigm for distributed systems in the context of web services, to simplify access to these services by using HTTP basic operations instead of adding a new data processing layer on the transmission and communication stack (Fielding, 2000).
In SOAP a service provider publishes this specific description to a service registry. Service consumers can use this service by requesting a instance (Tihomirovs and Grabis, 2016). DIN SPEC 16593-1 (DIN, 2018) describes SOAP-based web services as a technology that uses interface languages to allow a flexible choice of parameter names and its definition of their meaning. REST-based web services are managing self-defining resources and providing these resources which can be identified by a Uniform Resource Identifier (Tihomirovs and Grabis, 2016).
Zorgati, Djemaa and Amor define syntax based discovery as a mechanism using a protocol discovery based on a central service registry (2019). In this registry, each service has a unique identifier. Its properties are described either by its name or by an attached service description (Dong, Hussain and

Chang, 2013). Semantics based discovery relies on semantic web technologies to refer to services by ontologies.

A general example of a syntax based service discovery is based on the DNS mechanism (Klauck and Kirsche, 2013). DNS is still the most common approach for link domain and hostnames in URLs to IP addresses of physical computers. DNS-SD realizes a syntax based discovery (Klauck and Kirsche, 2013). DNS-SD allows services to be published and found by so-called DNS resource records (Cheshire and Krochmal, 2011). Another approach for the syntax based discovery is UDDI. It is motivated by the idea of a standardized directory of services (universal business registry). A business registration based on UDDI supplies three distinct sets of information: White Pages (address, contact, identifier), Yellow Pages (industrial categorizations based on standard taxonomies), and Green Pages (technical information about services). Service consumers are linked to service providers by a public brokerage system (Bellwood, 2002). Dong, Hussain, and Chang (2013) describe mechanisms for syntax based service discovery as hard to realize in an open environment with widely distributed resources. As a major obstacle, they describe the limitations of scalability and technical interoperability of heterogeneous systems in a commercial web environment.

With service description Languages (SDL) like OWL for Services (OWL-S) or Web Services Modeling Ontology (WSMO) web services can be described and made accessible as parts of larger domain ontologies (Dong, Hussain and Chang, 2013). Query languages like SPARQL, (Jacobs, 2010), can process their content (Harris and Seaborne, 2013). The semantics based approaches divide into hybrid and full semantics. Full semantics approaches use SDL to describe the services and apply semantics to discover them. A hybrid-based approach uses SDL for description or apply semantics for discovery.

Dong, Hussain and Chang classify different technologies and methods of the semantic service discoveries with six categories, e.g. discovery architectures and matching approaches. The main options for discovery architectures are the use of registers and peer-to-peer based approaches. Matching approaches are divided into logic-based, non-logic-based, hybrid, and adaptive ones (Dong, Hussain and Chang 2013).

A similar survey by Zorgati, Djemaa, and Amor (2019) discusses different approaches for syntax and semantics based discovery and compares them according to IoT requirements. It categorizes the syntax based service discoveries that are using a protocol into CoAP-based, MQTT-based, and DNS-based discoveries. Semantics based approaches use description languages to describe services and semantics to find those services. In summary, many approaches use a register or a peer-to-peer mechanism for discovery and using description languages such as WSMO, SAWSDL, OWL-S, or WSDL-S for description.

Lobov et al. (2019) show a concept in which a combination of SoA and agents for service composition and discovery is presented. This approach uses an ontology to describe the capabilities of resources in a production environment and is an example of a hybrid mechanism of service description and discovery. A SPARQL query identifies the next resource for an upcoming production step. In contrast to other approaches, the mechanism searches for a requested production step instead of services. The description of the services is based on WSDL and can be assigned to the syntactic description languages according to the classification of Zorgati, Djemaa, and Amor (Zorgati, Djemaa and Amor, 2019).

Fujii and Suda describe a framework consisting of three elements: CoSMoS, CoRE, and SeGSeC (2009). CoSMoS defines different resources together with their dependencies on operations. The approach uses an ontology for knowledge representation and applies different discovery technologies (Fujii and Suda, 2009). A semantic wrapper is used to translate a syntax based request into a semantics based.

Fumagalli et al. propose a similar approach using the digital platform eScop to combine embedded systems with an ontology-driven service-based architecture. (2014). The ontology describes a production system to enable service matching. This description is called eScop MSO and represents a type of domain ontology (Fumagalli, et al., 2014). Juan and Yanzhong define an OMS as a software system managing ontologies through their lifecycle containing its built, test, usage, and maintenance (2010). Within the European research project Productive4.0 another approach of an OMS is used. One aim of this project is to implement a digital platform for offering and searching diverse services (Ehm, et al., 2019). The architecture of this platform is described by Sipsas et al. (2020). Content of this platform is the Digital Reference (DR) Ontology, which combines knowledge of a supply network and is modelled by an ontology consisting of several small ontologies from different domains, e.g. production, process, and supply chain. This ontology can be regarded as a domain ontology and services described in OWL-S can be integrated into it.

4 Semantic Service Discovery

For the discovery of these services and their methods, three different approaches exist: syntax based, hybrid, and semantics based. It will be shown how they generally behave when their description is expanded by parameters, whereby a parameter in this context represents specific information of a service to distinguish it from other services, e.g. size or maximum weight. Figure 2 shows variants of different syntax based and semantics based discoveries. They differ in their mechanism for registry and discovery. A third variant combining both concepts using a semantic wrapper is also presented.

In a syntax based approach, parameters are stored in the service description documents and partially are encoded in the registration information of the services, i.e. the name and optional key-value pairs, to describe it as precisely as possible for discovery. Therefore, syntax based search depends on meaningful names of services and parameters that must be known by a requester in advance. In the syntax based variant, a provider registers a service by its name in the registry. When a service is requested, the requester contacts the registry. This request needs the specific name of the service the requester must know before. As a result, the registry responds to a list of all services and their endpoints, which match with the requested service.

The hybrid variant uses an additional semantic wrapper during service discovery (Fujii and Suda, 2009). The registration of a service by a provider is still syntax based using a specific name in the registry. However, a semantic wrapper is placed between the requester and the registry supporting a service request. The requester can query the ontology within the wrapper for services by using SPARQL. This ontology describes the properties (classes) and characteristics (individuals) of services. Each service is assigned with a name, which is stored in a data property of the individual. Therefore, the query process is based on two steps. First, the wrapper determines one or more syntactic names of services by a SPARQL query and uses those names to initiate a classic query at the registry. Now the wrapper submits an address, a name, or reference of the original requester to the registry in terms of "reply to". Thereafter a list with service names is sent to the requester by

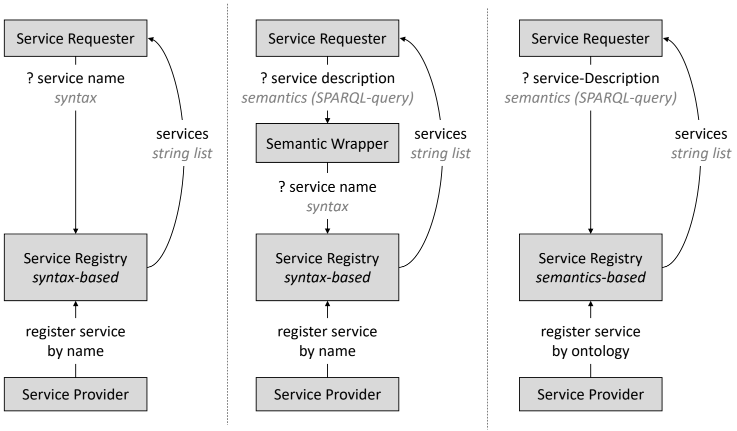


Figure 2: Approaches for Service Discovery: Syntax based (left), Hybrid (middle), Semantics based (right), own illustration

the registry. Registered services at a registry must be mapped into the ontology of the wrapper. This is done by a mapping process connecting the syntactic name of a service with existing classes and individuals of the ontology. Thereby a requester does not need to know the syntactic service name when making a request. It can also reach a service with different syntactic names if they are stored in the ontology.

In the full semantics based variant a wrapper is not necessary since service discovery is carried out exclusively semantically by an OMS, e.g. Apache Jena Fuseki. A provider registers a service at the registry using an ontology based service description. During this registration, a class of services is searched in the T-Box of the ontology describing this service to register it as an individual for this class, whereby any syntactic name can be chosen for the service. The registered service can be linked via relations (object properties) and matching properties (parameters), which both are included in the online hierarchy. A request by a SPARQL query is transferred directly to the registry respectively its ontology. This request finds corresponding services as individuals and results in a list with the names and endpoints of these services, which are stored as properties for the individual. A semantics based enables querying all parameters defining a service more precisely. Due to the specific search for a service using a SPARQL query only services matching requester's requirements will be shown. The requirements are a consistent structure of the ontology and a comprehensible classification of services as individuals in this ontology. Besides that, the semantics based approaches start with the description of the services that are transferred to a registry. These descriptions must be designed so that they can be inserted into an ontology. To achieve this, two main approaches exist:

The first approach requires a small ontology that is created by the manufacturer of a component for its service. Such ontologies can, for example, be expressed in OWL-S. This ontology is sent to the registry for registration.

By ontology merging the new partial ontology introduces all registered services into the already existing domain ontology in the registry (Martin, et al., 2004).

The second approach is that the service manufacturer sends a conventional list of parameters as key-value pairs, in which each element is supplemented by an annotation that refers to a known ontology. In the case of sensors, SOSA/SSN could be such an ontology. In this case, the target system operator would only have to announce in advance that he is working with the ontology. When registering, the key-value pairs and the service endpoints can then be included in the ontology of the registry according to their annotations. Additionally, the second case offers the possibility to be associated with an extension of the WADL file. For this purpose, WADL can use a reference to a grammar in the header, which is the basis of the description. Similarly, an element for one or more ontologies can be included in this header. The annotations are added to each element using additional XML tags or XML attributes. In such a case this would mean for the registry that there is consistency between the WADL file, and the parameters reported to the registry, provided that the parameters for the registry are filtered out of this file.

5 Use Case In-house Transport

The following chapter describes a semantics based discovery for an in-house transport task.

5.1 Scenario

A europallet stores a product which is requested to be transported to a supply location beneath an assembly line. The logistics of the production system contains various alternatives for transportation for carrying out this transport: forklifts, AGVs, and flying drones. They provide their capabilities as services that are registered at a central semantic service registry using an ontology. The product is represented by a digital twin, which takes care of its transport autonomously and can request a transport service. In addition to different technical properties and capabilities the heterogeneity of transport systems is increased by using systems purchased from different manufacturers. The delivery should be made by direct transport, i.e. no combination of different transport systems is considered. Besides restrictions for weight, the transport is also time-critical due to synchronization with assembly processes.

5.2 Syntax based Service Discovery

For realizing the shown use case, a system implements a central syntax based registry and discovery. In this system, components must register their service. Also, they store a description to describe this service and to define parameters as well as their value ranges. These parameters are nec-

essary for the service request. This description must follow a standard format and structure, so that a centralized system can evaluate and take these services into account in discovery. If there is no standard like UDDI or DNS-SD, evaluation mechanisms must be implemented in this system for matching different descriptions. Alternatively, the system can read and interpret different formats and structures.
The Transport system registers the service "TransportService" and sends additional parameters to the registry to be stored in a database to be used for matching with service requests. Information for service registration can be sent in JSON format. Relevant parameters for the use case are Asset, MaxWeight, and MaxSpeed. The service could contain even more, e.g. Provider, NetworkID, and ServiceCall. Figure 3 shows an exemplary JSON description of a forklift service.

```
{
 "Service": "TransportService",
 "Provider": "Forklift 4712",
 "NetworkID": 151.12.5.125:23001,
 "Parameters ":
   {
    "Assets":["SLC", "Europallet", "Container"],
    "MaxWeight": 1500,
    "MaxSpeedValue": 1.5,
    "MaxSpeedUoM": "meter per second",
    "ServiceCall": ["Asset", "Weight", "Start", "Destination"]
   }
}
```

Figure 3: Syntax based description of a forklift service in JSON, own illustration

If a request is made, all descriptions of registered services filtered by the specified parameters. The system reports only services which fulfil the request. In the use case, a load carrier requests a "TransportService" with appropriate parameters for weight and speed. But for filtering for a specific speed, both the requested speed value and the correct unit of measure have to be stated in the query. As a result, the registry returns a list of providers. It can retrieve their services with the information of the service call.

5.3 Transport Ontology

The service ontology for an in-house transport is shown in figure 4, which is an excerpt of an ontology containing all resources and services of the warehouse and production system to illustrate the mentioned use case. It describes the five-class hierarchy moving_service, move asset, resource, data format, and protocol.
In the class moved_asset movable physical and digital handling units as well as humans are listed to describe transported objects. The individuals of respective subclasses can be used to describe a service more precisely by adding additional data properties.

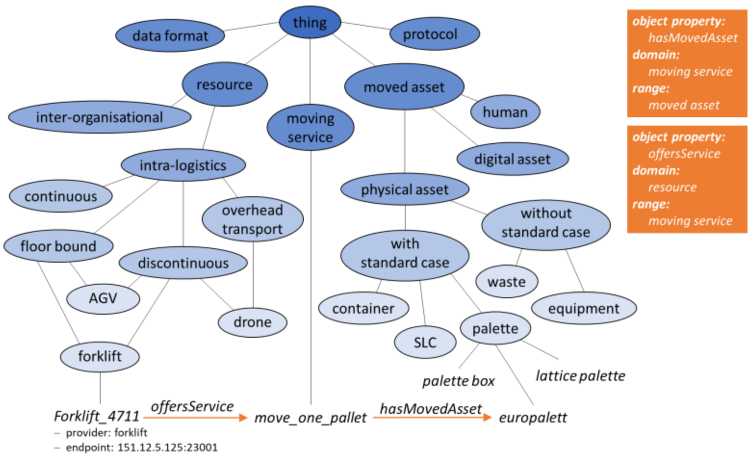


Figure 4: Service ontology for in-house transport, own illustration

In this use case europallet, palette box, and lattice palette are assigned to the palette subgroup as individuals. A specific maximum weight or standardized dimensions can be attached to these individuals via data properties.

The class moving service specifies existing services to transport a moved asset. The Resources are divided into "inter-organizational" and "intra-logistics". A corresponding service provided by forklifts is shown in figure 4 as an example of an in-house transport of pallets. Assets that are intended to use this service must possess the property "accessible for fork". Special application cases can be integrated analogously.

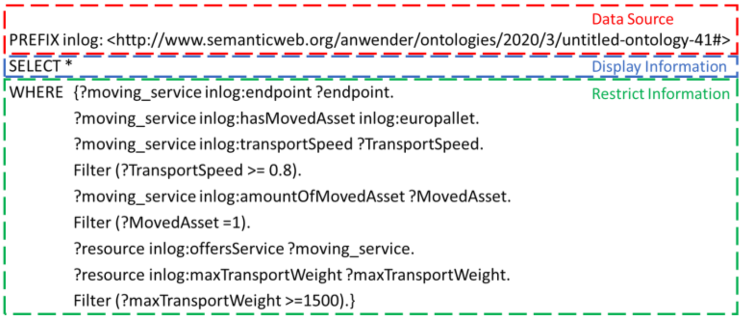
According to the system hierarchy of a SoA, a resource is a physical device representing a host whose integrated control software represents the software system providing a service. The resource forklift is assigned to "move_one_pallet", an individual of the class forklift transport service. This

assignment is described by the object property "offersService" and at the same time implemented by a data property of type String containing the endpoint of the service. The endpoint represents information about the host and therefore can implicitly describe the provider. If resources are included, they can be connected to the services they provide via an object property. In figure 4 this is indicated by the class resource.
The properties used to classify services can be viewed as parameters for a request. Since a transport service can be described by various parameters and it becomes difficult to guarantee the completeness of the taxonomy, other parameters instead are created as separate sub-taxonomies in the ontology. The parameters contained in these sub-taxonomies are linked with appropriate services. Therefore, services are linked to the transported objects in the in-house transport use case. The classes moved asset (Range) and moving service (Domain) are linked with each other by the object property "hasMovedAsset". As an instance of the object property has moved asset, the individual move_one_pallet is assigned to the individual europallet of the subclass palette.
The classes protocol (e.g. HTTP) and data format (e.g. JSON, WADL) can be used for further specification of the service realization.

5.4 SPARQL-Query

A service discovery for the discussed in-house transport use case, using the presented ontology, can be realized by a SPARQL query. Figure 5 shows an exemplary query, which the digital twin of the product must process to find a suitable transport service. The SPARQL query contains an upper, mid, and lower area. The upper area names prefixes that contain different data

sources. The middle area specifies the displayed information (SELECT) and restrictions for this information (WHERE) defines the lower area. In the SELECT clause all columns are displayed by using "*", that match all restrictions. It is filtered by conditions defined in the WHERE clause. These conditions describe the transport request. The first line returns all services in the system with their associated endpoints. Any transport services of individual resources represented in the ontology are added by the second line. The third line filters the results on all resources that can transport a europallet. The last six lines add filters to remove the services which are not matching the restrictions for weight and transport speed. The lines querying all existing values are first presented in pairs querying all existing values and thereafter filter them by a relational operator. In the in-house transport scenario these restrictions are “speed ≥ 0.8m/s”, “weight ≥ 1500kg”, and "amount of moved asset = 1".
A list of service endpoints and their resources are returned because of this



```
PREFIX inlog: <http://www.semanticweb.org/anwender/ontologies/2020/3/untitled-ontology-41#>
SELECT *
WHERE  {?moving_service inlog:endpoint ?endpoint.
        ?moving_service inlog:hasMovedAsset inlog:europallet.
        ?moving_service inlog:transportSpeed ?TransportSpeed.
        Filter (?TransportSpeed >= 0.8).
        ?moving_service inlog:amountOfMovedAsset ?MovedAsset.
        Filter (?MovedAsset =1).
        ?resource inlog:offersService ?moving_service.
        ?resource inlog:maxTransportWeight ?maxTransportWeight.
        Filter (?maxTransportWeight >=1500).}
```

Figure 5: SPARQL query as Service Discovery, own illustration

query. Those endpoints are the network address of resources (host) on which the service software is running, added by the port of the service. The

SPARQL query of figure 5 lists move_one_pallet as possible moving service which can be offered by forklift_4712 and AGV_4731. The moving services of Forklift_4711 and drone_4721 does not correspond to the parameters of the SPARQL query.
The shown system was simplified for the presented use case, as it does not elaborate the methods of the services.

6 Discussion

Three types of service description are discussed in this work: (i) description of a service for registration, e.g. a record in DNS-SD with endpoint and key-value pairs expressing some important characteristics of the service,(ii) automatically processable service description, e.g. a WADL document or an OPC UA information model, and (iii) additional documentation of a service in form of a text document, e.g. service descriptions provided by the Service Description, Interface Design Description, and System Design Description in the Arrowhead documentation model (Delsing, 2017). It should be noted that the first kind is not referred to as "service description" by the SoA community, but as "service registration record" or "service advertisement information". However, the nature of this information is to describe a service so that it can used for discovery.

Many RESTful services and APIs rely on syntax based approaches for information for registry and machine processable descriptions. Both human-directed textual descriptions and machine-directed formalized descriptions like WADL documents carry any machine-processable semantics. WADL documents often describe only data formats instead of information semantics, and textual documents use examples for description which need to be understood by humans. Hence, automated service or API discovery and composition are not possible (Verborgh, et al., 2014).

Furthermore, a description used for registration is different from automatically processable descriptions by a WADL document Therefore, the existence of detailed descriptions and PDF documents does not say anything about how a service is registered and how it can be searched for.

A requirement for search of services is knowledge about names and various notations of a service and its characteristics describing keys and values. Without this knowledge a search query can only be based on educated guesses by which, many potential services may be missed, or, in case of very unspecific queries, too much inadequate services are returned. If every key with every value is queried, then the parameters have no restrictive effect on a search and the discovery of the right service might become time consuming. A semantics based mechanism does not need to know conventions of services provided or consumed by other systems regarding syntax of names and parameters with value ranges. Standardized specifications can be replaced by synonyms in the ontology. By using an OMS, a semantics based can handle a high number of combinatorial possibilities. In general, a semantics based discovery performs a parameter specific and thus a demand-oriented request. The query result better meets the query intention, even if exact names of services or parameters are not known.

The use case implementation from Section 5 delivers a proof of concept for the pure semantics based discovery approach. The main components for this approach are i) a semantic service description by an ontology, expressed in a standard ontology serialization format like OWL, ii) a domain ontology to which the description ontology can be linked or merged, iii) an OMS that contains both ontologies and acts as registry component of the SoA, and iv) the use of SPARQL as logic-based matching approach for query answering, that has to be provided as service by the OMS.

Since OMS and SPARQL services are often seen as standard in semantics based environments, it is important to mention them as necessary components for a semantics based discovery approach. Many authors focus in

their papers on this subject on semantic descriptions or annotations of services. However, these descriptions can only be effective for service searches if they are managed by an OMS and the OMS is used as a service registration component. Hence, approaches for the implementation of Cyber-physical production and logistics systems (CPPS and CPLS) should use a general architecture as exemplified by the proof of concept.
Furthermore, for practical applications a well established domain ontology should be used. The DR Ontology, developed in the Productive4.0 project, is a promising approach for this. It can and will be extended by subsequent projects, e.g. with general domain descriptions for several industries (Ehm, et al., 2019).
The need for semantics based discovery approaches in CPPS and CPLS results from their inherent distributed, complex, and heterogeneous nature and the requirements on their dynamics. These systems are requested to be able to dynamically react to new situations, e.g. by dynamically reconfigure their processes and operations. This requires the ability to dynamically discover appropriate services and compose them. That is, the SoA realizing the IT part of these systems must be able to realize the "late binding" principle for services in heterogeneous environments with changing components and unknown service names. This contradicts with established software development approaches in the Web Service or RESTful http based service world, where service bindings are most often fixed at design time, or late binding bases on a priori known service names. There, textual, non-machine-readable service descriptions can be used by the software developers to know the names and understand the semantics of the services they want to use.

Even OPC UA does not fully support efficient late binding. Service descriptions as part of information models are automatically accessible and machine-readable due to the OPC UA standard services. However, to analyse the semantics of application services offered by an OPC UA Server requires to browse the information model, find and process the according data and conclude on the semantics. In case of many OPC UA Servers are available in a production or logistics environment, this must be performed for many different servers which makes the discovery, decision and composition inefficient. The reason behind this is, that OPC UA is not designed for highly dynamic systems, but to support efficient commissioning of components in production and logistics systems.

For late binding in the described context, existence of machine-readable and -processable descriptions of service semantics is mandatory. They must be used for service registry and discovery. With this, the information gap between information for registry and service descriptions can be minimized, and late binding can be executed efficiently.

7 Conclusion and future work

The presented work discusses three different type of service discovery: syntax based, hybrid, and semantics based. An in-house transport use case is provided showing the processing of a transport request for a europallet using a semantics based discovery. A limitation of syntax based service descriptions for registration and discovery regarding names, parameters and value ranges is discussed, revealing a requirement of specific knowledge about services in advance.
Also, a proof of concept for a pure semantics based discovery approach is presented, which can be divided in its four-part architecture, consisting of a semantic description, e.g. in OWL, a domain ontology, an OMS and the use of a logic-based matching approach like using a SPARQL query.
In further research, other use cases will be investigated for the applicability of a semantic discovery and composition in CPS. For this purpose, a flexible manufacturing system and an extended in-house transport system will be examined due to their high flexibility and complexity. The objective is to define solutions for open points like, e.g. the creation of a uniform general ontologyor the scalability performance of SPARQL queries, when the number of parameters and size of the ontology increases.

References

Baumgärtel, H. and Verbeet, R., 2020. Service and Agent based System Architectures for Industrie 4.0 Systems. In: B. Vogel-Heuser, T. Bauernhansl, and M. ten Hompel, eds. 2020. Handbuch Industrie 4.0. Industrie-4.0-Anwendungsszenarien für die Automatisierung. 3rd ed. Berlin, Heidelberg: Springer-Verlag.

Bellwood, T., 2002. UDDI Version 2.04: API Specification. [online] Available at: <http://www.uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm> [Accessed 12 May 2020].

BMWi, 2017. Fortschreibung der Anwendungsszenarien der Plattform Industrie 4.0: Fortentwicklung des Referenzmodells für die Industrie 4.0-Komponente SG Modelle und Standards.

Cardoso, J. and Sheth, A., 2005. Introduction to Semantic Web Services and Web Process Composition. In: J. Cardoso, and A. Sheth, eds. 2005. Semantic web services and web process composition. First international workshop, SWSWPC 2004, San Diego, CA, USA, July 6, 2004 ; revised selected papers. Berlin: Springer, pp. 1–13.

Cheshire, S. and Krochmal, M., 2011. DNS-Based Service Discovery. [online] Available at: <https://tools.ietf.org/html/draft-cheshire-dnsext-dns-sd-11> [Accessed 12 May 2020].

Delsing, J., 2017. IoT automation: Arrowhead framework. Boca Raton, FL: CRC Press, Taylor & Francis Group.

DIN, 2018. DIN SPEC 16593-1. DIN SPEC 16593-1:2018-04, RM-SA_- Referenzmodell für Industrie 4.0 Servicearchitekturen_- Teil_1: Grundkonzepte einer interaktionsbasierten Architektur; Text_Englisch. Berlin: Beuth Verlag GmbH.

Dong, H., Hussain, F. K. and Chang, E., 2013. Semantic Web Service matchmakers: state of the art and challenges. Concurrency and Computation: Practice and Experience, [e-journal] 25(7), pp. 961–988. http://dx.doi.org/10.1002/cpe.2886.

Ehm, H., Ramzy, N., Moder, P., Summerer, C., Fetz, S. and Neau, C., 2019. Digital Reference – A Semantic Web for Semiconductor Manufacturing and Supply Chains Containing Semiconductors. In: 2019 Winter Simulation Conference (WSC). 2019 Winter Simulation Conference (WSC). National Harbor, MD, USA, 08.12.2019 - 11.12.2019: IEEE, pp. 2409–2418.

Fielding, R. T., 2000. Representational State Transfer (REST). [online] Available at: <https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm> [Accessed 12 May 2020].

Fujii, K. and Suda, T., 2009. Semantics-based context-aware dynamic service composition. ACM Transactions on Autonomous and Adaptive Systems, [e-journal] 4(2), pp. 1–31. http://dx.doi.org/10.1145/1516533.1516536.

Fumagalli, L., Pala, S., Garetti, M. and Negri, E., 2014. Ontology-Based Modeling of Manufacturing and Logistics Systems for a New MES Architecture. In: B. Grabot, B. Vallespir, S. Gomes, A. Bouras, and D. Kiritsis, eds. 2014. Advances in Production Management Systems. Innovative and Knowledge-Based Production Management in a Global-Local World. IFIP WG 5.7 International Conference, APMS 2014, Ajaccio, France, September 20-24, 2014, Proceedings, Part I. Berlin, Heidelberg, s.l.: Springer Berlin Heidelberg, pp. 192–200.

Harris, S. and Seaborne, A., 2013. W3C SPARQL 1.1 Query Language. [online] Available at: <https://www.w3.org/TR/sparql11-query/> [Accessed 12 May 2020].

Jacobs, I., 2010. Description of W3C Technology Stack Illustration. [online] Available at: <https://www.w3.org/Consortium/techstack-desc.html> [Accessed 13 May 2020].

Juan, Y. and Yanzhong, D., 2010. A Framework of Ontology Management System. In: E-Business and E-Government (ICEE), 2010 International Conference on. 2010 International Conference on E-Business and E-Government (ICEE). Guangzhou, China, 5/7/2010 - 5/9/2010: IEEE, pp. 1719–1722.

Klauck, R. and Kirsche, M., 2013. Enhanced DNS message compression - Optimizing mDNS/DNS-SD for the use in 6LoWPANs. In: 2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops). 2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops 2013). San Diego, CA, 18.03.2013 - 22.03.2013: IEEE, pp. 596–601.

Liegener, B., 2012. Service Composition: Definitions. [online] Available at: <https://www.s-cube-network.eu/km/terms/s/service-composition/index.html> [Accessed 18 February 2020].

Lobov, A., Lopez, F. U., Herrera, V. V., Puttonen, J. and Lastra, J., 2019. Semantic Web Services framework for manufacturing industries. In: 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC). 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC). Bari, Italy: IEEE, pp. 2104–2108.

Martin, D., Burstein, M., Hobbs, J., Nokia, O. L., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parisa, B., Payne, T., Srinivasan, N. and Sycaram, K., 2004. OWL-S: Semantic Markup for Web Services. [online] Available at: <http://www.w3.org/Submission/OWL-S>.

Moder, P., Ehm, H., Baumgärtel, H. and Ramzy, N., 2020. Semantic Web: Befähiger der Industrie 4.0. In: B. Vogel-Heuser, T. Bauernhansl, and M. ten Hompel, eds. 2020. Handbuch Industrie 4.0. Industrie-4.0-Anwendungsszenarien für die Automatisierung. 3rd ed. Berlin, Heidelberg: Springer-Verlag, pp. 1–25.

OASIS, 2006. Reference Model for Service Oriented Architecture 1.0: Committee Specification 1. [online] Available at: <http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf> [Accessed 18 February 2020].

Sipsas, K., Zalonis, A., Broechler, R., Baumgärtel, H. and Ehm, H., 2020. A platform appliaction for exploiting the Digital Reference Ontology. In: IEEE International Conference on Industrial Cyber-Physical Systems 2020. Tampere, Finland, 10.06.2020 - 12.06.2020: IEEE.

Tihomirovs, J. and Grabis, J., 2016. Comparison of SOAP and REST Based Web Services Using Software Evaluation Metrics. Information Technology and Management Science, [e-journal] 19(1). http://dx.doi.org/10.1515/itms-2016-0017.

Verbeet, R. and Baumgärtel, H., 2020. Implementierung von autonomen Industrie 4.0-Systemen mit BDI-Agenten. In: B. Vogel-Heuser, T. Bauernhansl, and M. ten Hompel, eds. 2020. Handbuch Industrie 4.0. Industrie-4.0-Anwendungsszenarien für die Automatisierung. 3rd ed. Berlin, Heidelberg: Springer-Verlag.

Verborgh, R., Harth, A., Maleshkova, M., Stadtmüller, S., Steiner, T., Taheriyan, M. and and Van de Walle, Rik, 2014. Survey of Semantic Description of REST APIs. In: C. Pautasso, E. Wilde, and R. Alarcon, eds. 2014. REST: advanced research topics and practical applications. New York, NY, Heidelberg, Dordrecht: Springer, p. 70–70.

Zorgati, H., Djemaa, R. B. and Amor, I. A. B., 2019. Service discovery techniques in Internet of Things: a survey. In: 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC). 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC). Bari, Italy: IEEE, pp. 1720–1725.