Future Internet **2012**, *4*, 737-761; doi:10.3390/fi4030737

OPEN ACCESS

future internet

ISSN 1999-5903

www.mdpi.com/journal/futureinternet

Article

Context-Based Orchestration for Control of Resource-Efficient Manufacturing Processes

Matthias Loskyll [1,*], Ines Heck [2], Jochen Schlick [1] and Michael Schwarz [1]

[1] German Research Center for Artificial Intelligence (DFKI), Innovative Factory Systems, Trippstadter Strasse 122, Kaiserslautern 67663, Germany; E-Mails: jochen.schlick@dfki.de (J.S.); michael.schwarz@dfki.de (M.S.)

[2] Technology-Initiative SmartFactoryKL e.V., Trippstadter Strasse 122, Kaiserslautern 67663, Germany; E-Mail: heck@smartfactory.de

* Author to whom correspondence should be addressed; E-Mail: matthias.loskyll@dfki.de; Tel.: +49-631-20575-5285; Fax: +49-631-20575-3402.

Received: 18 June 2012; in revised form: 31 July 2012 / Accepted: 9 August 2012 / Published: 14 August 2012

Abstract: The increasing competition between manufacturers, the shortening of innovation cycles and the growing importance of resource-efficient manufacturing demand a higher versatility of factory automation. Service-oriented approaches depict a promising possibility to realize new control architectures by encapsulating the functionality of mechatronic devices into services. An efficient discovery, context-based selection and dynamic orchestration of these services are the key features for the creation of highly adaptable manufacturing processes. We describe a semantic service discovery and ad-hoc orchestration system, which is able to react to new process variants and changed contextual information (e.g., failure of field devices, requirements on the consumption of resources). Because a standardized vocabulary, especially for the description of mechatronic functionalities, is still missing in the manufacturing domain, the semantic description of services, processes and manufacturing plants as well as the semantic interpretation of contextual information play an important part.

Keywords: semantic technologies; ontology development; semantic services; semantic interoperable architectures; manufacturing processes; smart factory

---

1. Introduction

Industrial manufacturing is one of the world's biggest economic factors. While Internet technologies keep changing the everyday life and open up new business fields in the consumer world, the domain of industrial manufacturing seems to be much more difficult to enter. However, in recent years, megatrends like globalization and dynamism of product life cycles require companies to be more and more agile and flexible. Production equipment and processes have to be adapted more and more quickly to new products and product variants. Modularization and reuse of manufacturing equipment is one central approach to address those issues. In order to fully exploit the benefits open interfaces as well as seamless integration of business and technical processes becomes ever more important. Internet technologies like HTTP, SOAP and Semantic Web are providing means to realize open interfaces, to provide well established communication channels and to seamlessly integrate business processes with technical processes.

In order to close the digital gap between computer systems and the real world products, machines and other equipment are being equipped with auto ID technologies and computing and communication facilities. In this way, smart factories become a promising application of the Internet of Things [1]. From the point of view of production automation, the classical paradigms of control technology grounding on a signal based view and designing closed systems are not suited anymore for the control of equipment in future smart factories. New open approaches have to be developed.

One approach is the concept of Service-oriented Architectures (SOA). In the field of business IT, SOA has been an established element for several years and is used primarily to orchestrate and execute business processes. Basically, SOA is an abstract concept for software architectures that represents different methods or applications as loosely coupled and openly accessible services and in this way allows platform-independent use and reuse. In the application field of business IT, SOA helps to increase flexibility and interoperability.

In order to integrate business and technical manufacturing processes seamlessly, the SOA idea has to be adapted to industrial automation technology. The basic approach is to encapsulate all control functions within a factory control system as self-contained services. Business processes are cascaded down to the level of technical sub processes. They are in turn provided by modules or even single technical devices. Those devices represent mechatronic systems consisting of sensors, actuators and control intelligence [2,3]. In order to use established technologies for SOA (e.g., SOAP or REST) the mechatronic modules have to be extended with communication capabilities using TCP/IP and HTTP.

In traditional SOA approaches the description of service interfaces using current standards like WSDL is of pure syntactic nature. This means that the meaning of a service's capabilities, for instance, must be interpreted by the service user. Furthermore, service repositories based on UDDI do only provide a keyword-based search rather than a capability-based discovery of services. These difficulties also affect the orchestration of services. As service orchestration using languages like BPEL is based on the syntactic description of services, process designers must bind appropriate services at design time, resulting in a static orchestration. In order to meet the requirements of future smart factories, it must be possible to orchestrate processes dynamically and based on the current context.

Semantic web service technologies can be used to describe the meaning of service capabilities in a machine-understandable manner by adding a semantic layer to the syntactic service descriptions. In

this way, an efficient discovery and dynamic orchestration of services can be realized [4]. This concept offers the possibility to describe the manufacturing process in an abstract way on a semantic level and to assign concrete services of devices and manufacturing modules right at the moment the product enters the manufacturing line. The abstract description of the process could even be attached directly to the product (e.g., stored on a digital product memory [5]) turning the product into an active component of the manufacturing line.

When dynamically orchestrating web services contextual information can be used to influence the choice or the parameterization of the respective service leading to the concept of context-based orchestration. Contextual information can be information about current environmental influences or the current state of products, machines, etc. One of the most common definitions of context describes it as "any information that can be used to characterize the situation of an entity" [6]. In the domain of production automation, the production process and the associated task that needs to be fulfilled are the most important aspects. All production entities that are relevant for the execution of this task, either by actively participating in the process or by supplying input (e.g., material, information) can deliver contextual information about themselves. Influencing the discovery of device and module services by taking advantage of contextual information at deployment or run time significantly increases the agility of smart factories and boosts productivity when manufacturing highly individualized products.

This article describes how semantic technologies can be applied to the dynamic context-based orchestration of technical manufacturing processes based on the concept of service-oriented architectures. Whereas existing syntactic approaches (e.g., using web services) are sufficient to realize a technical interoperability, a semantic interoperability can only be achieved by using sophisticated semantic models. Therefore, we discuss how a structure of modular ontologies can be developed and which upper ontology is suited to act as a common interoperability model for our approach.

After a brief introduction to the basic paradigms and concepts of service-oriented architectures and semantic web services, the article shows how these approaches can be transferred to the field of production automation (Section 2). Based on three central use cases, the authors derive which kind of semantic models are needed for the semantic description of web services provided by field devices, their discovery, selection and orchestration. The development of respective ontologies is illustrated in Section 3. In addition, a system architecture is presented and the central aspects of implementation are discussed. In an experimental setup the functionality of the theoretical approaches is proven by means of the scenario of resource-efficient manufacturing of a smart product (Section 4). The article concludes with a short summary and an outlook for future work.

2. Background and Related Work

2.1. Service-Oriented Architectures

Service-oriented Architecture (SOA) describes a collection of design principles for the development of distributed systems. SOA relies on the paradigm of service-orientation, *i.e.*, the different software components provide their functionalities as loosely-coupled services over a network. This means that services provide functionalities over standardized interfaces independent from the underlying implementation. Web service technologies have emerged as the most prevailing implementation of

SOA. XML-based languages like WSDL (Web Service Description Language) are used to describe a service's interface, functionalities and characteristics. The most important activities in a SOA-based system are discovery and orchestration. The UDDI (Universal Description Discovery and Integration) specification depicts a way to build service repositories, in which services can be published by service providers and searched and discovered by service requesters. The combination of existing web services to more complex services or processes is usually performed using BPEL (Business Process Execution Language).

There are several issues concerning the implementation and usage of systems based on the SOA paradigm using syntactic approaches [7]. Semantic technologies can be used to describe the meaning of service capabilities in a machine-understandable manner by adding a semantic layer to the syntactic service descriptions. In this way, an efficient discovery and dynamic orchestration of services can be realized. In the following sections, semantic technologies and the different approaches for the definition of semantic web services are illustrated.

2.2. Semantic Technologies

Semantic technologies allow the formal description of data and support the semantic processing of data by machines, *i.e.*, the interpretation of electronically stored pieces of information with regard to their content and meaning. The formal, explicit representation of knowledge forms the cornerstone of the Semantic Web and includes both the modeling of knowledge and the definition of formal logics, which provide rules to draw inferences over the modeled knowledge base.

While several semantic modeling approaches for the representation of knowledge with different expressional power exist (e.g., taxonomies, thesauri, topic maps), ontologies depict the most popular and the most powerful approach of explicit knowledge representation. Ontologies, often defined as "an explicit specification of a conceptualization" [8], enable the modeling of information as an independent knowledge base consisting of three basic structures, namely classes, relations and instances.

Ontologies facilitate the structured exchange of information among heterogeneous systems, resulting in a semantic interoperability. To this end, special description languages are needed. The Web Ontology Language (OWL) is the ontology description language of the Semantic Web initiative standardized by the W3C [9]. OWL provides formal semantics and an RDF/XML-based syntax. The Resource Description Framework (RDF) is a formal description language for meta data, which describes information by means of so called RDF triples (subject–predicate–object) [10]. RDFS (RDF Schema) extends RDF by properties, domain and range constructs and hierarchical structures. It is already possible to model simple taxonomies using RDFS. However, OWL is the most expressive of these description languages. It provides additional features to describe complex logical expressions, cardinalities and axioms (e.g., disjunction) *etc*.

2.3. Semantic Web Services

The syntactic description of web services and their orchestration results in a semantic gap between the syntactic description of the web service interface and the meaning of the underlying functionality. On the basis of a semantic annotation using semantic technologies, the meaning of a web service definition can be described in a machine-understandable manner. This additional semantic layer helps

to enable the efficient discovery and the dynamic orchestration of services to build higher-value services or processes.

Several approaches exist to describe web services, their operations and parameter types semantically [7]. The most common semantic web service technologies include SAWSDL, WSMO and OWL-S.

SAWSDL (Semantic Annotations for WSDL) is a W3C recommendation, which describes a lightweight mechanism to add semantics to web services [11]. It is aimed at annotating certain parts of the web service's WSDL descriptions by adding references to semantic models. It does not specify any description language for the referenced semantic model, i.e. arbitrary models can be used such as UML or OWL ontologies. SAWSDL uses the *modelReference* attribute to annotate XML Schema type definitions, element declarations, portTypes, operations, and messages. In addition, the attributes *liftingSchemaMapping* and *loweringSchemaMapping* are used for the mapping between the technical representation of parameter types using XML and the corresponding semantic concepts. In contrast to other approaches such as OWL-S, there is no support for the description of preconditions and effects of a service.

WSMO (Web Service Modeling Ontology) provides a conceptional model for the description of different aspects of a semantic web service [12]. Its corresponding description language WSML (Web Service Modeling Language) is used to model WSMO services in a formal manner. WSMO defines four top level concepts, namely ontologies, goals, services and mediators. Ontologies provide the terminology that is used by the other WSMO elements. Goals describe the task a web service is required to solve. Services provide a semantic description of the web service's functional and behavioral aspects. Mediators are used to resolve mismatches between different WSMO elements.

OWL-S is a technology to describe the semantics of a web service based on OWL ontologies [13]. To achieve this, it consists of three types of knowledge about a web service: what does the service do (ServiceProfile), how does the service work (ServiceModel) and how can the service be invoked (ServiceGrounding). As opposed to SAWSDL, which follows a bottom-up approach to annotate services, OWL-S models the semantic service description on a high level of abstraction as an OWL ontology and then links this description to the concrete WSDL file. This approach is very expressive and makes it possible to describe highly powerful semantics of a service. However, at the same time, this increases the complexity of the description process significantly.

OWL-S not only offers ways to describe single web services semantically, but also to model processes composed of single web services. To this end, OWL-S distinguishes between so called Atomic Processes, Composite Processes and Simple Processes. An Atomic Process can be invoked directly as it is connected to a concrete WSDL file. It corresponds to the operation of a WSDL service. A Composite Process can be composed of Atomic Processes or further Composite Processes. The composition is described by OWL-S control constructs like Sequence, Split, and so on. A Simple Process represents an abstract template of a service, *i.e.*, it depicts a semantic description of a service without concrete linking to a WSDL file. However, it can be assigned to an Atomic Process at runtime using the *realizedBy* reference.

2.4. Service-Oriented Architectures in Production Automation

The semantic discovery and context-based orchestration system described in this article builds upon an architectural approach characterized by the concept of service-oriented architectures in production automation. The state of the art of the application of service-oriented architectures in this field is characterized by the technical realization based on web standards. The two outstanding approaches that dealt with the set-up of service-oriented architectures in production automation arise from the EU-founded research projects SOCRADES and PABADIS'PROMISE. In PABADIS'PROMISE, a control architecture was developed that enables a decision responsibility distribution among acting control entities [14]. The ERP system comprises several classes of business functions that are encapsulated as services. The connection to the agent-based MES system is realized through web services. The SOCRADES consortium developed a web service based communication architecture for the industrial automation domain [15]. The focus was the connection of field devices with high-level control systems like MES and ERP systems. Therefore, the communication technologies DPWS and OPC-UA were used.

The use of the paradigm of service-oriented architectures in the context of industrial automation systems is intended to significantly decrease the effort for integration and programming of automation components. The two basic aspects of this approach are the standardization of communication interfaces and the functional encapsulation of mechatronic functions that enable a high flexibility and an improved control of complexity for automation systems. The basic functions of the manufacturing equipment that execute and monitor the production process are represented as basic services. These basic services can be aggregated to composed services that control the production process.

2.5. Semantic Web Services in Production Automation

With the evolving usage of service-oriented architectures in production systems, the semantic description of services becomes more important. Particularly, within the scope of the SOCRADES project, concepts for the usage of ontologies in the production domain [16] and for the semantic discovery and orchestration of services to production processes [17] have been developed. However, they only show the feasibility of semantic service implementations in simplified setups or simulations. We put emphasis on the implementation of industry-related experimental setups within a real-world research facility, the *SmartFactory*^{KL}. In addition, we believe that clearly defined methodologies for the creation of semantics-supported systems as well as the construction of a fundamental structure of reusable ontologies are necessary in order to make semantic technologies applicable to the production domain.

Concerning the discovery of services, Guinard *et al.* [18] describe a concept for the discovery and selection of real-world services provided by embedded devices. They build upon the results of the SOCRADES project concerning the DPWS networking discovery mechanism. In addition, they discuss a concept for on-demand provisioning of missing services, which depicts an interesting idea for a future extension of our system. Samaras *et al.* [19] describe a concept to integrate resource constrained devices into enterprise information systems. To this end, they propose a discovery mechanism based on semantic annotations. The central topic of their work is the adaption of existing

web service standards to work with resource restricted devices. Both works do not focus on the usage of semantic services within the domain of factory automation as the basis for a context-based orchestration.

Several works dealing with the orchestration of services in factory automation propose a pure syntactical orchestration of services based on BPEL [20,21]. Indeed, these approaches are rather suited for static workflows than for a dynamic orchestration of services [22]. In particular, Ferrándiz-Colmeiro *et al.* [23] propose a concept closely-related to our vision of a context-based orchestration of services to adapt production processes. They describe an orchestration system, which uses ontologies to transform abstract process models to concrete BPEL processes tailored to the capabilities of the respective industrial plant, which are represented as services in a UDDI repository. This work depicts an important basis for the future research on semantic orchestration of services in the production domain. In this article, we build on the results of this approach, but focus on the construction of a fundamental structure of reusable ontologies, the creation of a methodology for the semantic discovery and context-based orchestration of services in factory automation and the implementation of industry-related experimental setups within the *SmartFactory*^{KL}.

3. Semantic Service Description and Context-Based Orchestration

3.1. Use Cases

Manufacturing companies have been under pressure recently due to shorter product life cycles and greater product individualization. In order to stay competitive in this environment, processes and systems need to become more flexible and agile [24,25]. This new requirement poses a great challenge for today's automation systems because changing a production process for example from one product to a new product generation is usually associated with a high amount of programming and configuration. Three exemplary use cases illustrate different aspects of changing production processes and are used to deduce requirements for a dynamic context-based orchestration and the underlying semantic models with the goal of creating flexible automation systems.

The first use case is that of a field device in the production process being replaced with a new possibly better (e.g., resource-efficient) device that can fulfill the same task. This can often be the case in production plants because of component failure or the need to improve some process parameter by installing a better device. With traditional automation systems, this device has to be manually integrated into the programming environment of the PLC (programmable logic controller). This requires the configuration of the communication interface and adaption of the PLC program to the new devices' functional interface (often on signal-level), which depicts a time-consuming process that requires on-line testing to verify the correctness of the new program and thus expensive down-times of machinery. In order to achieve an improvement of this problem, we envision a plug-and-play-like replacement of components that automatically integrates the device into the automation system.

The second use case which describes the lack of flexibility in today's production environments and which we want to address with our approach focusses on the need to change the production process because some component is not available on short notice. In some cases this could be countered with rerouting the current product to another production station (e.g., if there are several equivalent welding robot cells) or changing the sequence of production steps. This ad-hoc reaction to changes in

component availability can keep the production process running and reduce down-time losses. The problem that needs to be solved in order to achieve this dynamic process rerouting is finding a device or production unit that can fulfill the same task or a new sequence of process steps that result in the same outcome.

The third problem that we want to focus on results from the need for greater product individualization because of consumer demand. For each product variant a concrete program needs to be specified for each production station and executed once the product reaches this station. Changes in the product specification result in the need to modify each program at the decentralized stations. Recent research offers a new possibility of dealing with this challenge: a product-mediated communication within a production system [5] allows a product to carry information about the customer order or the status of its production. Extending this idea beyond merely carrying status and order information, a product could also carry an abstract representation of the manufacturing processes needed for its completion. Enabling automation systems to read this representation and automatically choose and execute appropriate concrete process steps for each abstract stage can lead to a greatly improved flexibility in production processes and new possibilities for product individualization.

Having discussed the three use cases, the following requirements can be derived that must be met by a semantic discovery and context-based orchestration system to achieve flexible manufacturing control:

1. An automatic registration of devices in the system must be possible. To this end, a network discovery mechanism is needed as well as a way to uniquely identify the device.
2. To be able to perform a selection of appropriate field device services, the device must know about its capabilities and provided functions, which should be modeled on a semantic level in order to make an automatic discovery possible.
3. Additional information is needed to improve the matching and selection process. For our use cases this includes information about functional parameters, consumed resources and field device types.
4. In order to react to unforeseen events (e.g. device failures), contextual information must be gathered, interpreted and used in the discovery and selection process of device services.
5. Ways to represent abstract descriptions of manufacturing processes are needed, which can be transformed into concrete, directly executable process representations in an easy manner.

3.2. Construction of Ontologies

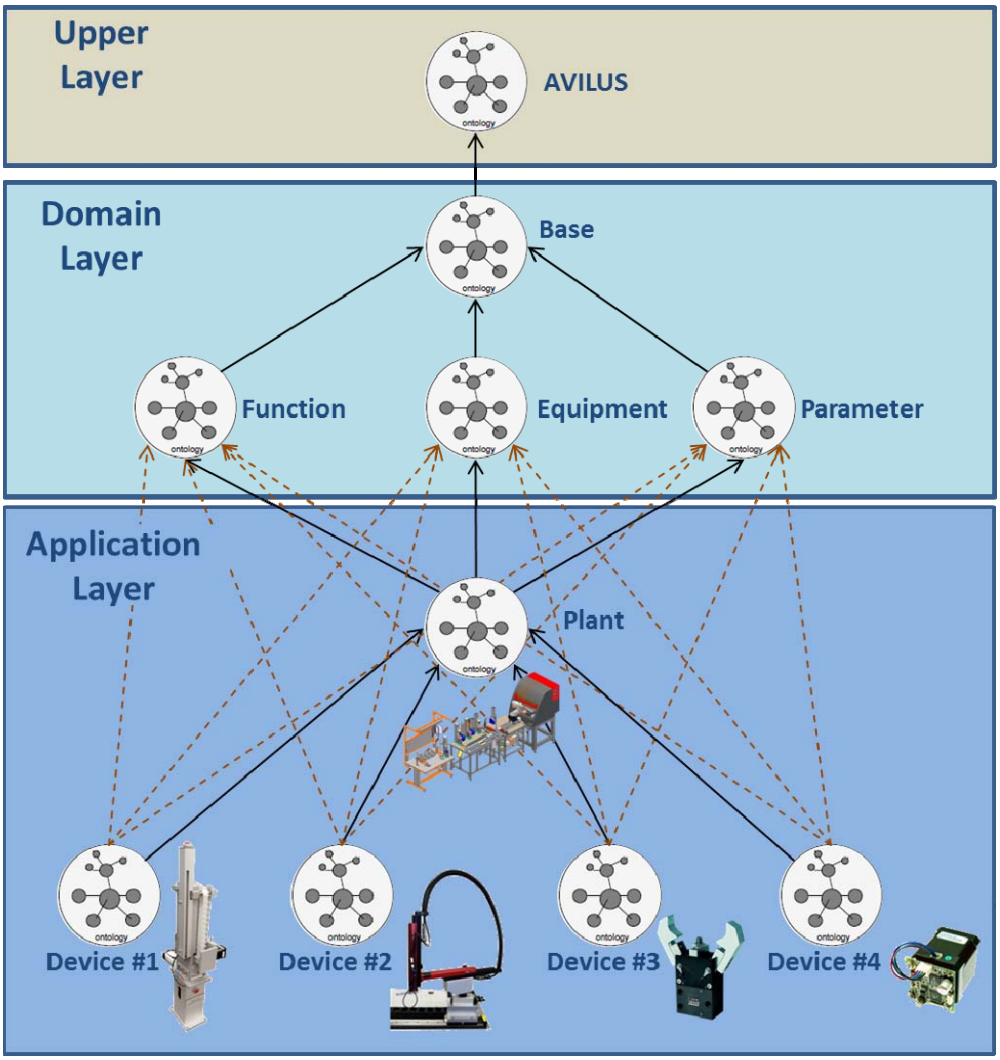
Based on the requirements derived from the use cases described in the last section, we developed a structure of modular ontologies, which depict the basis for the semantic description of services provided by field devices and which are used for semantic reasoning in matching and selection processes. In the following sections, this ontological structure is illustrated and the different contained ontologies are described in detail.

3.2.1. Ontological Structure

According to numerous authors (e.g., [26–29]) the usability and reusability of an ontology can be improved by different levels of abstraction, while the intelligibility and adaptability may benefit from

modularization. The common abstraction is the subdivision in upper ontology, domain ontology, task ontology and application ontology. Due the very vague definition how to really implement a task ontology we did not utilize it in our model, while the other three retained. Figure 1 shows the structure of ontologies, which act as the semantic core models for the semantic service description and context-based orchestration. Each level itself is represented by a collection of smaller sub-ontologies, which are modeled using OWL and Protégé 4.2.

Figure 1. Structure of modular ontologies for the semantic service description and context-based orchestration.



3.2.2. Selection of an Upper Ontology

An upper ontology (or top-level ontology) contains very general knowledge across different domains. By referring to the general concepts of an upper ontology from more concrete concepts of domain and task ontologies, a semantic interoperability can be reached. That is why we investigated several existing upper ontologies, which could act as a common interoperability model for a structure of ontologies used for the semantic description of field device functionalities in assembly and handling. In addition to common upper ontologies like SUMO, Cyc or DOLCE, we considered upper

ontologies that were developed with the objective of using them for applications in the manufacturing domain. There are three ontologies that appeared relevant for an in-depth investigation: MASON, ADACOR and AVILUS.

MASON (Manufacturing's Semantics ONtology) [30] is aimed to provide a common semantic net in the manufacturing domain. Initially, it was used as knowledge base for an expert system used for cost estimation of the production of mechanical parts. This is the reason why MASON has a strongly product-centered point of view as well as detailed conceptualizations in the domain of metal working. The definition of the core concepts "entity", "operation" and "resource" is roughly oriented on the manufacturing decomposition proposed by Martin and D'Acunto [31]. An examination of the OWL files revealed that the concepts provided by the ontology possess only weak to none axiomatic definition, leaving its meaning implicitly defined. However, the axiomatic definition of concepts is one of the main requirements of an upper ontology.

The ADACOR-Ontology, which is based on the DOLCE upper ontology, was developed by Borgo and Leitão as a part of the ADACOR architecture [32]. The scope of the ADACOR-Ontology lies on manufacturing scheduling and control operations, thus providing a sophisticated conceptualization to model operations, process plans, work orders and production plans regarding customer orders and even production forecasts. Several studies have been published [33,34] which prove the axiomatic descriptions of the ontology's core concepts and show the successful application in the ADACOR architecture, where it has been used in a multi-agent manufacturing control system. Unfortunately, the files of the ontology are not openly available. Hence to be applied outside of the ADACOR project, the ontology has to be recreated with the information provided in the documentation. Due to the incompleteness of this published information, especially regarding the axiomatic definitions of most concepts, this is only possible to a limited degree and involves a not negligible amount of effort.

The AVILUS-Ontology [35] is the result of a subproject of the AVILUS joint project, regarding product lifecycle management (PLM) as well as the management of production means with the goal of providing an application- and domain-independent model for information integration. Already existing neutral data formats have shown to be not expressive enough to serve this purpose, while the scope of special PLM formats is usually restricted to products only, excluding means of production. The Ontology is implemented in OWL DL and based on the design of the SUMO (Suggested Upper Merged Ontology) providing a great deal of interoperability. By using a modular design, consisting of a base ontology and several domain specific ontologies for mechanics, electrics, services *etc.*, it ensures the adaptability to the user's needs. Because of its intended use for PLM as well as production means management, there is only one concept, called "entity", to represent both products and the means of production. Thus the ontology may take a product-centered point of view or a facility-centered point of view, depending which fits best for a given application.

Given its characteristics, among the three prior discussed ontologies, the AVILUS-Ontology has shown to be the most suitable one for our use case. Multiple connections between AVILUS concepts and concepts of our domain ontology could be established. For the linking of "device", one of our main concepts, there were two possible mappings in it, "Resource" and "PhysicalTool", both terminal sub concepts of "entity". The decision in favor of "PhysicalTool" is founded in the general axiomatic similarity of both concepts with the difference that in "PhysicalTool" there is already stated, by means of the object property "isToolOf", that a "PhysicalTool" is necessary to exert a specific process. Due to

the mapping of "device" and "PhysicalTool" the obvious consequence is to link our "operation" concept, modeling an action performed by a device, with the concept "process" in AVILUS. At last, all concepts provided in our parameter ontology are linked in some way to the AVILUS "attribute" concept, which is used to describe either a process or an entity in more detail. The concepts of the parameter ontology can roughly be differentiated in two categories, (1) the ones which are used in small ontologies provided by a device to provide information about itself, e.g., the kind of resource which is used and it's normalized consumption of it; and (2) the concepts referred to from the plant ontology in which context-dependent information is stored (e.g., state of a field device). This distinction finds its match in the two specializing concepts "InternalAttribute", modeling intrinsic properties like the information of the first category, and "RelationalAttribute", which describes an entity in the context of its environment, corresponding to the second category. Hence, we achieved a sensible linkage of our ontology to the AVILUS-Ontology, including sub-concepts all of its three core concepts "entity", "process" and "attribute", fully utilizing its semantics from the perspective of our use case.

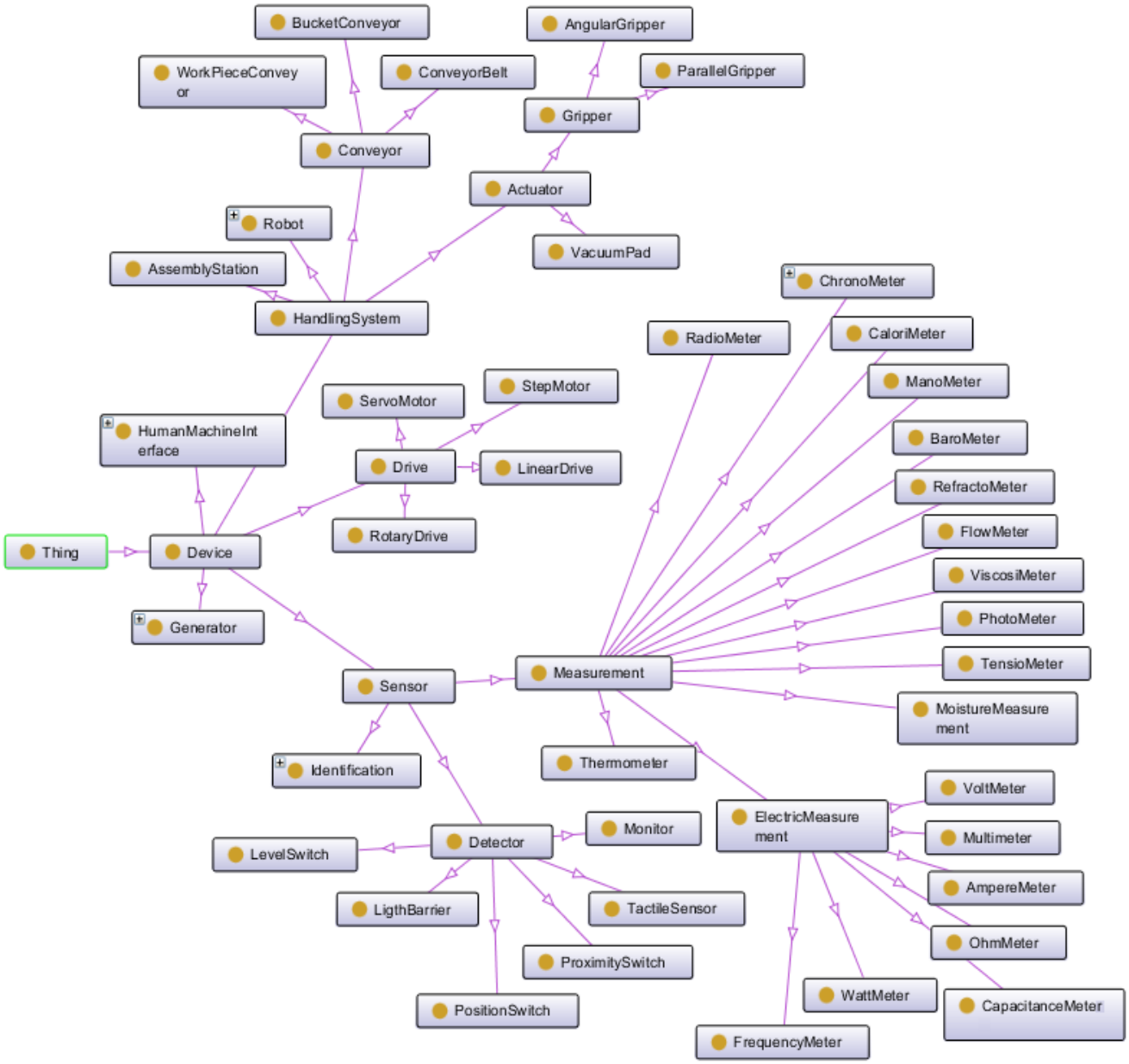
3.2.3. Base Ontology

The base ontology is the core of our domain ontology and serves two basic purposes. First, it provides the framework in which the other modules can be linked together in a way that the whole domain level profits from the interaction of them. To this end, the base ontology integrates further sub-ontologies using the OWL-import mechanism. Its second task is to serve as an interface of the domain level to the upper ontology level. Its head concepts *Operation*, *Device*, *Product* represent the well-known breakdown of manufacturing in the subunits "process", "resource" and "product", but with the special emphasis of devices as resource. The mapping to the AVILUS base ontology is performed using the *owl:equivalentClass* attribute. The complexity of our base ontology is held to a minimum, enabling great flexibility in the design of existing modules and easy addition of further modules.

3.2.4. Equipment Ontology

The equipment ontology contains a collection of device categories (types of devices) in the domain of production automation and manufacturing technology. The device categories are represented in a taxonomic structure. Names as well as the classification of devices in this taxonomy are strongly influenced by the eCl@ss classification and product description, which itself resembles the United Nations Standard Products and Services Code. Despite the heavy influence of eCl@ss on the equipment module it is not just a mere excerpt of it. Instead, we performed several changes in the taxonomic structure itself due to information transferred to the parameter ontology as well as the addition of device categories that are not clearly defined in eCl@ss. Figure 2 shows a segment of the equipment ontology.

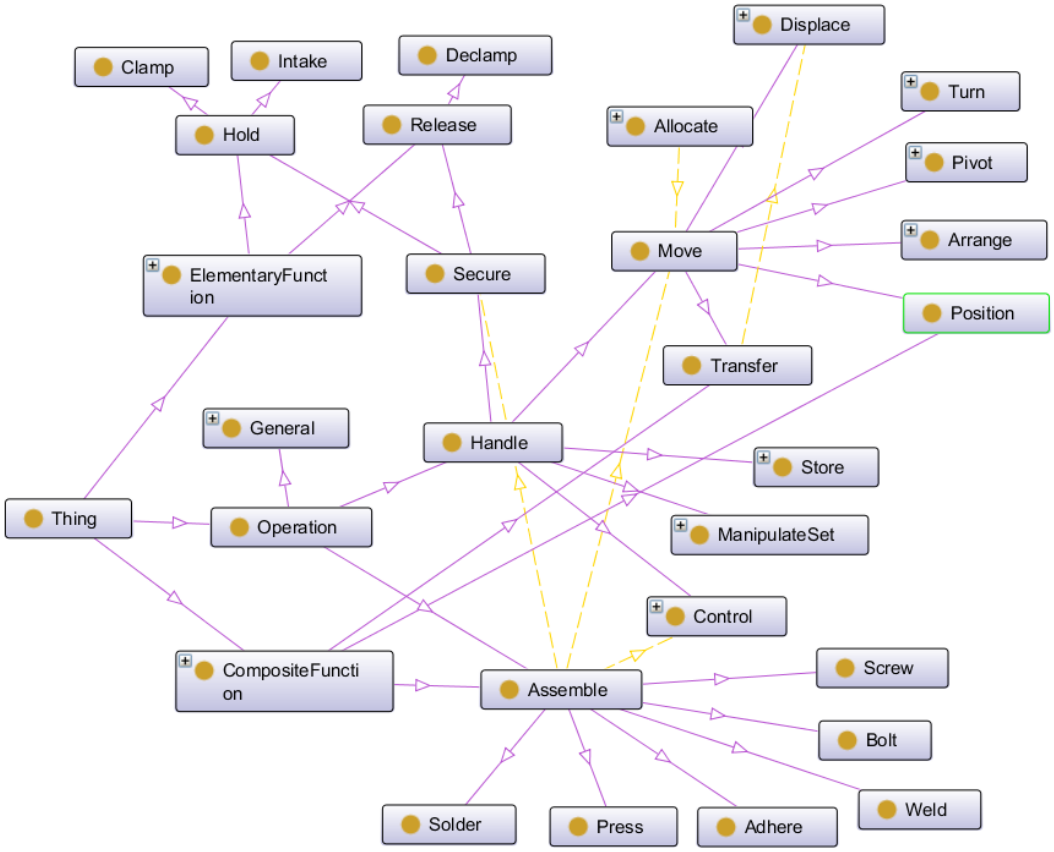
Figure 2. Part of the equipment ontology.



3.2.5. Function Ontology

The function ontology represents a structure of operations which can be performed by technical equipment. Similar to the equipment module it contains a basic taxonomy. On top of the taxonomic structure it provides relational information of aggregated functions. These relations are modeled through the definition of elementary functions, which cannot be derived further by means of functional decomposition, and composite functions, which in contrast can be expressed as an aggregation of elementary functions. The taxonomic structure as well as the functional decomposition is based on the guideline VDI 2860 “Assembly and handling” [36]. Figure 3 depicts a part of the function ontology.

Figure 3. Part of the function ontology.



3.2.6. Parameter Ontology

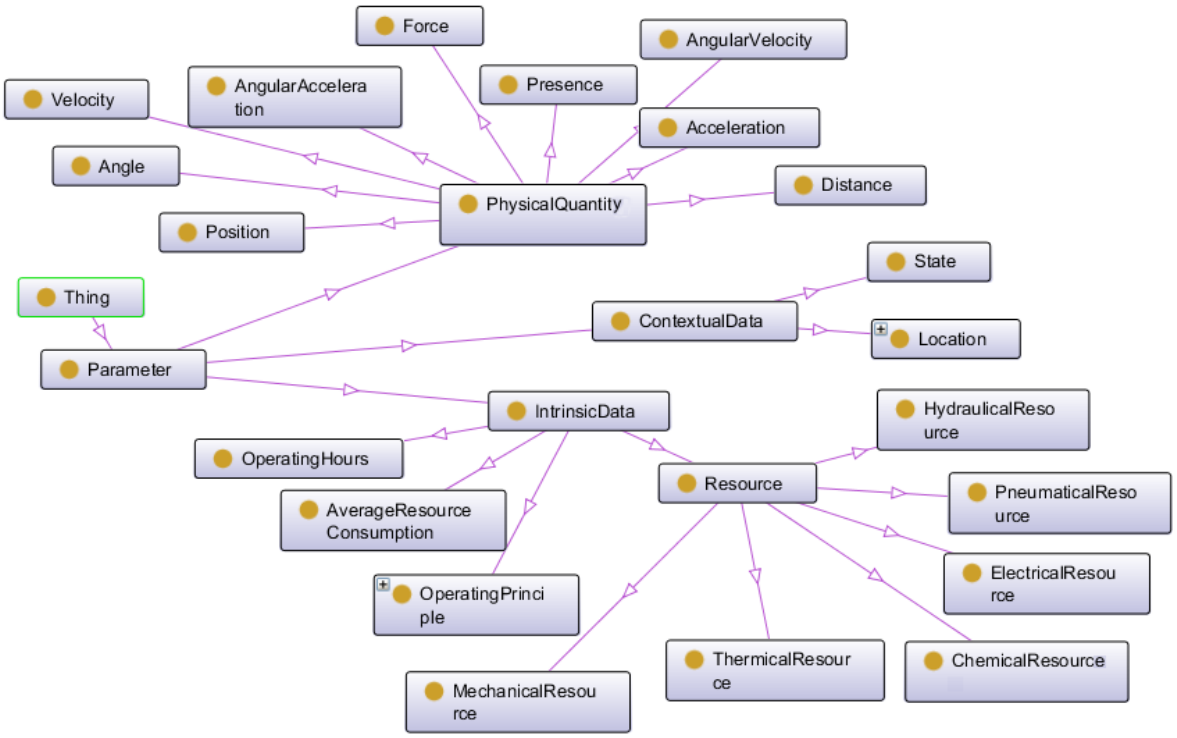
The parameter ontology provides means for the semantic annotation of input and output parameters of a service (e.g., physical quantities like *Angle*, *Velocity etc.*), but also for the description of non-functional properties. These non-functional properties can be divided into three categories: quality of service parameters (e.g., reliability, performance, safety), characteristics which are intrinsic for a device (e.g., the kind of consumed resource, its average resource consumption or its operating hours) and context-dependent information (e.g., location in the plant, current state of the device). Concerning the usage of quality of service parameters for the semantic description of services, one major issue is the definition and measurement of concrete metrics (e.g., "reliability high" or "performance" <1 ms). In order to perform a reasonable matching based on such metrics, experiments must be performed to deliver statistical values. However, this goes beyond the scope of the work described in this article. Nevertheless we modeled different quality of service attributes for future extensions of the system. Figure 4 depicts a part of the parameter ontology.

3.2.7. Plant Ontology

Unlike the already discussed ontologies, the plant ontology depicts an application-specific ontology because it represents the concrete structure of plant or manufacturing line. To this end, the existing field devices and components are represented as instances (imported from the device ontologies, compare Section 3.2.8), which have properties to represent the actual location of the devices in the

plant and the relations (e.g., structural, process-related, physical, electrical) between the devices. Existing knowledge sources such as circuit diagrams or CAD models have been used as a basis for the modeling process.

Figure 4. Part of the parameter ontology.



3.2.8. Device Ontologies

While all other ontologies are unique in our ontological structure, there is a device ontology for each field device or component. The device ontologies contain information about the device category, the operations provided by the device as well as its inherent characteristics. To model this information, each device ontology instantiates the concepts supplied by the ontologies belonging to the domain level and assigns concrete instances or values to the ObjectProperties or DatatypeProperties respectively. To uniquely identify a device, we use a combination of manufacturer, device type and serial number. This information is important for the later automatic registration and discovery of devices and their provided services.

3.3. Semantic Annotation of Web Services

The modeled ontologies described in Section 3.2 can be used for the semantic description of the web services provided by field devices based on OWL-S. Because OWL-S itself is based on OWL, it allows an easy integration with our domain and application ontologies. For each operation of a web service, an OWL-S AtomicProcess is created using Protégé 3.2.1 and the OWL-S Editor plugin. Thereby, the WSDL2OWL-S Tool is used initially to create the basic structure and the grounding of the OWL-S AtomicProcess. In the next step, the OWL-S description of a service operation is connected to the device ontology by referring to the respective device instance using the OWL-S

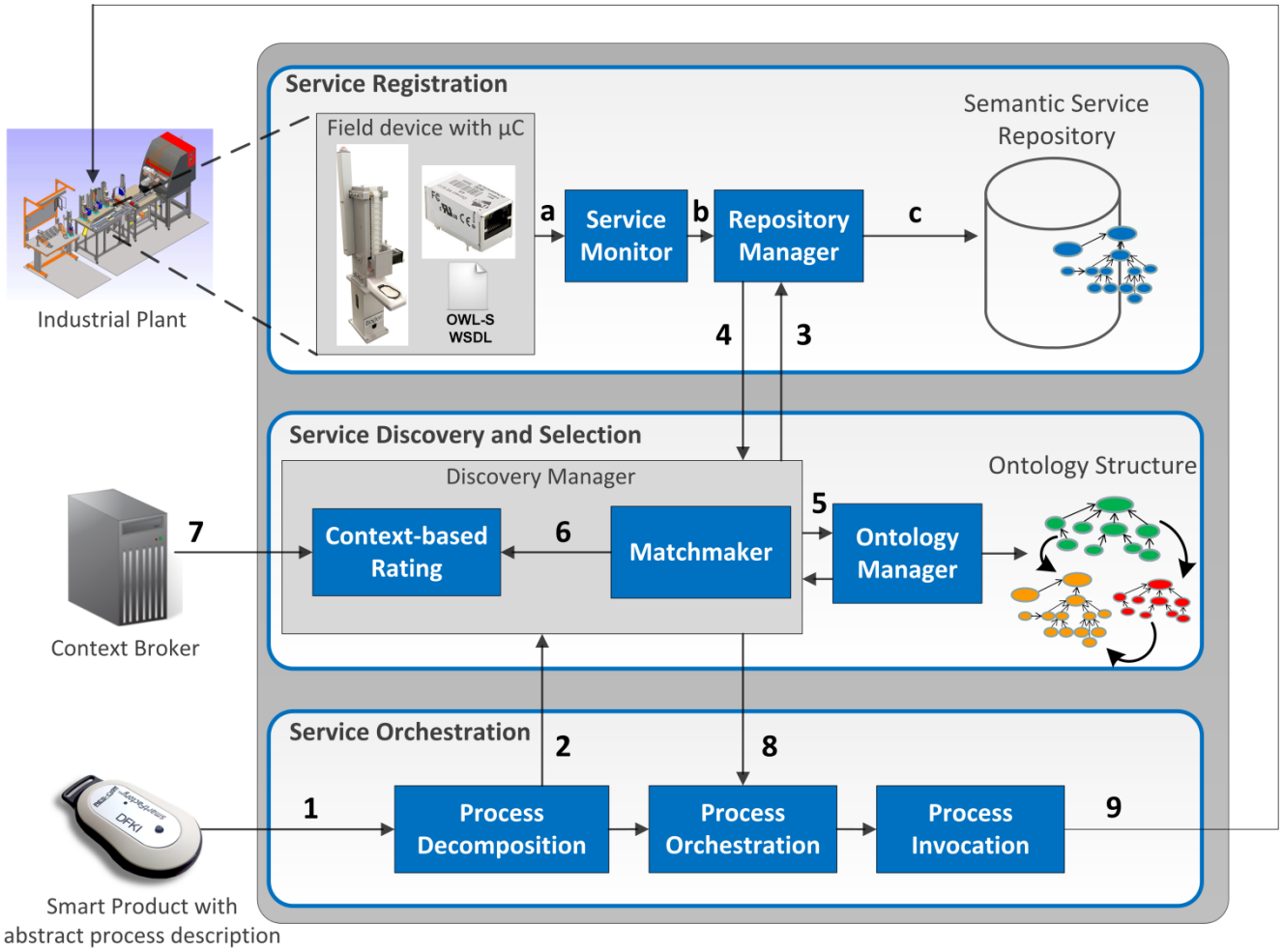
serviceCategory attribute. The semantic description of the input and output parameters is integrated in the OWL-S Grounding, which basically describes a mapping between syntactical descriptions of a web service (e.g., WSDL files) and the parameter concepts of our parameter ontology. The semantic annotation of further functional and non-functional service properties are modeled using the OWL-S serviceParameter reference. By modeling each aspect as a subclass of ServiceParameter, we are able to utilize arbitrary semantic mappings to our sub ontologies. In this way, instances such as the consumed resource or provided function can be referenced, but also numeric values can be assigned (e.g., operating hours of a device).

In addition to the direct annotation of our services in OWL-S, further implicitly assigned knowledge about the service can be used in the later matching, selection and orchestration processes using a reasoning system. For instance, as the OWL-S description of a service is assigned to its corresponding device ontology, which is again connected to the plant ontology, structural knowledge about the plant can be used to determine if the present service is dependent on other services in the plant. The same holds true for domain-specific knowledge modeled in the equipment ontology, for example.

3.4. Service Registration and Semantic Service Repository

The efficient discovery of services provided by field devices depicts the basis for a context-based orchestration for control of manufacturing processes. To this end, mechanisms for the registration and advertising of services on the network level as well as for the capability-based search and discovery of services are needed. For the automatic registration of devices and their services, we make use of the discovery mechanism of DPWS (Device Profile for Web Services) [37], which specifies a reduced web service stack (e.g., WS-Discovery, WS-Eventing) designed for the implementation of web services on resource-constrained devices such as embedded systems. Our approach is based on the implementation of DPWS services on microcontrollers, which are connected to the respective field devices. In addition to the web service implementation and its syntactic interface definition, the semantic description of the service (as discussed in Section 3.3) is stored on these microcontrollers as an OWL file. As soon as a field device is available, it broadcasts a hello message on the network, which is recognized by a java program (Service Monitor) running on a server (step **a** in Figure 5). This program can also send an initial probe message to find all services currently available. The Service Monitor retrieves the semantic description of the service and passes it to the Repository Manager (step **b** in Figure 5), which extracts all information about the service and stores it in the Semantic Service Repository (step **c** in Figure 5). We implemented the Semantic Service Repository based on the OWLIM triple store, which can be used to store and retrieve mass data in an efficient way. In this repository, information about a service is represented as RDF triples. The Repository Manager not only extracts the necessary information from the device ontologies and OWL-S descriptions of a service, but also provides high-level methods for adding and removing services to or from the Semantic Service Repository based on SPARQL queries.

Figure 5. Orchestration system architecture.



3.5. Semantic Discovery and Context-Based Orchestration

Our approach for a context-based orchestration of services is based on the concept of deriving the concrete process used to control the manufacturing process from an abstract process description specific to the product to be produced. This abstract description of the product-specific production process is modeled as an OWL-S CompositeProcess, which consists of abstract OWL-S SimpleProcesses, which do not have groundings to executable web services. These SimpleProcesses could represent either basic operations or high-level functions of our function ontology described in Section 3.2. The abstract process description could be directly attached to the product (e.g., stored on an RFID tag). By doing so, the product carries knowledge about its own production process.

The concrete instance of the process is generated depending on the actual structure of the production plant and the capabilities of its field devices making use of our semantic discovery and service selection system. The orchestration of the concrete process happens ad-hoc, *i.e.*, right at the moment the product enters the production plant. Even more, the orchestration system could adapt the process at runtime (e.g., in case of a faulty component) by finding components that provide an equivalent or similar service.

Figure 5 depicts the system architecture of our context-based orchestration framework, which is divided into three layers: the service registration (already discussed in Section 3.4), the service

discovery and selection and the service orchestration. The numbers represent the steps of the overall orchestration process. In the first step, the abstract process description of the current product (OWL-S CompositeProcess) is passed to the Process Decomposition, which decomposes the CompositeProcess into SimpleProcesses. For each SimpleProcess, the Discovery Manager is asked to find a matching AtomicProcess, *i.e.*, a concrete web service provided by a field device or component in the plant (step 2). To this end, the Discovery Manager requests all the web services that are currently available in the plant (step 3). The Repository Manager queries the Semantic Service Repository to get back all the services including their additional information, which are sent back to the Discovery Manager (step 4). The retrieved set of OWL-S AtomicProcesses acts as input for the Matchmaker (we used iSeM [38] in our implementation), which performs a functional matchmaking based on input/output parameters of the services (step 5). The Matchmaker generates a list of hypothetically matching services (on a functional level) and delivers it to the Context-based Rating Component (step 6). In the next step, contextual information (e.g., current state of products, machines, resource consumptions), which is queried from a Context Broker (compare Section 3.6) is used to influence the rating of the hypothetically matching services. In addition to contextual information, the rating process is influenced by both domain and application knowledge inferred from our different ontologies (e.g., plant ontology, equipment ontology, functional ontology). The Rating Component assigns weights to different matching criteria (e.g., provided operation, equipment category, consumed resource, Quality of Service attributes) and calculates a total score for each service. The service with the highest score is selected and sent to the Process Orchestration (step 8), which performs a re-composition of the process taking the input/output annotations of the single services but also knowledge from the function ontology and the equipment ontology into account. The procedure from step 1 to 8 is repeated until a concrete AtomicProcess is found for each SimpleProcess contained in the abstract process description. In the last step (step 9), the resulting concrete OWL-S CompositeProcess is forwarded to the OWL-S Engine, which invokes the contained services following the respective control constructs in order to control the manufacturing process for the present product in the plant.

The developed system for semantic discovery and context-based orchestration can not only be used to realize an ad-hoc orchestration tailored for the present product (or product variant), but also for the plug-and-play integration of new field devices and the adaption of the manufacturing process in response to unforeseen events (e.g., device failures, changed requirements with respect to resource consumption). The former use case can be handled by the registration of the newly integrated service by means of its semantic description. The latter one is addressed with the context-based rating and selection of services and with the possibility to perform a re-orchestration at runtime (if necessary).

3.6. Context Interpretation for Service Rating and Selection

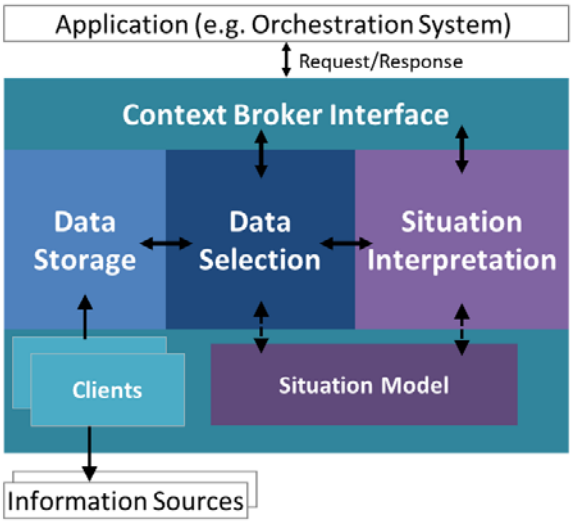
The classical definition, as proposed by [6], describes context as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves." This definition from the field of context-aware mobile computing research needs to be adapted to reflect the different focus of context-awareness in the industrial domain and its use in the orchestration of web services to create flexible production processes: not the interaction between a user

and the current application is the most important aspect but rather the production process and the associated task that needs to be fulfilled. All production entities that are relevant to the execution of this task, either by actively participating in the process or supplying relevant input (e.g., material, information) can supply information about themselves, providing the context of the actual process and task.

In an additional processing step, these separate pieces of context information can be aggregated to a so-called situation using predefined knowledge and rules. A situation is always tightly associated with an object and focused on a certain question (situation of X with regard to Y, e.g., "situation of production unit 2 with regard to energy" or "situation of robot cell with regard to safety"). Using situations offers a better way of integrating the contextual information into applications without having to deal with each piece of information separately. Furthermore, the logic of how a specific situation is determined is not implicitly contained in the application's code but is represented explicitly in a re-usable and extendable ontology.

A dynamic adaptation of a production process to achieve a certain optimization goal (e.g., resource consumption) can only be realized if the actual situation of the process and its components is known in as much detail as possible. Using only information directly from the process often does not accurately represent the actual status with respect to the orchestration goal, leading to not optimal decisions. To achieve this context-based process orchestration a central server (so-called Context Broker) serves as a provider of the context information and the resulting situation made-up by the exact pieces of context information (see Figure 6).

Figure 6. Structure of Context Broker used by orchestration system.



This Context Broker uses different client libraries to continuously call various information sources within the production environment (e.g., via OPC UA), read the enabled variables and store the values in the internal data storage. Applications (in this case the orchestration system) can request the current situation with regard to a certain question by making a HTTP request to the context broker's RESTful interface. The interpretation component then selects all context information that is relevant to determine the requested situation type from the internal database and inserts the current values into the situation ontology. This ontology contains concepts for all specified situations and rules describing

which combination of separate pieces of information make up these situations. Using a reasoning algorithm on the ontology with the current values, the current situation can be classified and served back to the application as a response to the HTTP request. Alternatively, just the set of relevant context information can be sent back to the client for evaluation.

The proposed orchestration system uses the Context Broker as a source of information during the service rating process to yield more accurate results regarding the appropriateness of a concrete web service for a certain process step. In addition, the orchestration system itself subscribes to the interpretation of the current plant situation by the Context Broker. This approach guarantees a tight integration of the orchestration process with the actual physical environment and its status.

4. Experimental Setup

In order to test and evaluate our concepts and technologies, we developed an industry-related experimental setup as part of the demonstration facility *SmartFactory*^{KL} [39]. The automated assembly station (Figure 7), in which the concepts described in Section 3 have been implemented, is part of a plant that produces an intelligent product, the so called SmartKeyFinder (Figure 8), which can be called via a smart phone app when having lost your keys. The SmartKeyFinder consists of three components: a casing cover, a casing bottom and a printed circuit board, which is equipped with a LED, a loudspeaker, a button and a Bluetooth module.

The material flow in the automated assembly station is ensured by an intelligent work piece carrier which has three places for casing covers available. Inside of the work piece carrier, there are three light sensors to detect if a new casing cover is loaded and an RFID reader used to read data from the RFID transponders attached to the casing covers. On these RFID transponders, information about the customized production order is stored. Both light sensors and RFID reader are connected to an embedded system, which communicates the determined order information via WLAN to our orchestration system.

The pick and place system located in the middle of the station successively picks the product parts (casing bottom, circuit board) out of their storages and places them in the respective assembly module. There are three assembly modules—two of them pneumatically operated; the third with electrical drive. In the next step, the casing bottom is fixed with a gripper, the circuit board is placed into the casing bottom and the casing cover is pressed on it, which completes the product assembly.

The several field devices (sensors and actors) contained in this experimental setup provide their functionality as DPWS web services, which are hosted on microcontrollers (Digi Connect ME 9210) as gateways. In addition to the syntactic DPWS definition by means of WSDL files, an OWL-S description is stored for each functionality provided by the respective field devices. In addition, each field device brings its own device ontology as an OWL file. Both the orchestration system and the semantic service repository are running on a server connected to the plant network.

Figure 7. The automated assembly module.

Figure 8. The intelligent sample product: the SmartKeyFinder.

Concerning the use cases described in Section 3.1, we implemented the ad-hoc orchestration and control of the manufacturing process based on an abstract process description and on the customer order (e.g., assembly optimized for delivery reliability or for saving resources). On the RFID transponder attached to the product, the URL of the respective abstract process description (OWL-S CompositeProcess consisting of SimpleProcesses) is stored. As soon as the product enters the plant,

the intelligent work piece carrier delivers the order information and the URL to the abstract process description to the orchestration system. After that, the different processing steps described in Section 3.5 are performed. For functional matchmaking iSeM is fed with the respective SimpleProcess (request) and all available AtomicProcesses from the OWLIM repository (offers). In this step, the parameter ontology described in Section 3.2.6 is used. The list of hypothetically matching services produced by iSeM is forwarded to the Rating Component, which performs a rating of the different services based on further functional and non-functional information (e.g., operation name, consumed resource, state of the device). For each piece of information, a matching is performed on the respective ontologies (e.g., function ontology, parameter ontology, equipment ontology). Depending on the assigned weight, the matching of the respective piece of information has a different influence on the overall rating of the service (e.g. operation name has a higher weight than the consumed resource). By doing so, we are able to consider customer wishes (e.g., resource-saving production), but also the functionalities provided by the different field devices of the plant as well as their state and availability during the orchestration process.

In the near future, we are going to implement the use case concerned with the re-orchestration of the manufacturing process as reaction to the failure of a device. More precisely, the pneumatic press in the first assembly module will show an increased consumption of compressed air, which is interpreted as a critical situation by the Context Broker and which causes the orchestration system to replace the service of assembly module 1 by a service of another assembly module. Even more, we are going to demonstrate the plug-and-play replacement of assembly module 3 by a new, resource-efficient assembly module.

In order to prove the benefits of a context-based orchestration system, an experimental comparison with state-of-the-art frameworks (e.g., conventional PLC) must be performed. We are going to consider respective evaluation processes in the future.

5. Conclusions and Future Work

This article describes how semantic technologies can be applied to the application field of production automation. It demonstrates the basic technical feasibility to execute technical processes in the domain of automation using internet technologies which were not initially developed for this domain. Furthermore, it shows that based on the modeling of information and knowledge the adaptability of technical processes can be increased significantly. The presented approach has the potential to offer benefits like seamless information integration, interoperability of factory and business IT systems and highly agile manufacturing equipment.

Semantic technologies are very well suited to realize interoperable information exchange. The presented work demonstrates the realization of context-based orchestration of semantic services on a real world example. The knowledge about equipment, functional characteristics, function parameters, plants and devices has been modeled to some extend in several works, but mostly in an informal manner. Besides existing upper ontologies and domain ontologies knowledge about factory concepts is written down in standards and guidelines. In this work those existing knowledge sources are used to create a modular structure of ontologies, which can be extended and combined to a great extent. Following this approach interoperability and reusability are ensured.

However, within the domain of industrial control the typical requirements to control systems are availability, reliability, determinism and real time behavior in terms of milliseconds. Amongst others, these are certainly reasons why programmable logic controllers with signal-based point of view and the need for extensive engineering still dominate this domain. Due to limited interoperability of those controllers the domains of automation, business IT and the internet are still strictly divided. On the technical level one promising approach to overcome this barrier is to rethink the system architecture of factory IT. While today this architecture has a strict hierarchical structure with centralized control structures, future architectures based on the paradigm of decentralization and openness can help to fulfill the classic demands to automated control systems while at the same time providing interoperability to open systems like the internet. Future work will have to focus on the reliability and predictability of dynamic context-based orchestration in manufacturing systems. Since in the presented approach open internet technologies are used, IT security and operational safety are further open issues.

Realizing context-based orchestration of manufacturing processes the transparency of the equipment's behavior for humans is difficult to keep. Since the decision how to parameterize processes and which service to invoke is taken automatically based on context information, it is very hard to understand a machines behavior for a human. This leads to the question of the human's role in a future smart factory. Here we should look back to the first attempts to use computer technology in factories in the beginnings of the 1980's. Fully automated plants should solve cost and quality problems on the basis of state-of-the-art computer technology. This led to production systems that were extremely complex in planning as well as in construction, operation, and maintenance and in the end did not fulfill the hopes. The solution was to drastically decrease technical complexity and put the humans back into the center of factories [40].

In future factories the humans should be the conductor of dynamically allocated production resources. So in future work the use of information about processes, equipment, functions, *etc.*, for assistance systems for human operators will have to be focused. Even work instructions for manual work could be generated dynamically on the basis of context-based dynamic orchestration placing the human worker into the center of the value creation process.

The realized concept shows how the transfer of internet technology into the domain of production automation realizes opportunities to cope with the challenge given by current megatrends. However besides all possibilities artificial intelligence offers we should remember past experiences and put the human into the center of responsibility.

Acknowledgments

This research was funded in part by the German Federal Ministry of Education and Research under grant number 01IA11001 (Project "RES-COM"). The responsibility for this publication lies with the authors.

References

1. Zuehlke, D. Smartfactory—Towards a factory-of-things. *Annu. Rev. Control* **2010**, *34*, 129–138.

2. Ollinger, L.; Schlick, J.; Hodek, S. Leveraging the agility of manufacturing chains by combining process-oriented production planning and service-oriented manufacturing automation. In *Proceedings of the 18th IFAC World Congress*, Milan, Italy, 28 August–2 September 2011.
3. Jammes, F.; Smit, H. Service-oriented Paradigms in Industrial Automation. *IEEE Trans. Ind. Inform.* **2005**, *1*, 62–70.
4. Loskyll, M.; Schlick, J.; Hodek, S. Semantic service discovery and orchestration for manufacturing processes. In *Proceedings of 16th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2011)*, Toulouse, France, 5–9 September 2011.
5. Stephan, P.; Meixner, G.; Koessling, H.; Floerchinger, F.; Ollinger, L. Product-mediated communication through digital object memories in heterogeneous value chains. In *Proceedings of IEEE International Conference on Pervasive Computing and Communications (PerCom)*, Mannheim, Germany, 29 March–2 April 2010; pp. 199–207.
6. Dey, A.K. Understanding and using context. *Pers. Ubiquitous Comput.* **2001**, *5*, 4–7.
7. Bhiri, S.; Gaaloul, W. Semantic web services for satisfying SOA requirements. *Adv. Web Semant. I* **2009**, *4891*, 374–395.
8. Gruber, T.R. A translation approach to portable ontology specifications. *Knowl. Acquis.* **1993**, *5*, 199–220.
9. Smith, M.K.; Welty, C.; McGuinness, D.L. *OWL Web Ontology Language Guide*; World Wide Web (W3C): Cambridge, MA, USA, 2004.
10. Manola, F.; Miller, E. *RDF Primer*; W3C Recommendation; World Wide Web (W3C): Cambridge, MA, USA, 2004.
11. Kopecký, J.; Vitvar, T. Sawsdl: Semantic annotations for WSDL and XML schema. *IEEE Internet Comput.* **2007**, *11*, 60–67.
12. Roman, D.; Keller, U.; Lausen, H. Web service modeling ontology. *Appl. Ontol.* **2005**, *1*, 77–106.
13. Martin, D.; Burstein, M. Bringing semantics to web services with OWL-S. *World Wide Web* **2007**, *10*, 243–277.
14. *Structure and Behaviour of a PABADIS'PROMISE System*; White Paper for PABADIS'PROMISE Consortium: Magdeburg, Germany, 2008; Available online: http://www.uni-magdeburg.de/iaf/cvs/pabadispromise/dokumente/p2_whitepaper1_final.pdf (accessed on 9 August 2012).
15. Souza, L. A web service based shop floor infrastructure. In *Proceedings of Internet of Things Conference*, Zurich, Switzerland, 2008; pp. 50–67.
16. Lastra, M.; Delamer, I. Ontologies for production automation. *Adv. Web Semant. I* **2009**, *4891*, 276–289.
17. Lastra, M.; Delamer, I.M. Semantic web services in factory automation: Fundamental insights and research roadmap. *IEEE Trans. Ind. Inform.* **2006**, *2*, 1–11.
18. Guinard, D. Interacting with the SOA-based Internet of things: Discovery, query, selection, and on-demand provisioning of web services. *IEEE Trans. Serv. Comput.***2010**, *3*, 223–235.
19. Samaras, I.K. Utilizing semantic web services in factory automation towards integrating resource constrained devices into enterprise information systems. In *Proceedings of Emerging Technologies and Factory Automation (ETFA'09)*, Mallorca, Spain, 22 September 2009; pp. 610–617.
20. Jammes, F. Orchestration of service-oriented manufacturing processes. In *Proceedings of Emerging Technologies and Factory (ETFA'05)*, Facolta' di Ingegneria, Catania, Italy, 19–22 September 2005.

21. Puttonen, J.; Lobov, A.; Martinez, J.L. An application of BPEL for service orchestration in an industrial environment. In *Proceedings of Emerging Technologies and Factory Automation (ETFA'08)*, Hamburg, Germany, 15–18 September 2008; pp. 530–537.
22. Ren, W. Searching for service-oriented strategies of dynamic composition of web services: A comparative perspective. In *Proceedings of 33rd Annual Conference of the IEEE Industrial Electronics Society*, Taipei, Taiwan, 5–8 November 2007; pp. 2615–2620.
23. Ferrándiz-Colmeiro, A.; Gilart-Iglesias, V.; Maciá-Pérez, F. Semantic processes modelling independent of manufacturing infrastructures. In *Proceedings of Emerging Technologies and Factory Automation (ETFA'10*), Bilbao, Spain, 13–16 September 2011; pp. 1–8.
24. Leitão, P. Agent-based distributed manufacturing control: A state-of-the-art survey. *Eng. Appl. Artif. Intell.* **2009**, *22*, 979–991.
25. Jovane, F.; Westkaemper, E. *The ManuFuture Road—Towards Competitive and Sustainable High-Adding-Value Manufacturing*; Springer: Berlin, Germany, 2009; pp. 149–163.
26. Chandrasekaran, B.; Johnson, T.R. Generic tasks and task structures: History, critique and new directions. In *Second Generation Expert Systems*; David, J.M., Krivine, J.P., Simmons, R., Eds.; Springer: New York, NY, USA, 1993; pp. 232–272.
27. Russ, T.; Valente, A.; MacGregor, R. Practical experiences in trading off ontology usability and reusability. In *Proceedings of 12th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada, 16–21 October 1999.
28. Borst, W.N. Construction of Engineering Ontologies for Knowledge Sharing and Reuse. Ph.D. Dissertation, Centre for Telematics and Information Technology, University of Twente, Enschede-Noord, the Netherlands, 1997.
29. Jarrar, M.; Meersman, R. Scalability and knowledge reusability in ontology modeling. In *Proceedings of the International Conference on Infrastructure for e-Business, e-Education, e-Science, and e-Medicine SSGRR2002*, L'Aquila, Italy, 28 July–4 August 2002.
30. Lemaignan, S.; Siadat, A.; Dantan, J.Y. MASON: A proposal for an ontology of manufacturing Domain. In *Proceedings of IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications*, Prague, Czech Republic, 15–16 June 2006; pp. 195–201.
31. Martin, P.; D'Acunto, A. Design of a production system: an application of integration product-process. *Int. J. Comput. Integr. Manuf.* **2003**, *16*, 509–516.
32. Borgo, S.; Leitão, P. Foundations for a core ontology of manufacturing. *Ontologies* **2007**, *14*, 751–776.
33. Leitão, P.; Colombo, A.W. Formal specification of holonic control system ADACOR product holon, using high-level Petri nets. In *Proceedings of IEEE International Conference on Industrial Informatics (INDIN'03)*, Alberta, Canada, 21–24 August 2003; pp. 263–272.
34. Leitão, P.; Restivo, F. Experimental validation of ADACOR holonic control system. *Holonic Muli-Agent Syst. Manuf.* **2005**, *3593*, 121–132.
35. Müller, A.; Hamadou, M. Informationen im PLM-Prozess (in German). In *Virtuelle Techniken im Industriellen Umfeld: Das AVILUS-Projekt—Technologien und Anwendungen*; Schreiber, W., Zimmermann, P., Eds.; Springer: Berlin, Germany, 2011; pp 19–31.
36. *Assembly and Handling; Handling Functions, Handling Units; Terminology, Definitions and Symbols* (in German); VDI 2860; Verband Deutscher Ingenieure: Berlin, Germany, 1990.

37. *OASIS Devices Profile for Web Services (DPWS)*, Version 1.1; OASIS Standard: Burlington, MA, USA, 2009.
38. Klusch, M.; Kapahnke, P. iSeM: Approximated Reasoning for adaptive hybrid selection of semantic services. In *Proceedings of the 4th International Conference on Semantic Computing (ICSC)*, Pittsburgh, PA, USA, 22–24 September 2010.
39. Zühlke, D. SmartFactory—From vision to reality in factory technologies. In *Proceeding of 17th International Federation of Automatic Control World Congress*, Coex, Korea, 2008; pp. 82–89.
40. Womack, J.; Jones, D. *The Machine that Changed the World: The Story of Lean Production*; Harper Perennial: New York, NY, USA, 1991.