Ontology-Based Approach to Decentralized Production Control in the Context of Cloud Manufacturing Execution Systems

Vom Fachbereich Maschinenbau und Verfahrenstechnik
der Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

genehmigte

Dissertation

von
Herrn
M.Sc. Badarinath Katti
aus Dharwad, Indien

2020

Promotionskommission:
Vorsitzender : Prof. Dr.-Ing. Jens Göbel
1. Berichterstatter : Prof. Dr.-Ing. Martin Ruskowski
2. Berichterstatter : Prof. Dr. Klaus Schneider

Tag der mündlichen Prüfung: 05.05.2020

D 386

Preface

This research work is supported by a doctoral grant from SAP SE. I would like to sincerely thank Veronika, for giving me this opportunity to work on my thesis, and also work in a wonderful "dream" team. I would like to sincerely thank PCo Team collegaues - Christian, Christine, Emil, Edgar, Hans-Juergen, Kathryn, Maria-Anna, Martin, Michael, Nalini, Rüdiger, Salih and Thomas. I have thanked Michael n number of times, and I still believe this number is very small, and he believes the opposite!

The thesis research was also carried out at the $SmartFactory^{KL}$ in German Research Center for Artificial Intelligence. I would sincerely like to thank Professor Zühlke for considering my profile admissible to work in his research team. A big thank you to Professor Ruskowski for being my doctoral supervisor, from instilling the confidence since our first meeting we had in the mid June 2017 to giving me the freedom to work on my research throughout the process. I could not have asked for a better Doktorvater! I cannot thank enough Professor Schneider, who was my second Doktorvater. I have always considerd him a father figure, and cannot list here all things he has done for me over the course of my stay in this foreign land. I would sincerely like to thank all the colleagues of $SmartFactory^{KL}$, who have helped me grow academically over the past few years. It would be unfair if I did not mention the folllowing names: Rüdiger, Rüdiger, Marius, Simon, and Rouven. Penultimately, Christiane! This thesis would not have materialized if not for her constant (moral, sometimes!) support, collaboration and dogged perseverance. I hope she knows that!

Lastly, my family - Pappa, Amma, Venu, Anusha, Samanvita, Kaka(s), Kaku(s), Atya, Mama, Vatsa, Pravalika, Janhavi, Raghu, Ramya, Sony, Mini.. (a big one at that, hence cannot name everyone here due to space constraints!), without which nothing would have been possible.

Each one of you is an inspiration for me in his/her own way. Thanks for being kind, considerate and patient with me all these days, months and years. I hope and aspire to be as good a researcher, programmer, leader and wonderful human being as you all are.

Kaiserslautern, May, 2020
Badarinath Katti

Contents

Preface **III**

Abstract **VII**

1 Introduction, Problem Statement and Overview **1**
1.1 Problem Statement 4
1.2 Establishing the boundaries of research 10
1.3 Solution Methodology and Organization of Thesis Manuscript 11

2 State of the Art **15**
2.1 Manufacturing Execution Systems 15
2.1.1 Enterprise Business Solutions in Manufacturing: Motivation for MES 15
2.1.2 Objectives of MES 17
2.1.3 MES Deployment Options 19
2.1.4 Previous Works in Cloud Manufacturing 22
2.2 Production Control Strategies 24
2.2.1 Centralized and Decentralized Control Systems 26
2.2.2 Well-Known Decentralized Architectures for Production Control . . . 27
2.3 DIN SPEC 91345: Reference Architecture Model for Industrie 4.0 34
2.3.1 Industrie 4.0 Component (I4.0 Component) 35
2.3.2 RAMI4.0 Communication Layer: IEC 62541 OPC UA Standard . . . 37
2.3.3 Status-Quo Evaluation 40
2.4 Service Oriented Manufacturing and Semantic Web Technologies 42
2.4.1 Service Oriented Architecture 42
2.4.2 Semantic Web 43
2.4.3 Semantic Web Services (SWS) 45
2.4.4 Previous Works 50

3 System Architecture **53**
3.1 Caching of Production Scheduling and Control Information 53
3.1.1 Characteristics of GeSCo 54
3.1.2 Anticipated challenges due to caching 55

3.2 Proposed System Architecture 58

4 Semantic Modeling 65
4.1 Assimilation of OWL-S Framework in OPC UA 69
4.1.1 Modified OWL-S Sub-Ontologies 70
4.1.2 Working Principle of OWL-S Augmented OPC UA 76
4.2 Assimilation of SAWSDL Specification in OPC UA 79
4.2.1 Organization of Application-Specific Method Nodes in OPC UA Server Address Space 80
4.2.2 Exploitation of OPC UA Node structure 81
4.2.3 Assumptions, conventions and Scope Definition 86
4.2.4 Publishing OPC UA Methods Grounding Metadata 87
4.2.5 Working Principle of SAWSDL Augmented OPC UA 89
4.3 Incorporation of a Hybrid of OWL-S Framework and SAWSDL Specification to OPC UA 90
4.3.1 Motivation for the Hybrid Approach 90
4.3.2 Methodology 94

5 Implementation 97
5.1 Classification of Rulesets 99
5.2 Integration of Ontology to Cloud based MES 103
5.3 Automatic Encoding of the Ontology 106
5.4 Creation of Ontology Resolution and Business Rules in SWRL 112
5.5 Centralized and Semi-Automatic Modeling of Capabilities of Manufacturing Resources 115
5.6 Ontology as Placeholder for Manufacturing Data Collection 118

6 Experimental Evaluation 120
6.1 Infrastructure Set-Up 120
6.2 Production Design Time Activities 123
6.3 Method Discovery and Orchestration 130
6.3.1 Use case 1: Quality Control 135
6.3.2 Use case 2: Resource Breakdown 136
6.4 Open Integrated Factory - Generation 2017: Supplementary Use Case 136
6.5 Implementation Considerations, Challenges, Results and Lessons Learned . . 137

7 Discussion: Summary and Outlook 141

Bibliography 146

Abstract

Decentralized manufacturing is an active research topic in current smart and open integrated factories, and is probably also the future state of practice in both the process and manufacturing industries. Decentralized systems are known to reduce the communication effort, increase flexibility, autonomy, adaptability and efficiency of the industrial automation systems. Meanwhile, the Manufacturing Execution System (MES) is a comprehensive automation software solution that coordinates all the responsibilities of modern production systems. Further, when the advantages and downsides of various MES offerings are explored in anticipation of changing production environments, the Cloud MES (CMES) emerges as the most flexible and affordable solution. The MES solution is essentially designed as a centralized manufacturing control unit, which goes against the principle of the decentralized manufacturing paradigm. Additionally, when operated as a cloud based solution, the MES faces another big challenge: connectivity and network latency.

To address these problems, this thesis introduces an edge layer called Generic Shop-Floor Connector (GeSCo) between CMES and shop-floor. In order to accomplish the assigned manufacturing tasks effectively, the edge layer is required to possess contextual awareness to make run-time decisions in production. Semantic technologies, on the other hand, assist in discerning the meaning, reasoning and drawing inferences from the data. To that end, the well-known Web Ontology Language for Web Services (OWL-S) framework and Semantic Annotation for Web Service Definition Language (SAWSDL) specification from the domain of semantic web services are incorporated to OPC UA application specific methods. Subsequently, a hybrid approach which is a combination of OWL-S and SAWSDL specifications is proposed. In order to enable market-like demand-supply synergy employing these semantic frameworks, abstract services in the production order and concrete services provided by the field devices on the shop-floor need to refer to a common production ontology. The creation of this ontology is a challenge of its own. This research also proposes a pragmatic automation of an encoding of a primary and light weight production ontology based on the source code of MES. The delegation of manufacturing runtime control by the CMES to this edge layer consequently facilitates dynamic manufacturing service discovery and adaptive orchestration plans. Furthermore, the derived hybrid approach is applied to a real use case to demonstrate its feasibility in industrial environments.

1 Introduction, Problem Statement and Overview

From the perspective of manufacturing industries, enterprise information systems are comprised of several business applications such as Enterprise Resource Planning (ERP), Product Lifecycle Management (PLM) and Manufacturing Execution Systems (MES) [Arab17] (see Figure 1.1). There are other business-oriented tools such as Business Process Management (BPM) and Human Resource Management (HRM) that either exist independently in some manufacturing facilities or their functionality is subsumed by other well-known applications such as ERP.

In an integrated manufacturing environment, an ERP maintains an enterprise database where each business transaction is entered, recorded, processed, monitored and reported. This in turn, presents an unified enterprise view of the business encompassing all the departments and their corresponding functions. ERP encompasses the areas of product design, operations and logistics, sales and marketing, information warehousing, materials planning, human resources, finance and project management [Umbl03]. On the other hand, PLM is a business software that manages all the data associated to a product during its lifecycle phases which include design, manufacturing, usage, maintenance, recycling and disposal [MATS10]. PLM is often referred to as a "single system of record" for product data across the product lifecycle. As maturity [Paul93] increases, product and process information flows from PLM to downstream layers such as ERP and MES [Grea17]. In manufacturing companies, both PLM and ERP complement each other. The PLM software enables the engineering design process [Haik18], and when the design is approved, ERP ensures quality products are manufactured in a time-bound and cost-controlled manner [Hill]. Both ERP and PLM business software comprise layer 4 of the automation pyramid.

Although ERP achieves better integration with regards to flow of information between business functions in and across the organizations, it alone cannot meet the intricate challenges presented by the Mass Customization in Production (MCP) [Silv01]. ERP concentrates on the managerial level of decision-making, and its shop-floor supervision is relatively weak to support frontline workers and supervisors [Zhon13, Yang16]. [Zhon13] also states that ERP needs real-time data to generate optimal results. However, both ERP (level 4) and

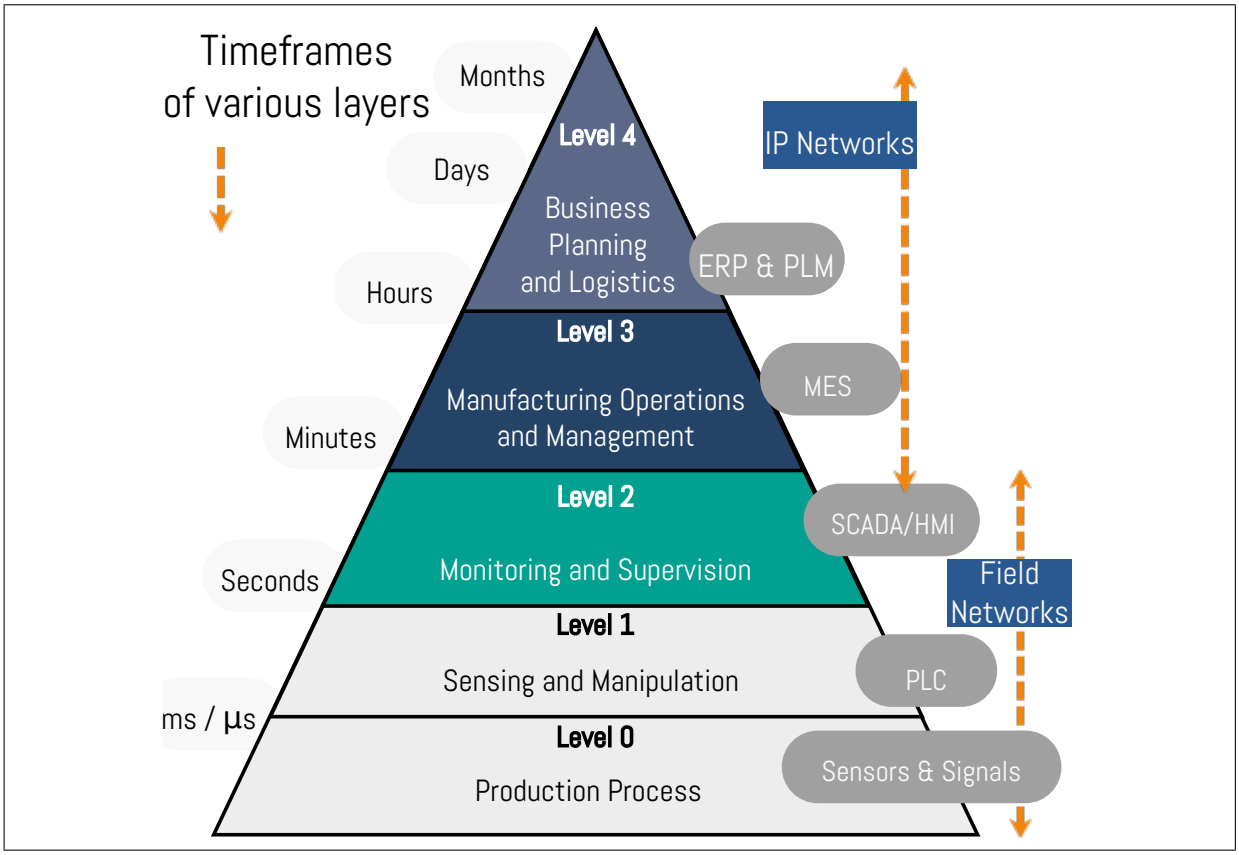

Figure 1.1: The conventional automation pyramid according to the ANSI/ISA-95 model. The five levels, 0-5, are defined in the middle. At each level, the typical system(s) used are showed to the right. Different levels are concerned with different timeframes which are visualised to the left [Aker18, Zueh10].

control layer (level 0 and 1) at the shop-floor operate on different time frames (refer Figure 1.1). Hence, the concept of 'soft-real-time' in ERP greatly differs with regards to the control layer. In the context of digital manufacturing, MES bridges this knowledge gap between the ERP and shop-floor by reacting to the events of the manufacturing shop-floor in real time. The IEC 62264-3:2016 standard [IEC16] divides the entire MES activities into four functional areas namely production, maintenance, quality and inventory management. Typical functionalities of production management in MES include sequencing the operations, monitoring the production and determination of the states of different entities involved in production with respect to real time. MES is also the focal point of this thesis.

Supervisory Control and Data Acquisition (SCADA) systems constitute layer 2 of the automation pyramid. They are software packages interfaced to the Programmable Logic Controllers (PLC), Distributed Control Systems (DCS), Intelligent Electronic Devices (IED) or other commercial hardware modules that form layer 1 of the ISA 95 model [DANE99]. SCADA systems consist of a number of remote terminal units (RTUs) that are connected to PLC, DCS and IED. These RTUs collect field data and transfer it to the master station, via a communication system. The master station displays the acquired data and allows the

human operator to perform remote control tasks. The accurate and timely data allows for optimization of the plant operation and process [Bail03].

The PLCs in level 1 of the automation pyramid are connected via field bus to level 0. Manufacturing resources on the factory shop-floor which consist of sensors, actuators, motors, switches, valves and contacts constitute level 0. Manufacturing resources are responsible for performing production tasks.

MES implements production and maintenance planning, production quality management and dynamic rescheduling through structural and functional interfaces to the SCADA systems [Colo12]. However, in recent times, there has been an increasing overlap between the roles of MES and SCADA systems in terms of production execution and control. In the same vein, functional areas of ERP and MES also run into each other [Modr09]. Therefore, the automation pyramid is not strictly a closed structure as shown in Figure 1.1.

With the advent of low-cost and smart sensors and subsequently Cyber Physical Systems (CPS), the sensors that are connected to the machines are now reachable as they can be addressed over the network via TCP/UDP over IP. Barring exceptional behaviors, ISA-95 also emphasizes that MES operates on the same time scales as SCADA. Thus, the manufacturing execution systems can directly co-ordinate with the machines in manufacturing plant without any time-compatibility issues. This development has given rise to the possibility of omitting the SCADA layer and delegating its responsibilities to the manufacturing execution. In many cases, SCADA systems and the connectivity solutions from the MES layer through the SCADA down to the shop floor have been characteristically vendor-specific. They do not follow industry standards and thus make it difficult to replace machines on the shop floor level. The trend of moving towards standardized communication protocols on all layers of the automation pyramid also fosters this development of circumvention of the SCADA layer. Figure 1.2 illustrates the practical positioning of the MES and SCADA in automation pyramid [Katt18b].

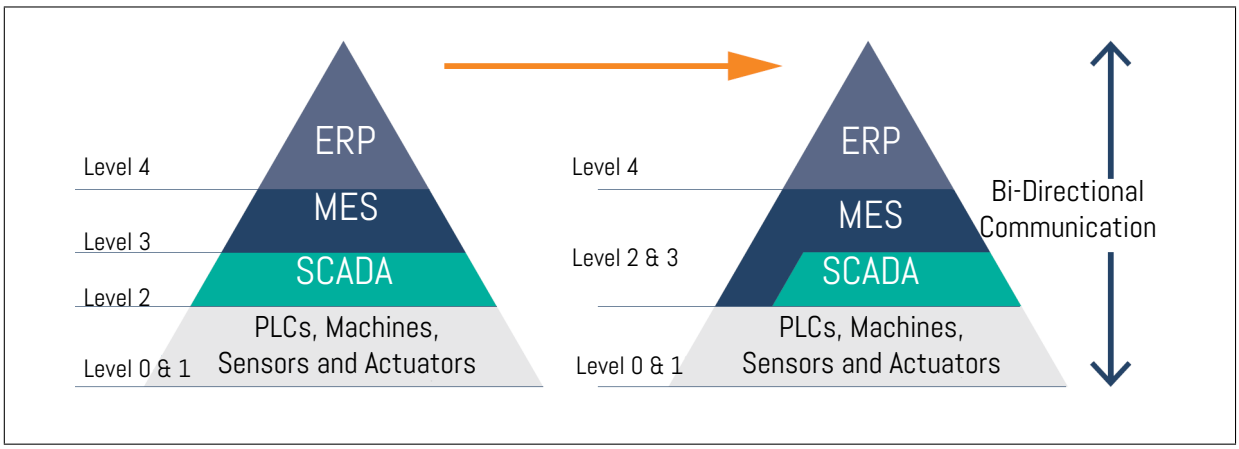

Figure 1.2: Evolution of classical Automation Pyramid [Katt18b].

The focus of this research work is the production management aspect of MES. Therefore, the research scope naturally broadens to accommodate the factors on the factory shop-floor that influence the production behavior at execution time.

The rest of this chapter is organized as follows: The definition and motivation of research problems that have been addressed in this research work is presented in section 1.1. Subsection 1.2 lists the concepts that are outside the scope of this thesis. Section 1.3 describes solution to overcome the research challenges.

1.1 Problem Statement

MES are operated in various forms and sizes across the manufacturing and process industries, such as homegrown MES, proprietary production control system, on-premise MES from third party vendor and cloud operated MES [Katt18b]. The cloud based MES is the subject matter of this thesis.

Motivation

During production execution, work stations comprising of plant modules, individual machines, even field devices seek information directly from cloud based MES. These work stations on the shop-floor request cloud MES for routing details at every stage of the production. Each work station collects the operation, Bill of Materials (BoM), machine parameters and other resource configuration details. Once this information is collected the machine is instructed on how to proceed with that step of the production process. Once that step of the production is completed, the work station informs the cloud MES the same along with the generated results. The cloud MES then processes the results and accordingly sets the next operation of the production. This process continues until all the planned operations to manufacture the planned component have been executed. During exceptional cases or conflicting goals, if the need arises, the routing path is changed, as instructed by cloud MES, to accommodate the exceptional situations. For example, the work in progress is diverted to a rework station if concerns regarding the quality of the products are raised. As work stations communicate directly with the cloud MES, it is possible to react immediately to changes in the production process such as priority customer orders and quality defects. It is also possible to take an alternative course of action in case of unavailable resources.

Though cloud MES has its advantages in terms of economy and scalability over classical on-premise MES [Katt18b], there are certain challenges in the cloud MES, or cloud computing technology in general. The cloud downtime and network latency are critical concerns for the manufacturer.

The communication between traditional MES and shop-floor takes place over WAN, which means that the transmission delay is not bounded. The network latency is a function of internet traffic that undergoes random fluctuation for the same bandwidth and infrastructure. Even though, today's internet links have been *over-provisioned* [Fral03] with high-speed links that experience little network congestion, when hundreds to thousands of shop-floor entities communicate at the same time, the degradation of network performance is inevitable. This latency becomes even more challenging in high speed manufacturing scenarios where the right information is required at the right time. Based on the experience, [Zueh10] also claims that WLAN links at industrial trade fairs often break down due to the network traffic generated within a relatively small enclosed area by a large number of smart devices which are equipped with wireless communication interfaces.

In the state of the art industries, the work stations constantly communicate with cloud MES to seek process parameters, recipe, machine configuration values and push the results describing status of products and resources during production control. This digitization of manufacturing produces large amounts of data that varies with the degree of automation on the shop-floor. This large dataset in turn has an adverse effect on data processing times on Create-Read-Update-Delete (CRUD) operations between factory shop-floor and cloud MES. This factor significantly contributes to the delay in response from cloud MES.

Typically, enterprise applications such as MES employ Simple Object Access Protocol (SOAP) [Box00] based web services for business-to-business (B2B) integration due to its W3 specifications and extensions that provide robust security, ACID [Haer83] compliance, successful/retry logic and provision of contracts. However, the complex envelope of SOAP consumes high network bandwidth, and extraction of SOAP envelope and parsing the contained XML information is an overhead compared to contemporary HTTP based light weight alternatives such as REST [Fiel00] based services. The virtualization principle of cloud computing [Zhan09] that can be applied at different levels such as computer hardware, operating system, storage and network also introduces its own series of packet delays and causes further performance degradation [Wang10]. Moreover, the problem of network latency which is encountered each time the request is created to fetch the next operation details from cloud MES does not auger well in high speed manufacturing scenarios.

In addition, although cloud providers claim near 100% availability, there are instances in the life cycle of cloud solutions where the services are disrupted due to many reasons such as electric failure, hardware failure, cascading failure on routers and cloud downtime arising out of data center migration, server update against vulnerability et cetera. These incidences, on an average, reduce the availability to 99.91%, which in other words a non-availability of 7.884 hours per year [Gagn14]. [Gagn14] also cites that Microsoft Windows Azure, which is one of key players in cloud business, had a total downtime of 272.04 hours in the year 2013.

Such network outages are not acceptable in the event of manufacturing a priority order.

Another argument against the cloud MES is its centralized governance. Decentralized systems are known to reduce the communication effort, increase flexibility, autonomy, adaptability and efficiency of the industrial automation systems [Marq17, Carv18, Kend17, Wagn10]. [Mour13] argues that owing to their benefits in terms of manufacturing costs, delivery times and agility, decentralized manufacturing approaches are preferred over centralized practices. Following arguments in this section are based with the focus on the decentralization in manufacturing.

OPC UA Information Model

The discussion in favor of decentralized manufacturing warrants a shift of focus on OPC Unified Architecture (OPC UA) technology. Currently, the Reference Architecture Model for Industrie 4.0 (RAMI 4.0) [Plat16] recommends only the IEC standard 62541 OPC UA for implementing the *communication layer* (refer Section 2.3). OPC UA is a platform and vendor independent communication technology for a secure and reliable data exchange over the different levels of the industrial automation pyramid. In addition, the information models of the OPC UA standard provide the foundation for semantic interoperability [OPC 19].

When the capabilities of a manufacturing resource are modeled in OPC UA servers, different resource vendors align to their own process and data model. It is typical of a manufacturing site to operate a variety of resources from diverse vendors that are based on different technologies and present different application interfaces and data formats. When the production line is set up with this arrangement for a specific product variant, the configuration is tightly coupled to the manufacturing resources. The changes to the production line for a different product variant require manual changes to the software configuration since this exercise involves consuming different manufacturing services of the resources and in some cases, it involves the introduction of new resources altogether. Even manual involvement of software reconfiguration is not smooth due to the heterogeneous production landscape constituted by these independently developed manufacturing resources. Additionally, the transmission of the right information to the right entity at the right time to solve complex manufacturing tasks is not possible when human interpretation is required at each step of the transaction. Moreover, manual configuration may also lead to performance degradation, anomalies and outages in practice. Even in the absence of error-free decision making, human involvement significantly slows down the data transfer and processing [Khil11]. Another associated problem is semantic under-specification: a term used by one information model may map to a different concept in another model, or different terms of different information models may refer to the same concept. These problems require interpretation of data by a human before

its utilization, and thus hamper interoperability among Cloud MES and on-premise software stack involved in manufacturing execution and control.

OPC UA Companion Specifications In order to make an optimal use of resources during production design and control, a standardized process and resource related information should be available. With an intention of encapsulating the services of manufacturing resources in a transparent and vendor-independent manner, OPC UA companion specifications (also called OPC UA Collaborations) of manufacturing resources across various types of industries are being released. The companion specifications enable building identical OPC UA servers for all the manufacturing resources that provide the same functionality, irrespective of the vendor. In other words, this allows identical modeling of a *type* of manufacturing resource from different vendors. Such a similar resource modeling warrants the same treatment from the perspective of an OPC UA client, and hence, significantly streamlines the task of obtaining the vital process data. The companion specifications that provide these standardized mappings between the manufacturing resources and the corresponding OPC UA information model are the ideal solution to achieve semantic interoperability.

In addition, the companion specifications also extend the semantic vocabulary provided by the standard OPC UA specification. The companion specifications enable resource to resource communication (horizontal integration), and production design and control solutions to resource communication (vertical integration) without referring to the technical service interfaces. The resources self-describe their information model which in turn substantially reduce the production line configuration effort.

Current Usage of Companion Specifications The self-description of resources substantially reduces the complexity involved in central production design and configuration. The production designer assigns the Production Order (PO) requirements to these standard methods of OPC UA server suggested by the common information model. In a cloud based manufacturing scenario, the companion specification of a particular manufacturing resource is hooked to the cloud MES in the form of an XML file containing the nodeset of the OPC UA server. The cloud MES then reads through the OPC UA node file to create an abstract resource at design time. Each manufacturing resource consists of numerous field devices which in turn greatly complicates the tag and method structure of the OPC UA server. Therefore, only requisite and partial nodeset structure is instantiated to create a resource digital twin in the cloud during production design phase, and its functionality is mapped against the service required to fulfill the business needs. Such an upfront mapping configuration consisting of a static orchestration plan is created in the cloud and dispatched to the edge layer of the manufacturing shop-floor to take over the control of the production. From a centralized manufacturing point-of-view, such a production design increases simplicity, and also instills confidence in production managers on account of its transparency and absence

of emergent behaviors.

Inadequacies of OPC UA Companion Specifications

- The companion specifications that are prescribed on paper aim to provide only an identical OPC UA information model of resources that deliver the same manufacturing services. For example, the companion specification allows an OPC UA client to interact in an identical manner with the devices of two different manufacturers that expose the same semantic information model and API. Despite this fact, this informal information model does not provide sound reasoning power to the machines. In other words, it allows a human to easily understand a machine model, but it does not assist in machine-machine automation under all circumstances.
- Besides providing the OPC UA servers of similar resources with the same information model, these various manufacturing resource vendors also provide additional functionalities in order to differentiate themselves from the market. The consumption of the additional services requires referring to vendor specifications, and hence an involvement of a human user cannot be avoided.
- Modern manufacturing plants contain countless numbers of manufacturing resources at an abstract level, and field devices at a detailed level for the purpose of regulation, control, analysis and visualization of production. Considering this complex arrangement of the manufacturing site, it is also impractical to assume that every entity shall possess an OPC UA companion specification.
- Another line of argument is the provision of rich information modeling in the OPC UA that offers the description of the data and the contextual information. Though

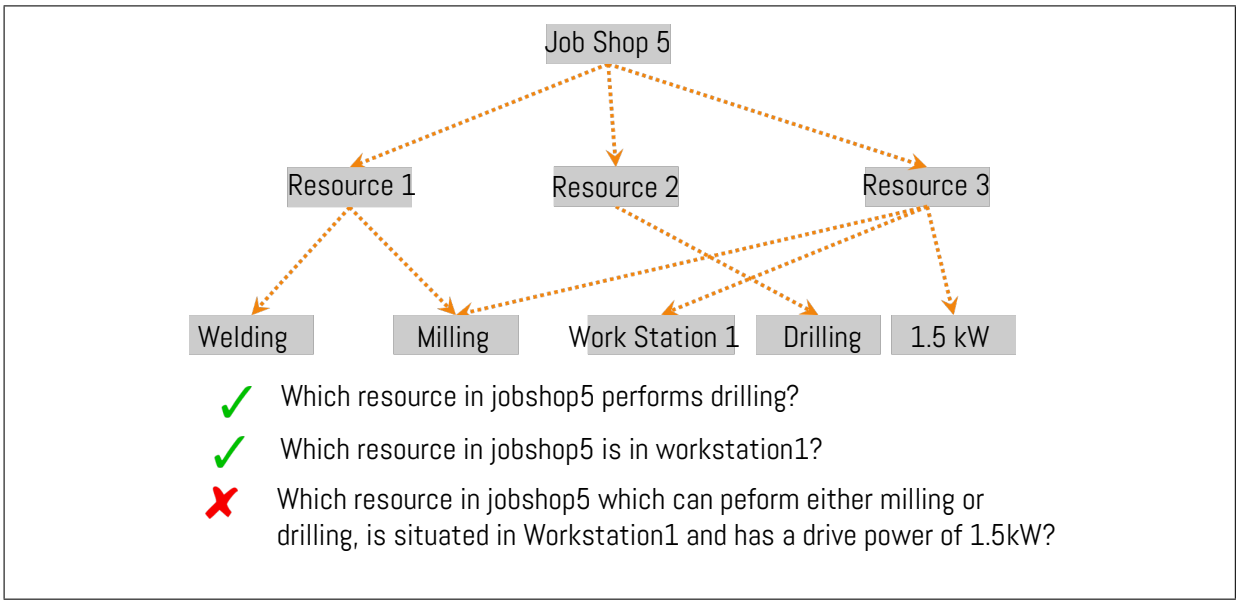

Figure 1.3: Illustration of Logical Reasoning Incompetency of OPC UA

OPC UA working groups have covered the bases with regards to most aspects of semantic information modeling, a reasoning engine that recognizes conflicting semantic definitions and infers logical consequences using the principles of first order predicate logic or description logics is still missing in OPC UA. Refer Section 2.3.3 for detailed description.

To apply strict and consistent rules in information modeling, the entities should be subjected to formal definitions. Currently, the OPC UA can design from a simple to the most complex information model depending on the application requirements. It is highly probable that errors are introduced inadvertently during the design of an information model of such complex systems. Therefore, it is even more important to enforce consistency checking during such modeling process. In contrast, formal ontologies do not just list real-world concepts into an ontological model, they also have the power of reasoning. Figure 1.3 provides an example of above explanation where the OPC UA with its current capabilities can only answer questions with green tick bullets, but not crossed in red. Such *intelligence* is required to reason on the services provided by manufacturing resources, assess the resource conditions for better coordination in the production and evaluate the pre- and post-conditions of an OPC UA method execution in the PO orchestration. The above-mentioned example is for the purpose of brevity, but this type of reasoning can also scale to represent the conditions of complex manufacturing systems. Mere provision of manufacturing service without adhering to its business and production pre- and post-conditions is not considered useful. Such conditions that not only involve connecting the mere references, but also involve logical and reasoning expressions which are represented in the information model at design time, and need to be processed at runtime to assess contextual information at factory shop floor, and it is not possible with the current OPC UA specification. To this end, the vocabulary provided by the OPC UA address space should transform to formal ontologies.

In a nutshell, the OPC UA information model and accompanying companion specifications do not assist in autonomous decision making of the entities in the shop-floor in the context of decentralized manufacturing. In the state of art, the companion specification only allows a human user to couple the required manufacturing services to the fixed physical resources on the shop-floor during the production design step on the assumption that a specific resource type contains the required application method. However, this static production design does not sustain disruptions in case of exception scenarios in production.

The argument of network latency in the centralized cloud MES, and subsequent discussion on emergence of decentralized control in manufacturing leads to the following main research question:

Research Question 1: *In a production setting controlled by cloud MES, is it feasible to delegate the manufacturing execution and control to the production network layer in order to enable decentralization of production? In other words, how to empower the factory shop-floor to become self-reliant in terms of production related decision-making and execution based on its local conditions?*

The above discussion leads to the following supplementary research question:

Research Question 2: *In the light of decentralized control, it is necessary for the decision making entity on the production shop-floor to have the capability of reasoning and drawing inferences on the suitability of manufacturing services of the OPC UA servers for the given production order (PO) requirements. How to construct such automated reasoning systems?*

The current trend is the ever-increasing tendency of shifting the manufacturing facilities towards decentralized control systems [Mour13]. However, designing such a completely decentralized system is a complex task [Tsit84] and also reduces the transparency of manufacturing processes to the business applications such as MES and ERP. The thesis strives to strike the right balance between these two extreme paradigms of production control.

The design and evaluation of an effective communication strategy between production facilities, equipment, individual sensors, actuators on one side and a centralized cloud MES on the other to answer the above-listed research questions is the aim of this dissertation.

1.2 Establishing the boundaries of research

There are several aspects that are out of scope of this thesis:

- The loss of governance is perceived as another big impediment to the acceptance of cloud based manufacturing solutions [Mang11]. When business applications are moved to the cloud, it forces the organizations to accept the control of the service provider on several important issues and areas of business and manufacturing data. As a result, the cloud solution provider will have overarching influence on the business processes. The fact that the valuable enterprise data resides outside the company firewall raises serious security and privacy concerns. However, security concerns of the cloud based MES are outside the purview of this research work.
- It is not possible to accurately determine the latency between two fixed points since the data packets encapsulated at the network layer of OSI reference model [Zimm80] need to pass through several proprietary routers of the internet before reaching the destination. Each of these routers has unpredictable traffic which is dependent on a variety of factors. Moreover, the network latency which is a function of internet traffic

also undergoes random fluctuation for a given bandwidth and infrastructure. Therefore, instead of imposing hard real-time constraints, the practical unit of measurement should be average time for the network latency. Additionally, the focus of the thesis is the high level process automation, which usually do not pose real time requirements.

- Modern MES are capable of replicating most of the functionalities of ERP with regards to manufacturing activities. Creation of Production Order/ Shop Order, which is one of the activities pertinent to production management aspect, can be carried out independently in MES. Therefore, the ERP is not part of this study.
- There are several standards to implement the communication layer in the industrial automation setting such as OPC [OPC 02], OPC Unified Architecture (OPC UA) [OPC 17a], MTConnect [Vija08], Woopsa [Web 16], Data Distribution Service [Kang12]. However, the Reference Architecture Model for Industrie 4.0 (RAMI 4.0) recommends only the IEC standard 62541 OPC UA. Moreover, as per Platform Industrie 4.0, even to comply with the "Industrie 4.0-enabled" at lowest category, an industrial resource/product should integrate the OPC UA information model [Hopp17]. Therefore, this work concentrates on the OPC UA standard. From a communication protocol perspective, the signed and encrypted UA native binary protocol is opted since its performance is better in comparison to the WS-* based SOAP/HTTP(S) OPC UA implementation. Other protocols such as MQTT and AMQP are not discussed in this study.
- This research targets only the discrete event automation systems. Although the thesis solution can be extrapolated to continuous-time systems and controllers, it has not been subjected to evaluation.

1.3 Solution Methodology and Organization of Thesis Manuscript

Taking the following actions is one of the first steps of the multipronged strategy to overcome the problems described in Section 1.1:

1. Reduction of geographical distance between the cyber physical systems (CPS) on the factory shop floor and MES
2. Shifting of validations on intermediate results that are necessary to ensure the production does not deviate from the plan from centralized cloud MES to factory network layer

The above-mentioned first action negates the network delay. As the data grows in MES,

the average duration of time needed to upload, download, process and analyze the data also grows in correlation. The second action overcomes the large dataset processing delay that stems in MES and thereby quickly arrive at feedback relevant instructions within a short time delay. The implication of these above actions is that the control data which is required for production execution should be close to the shop floor. In other words, this data should be cached at the proximity to the shop floor. This initiative of caching the production data is the first measure in the direction of promoting autonomy for production systems.

Despite the collaborative nature of decentralized production processes, a facilitator should be identified that owns the process for successful completion of production. Decentralized production involves interaction among participants of the shop-floor, and the facilitator helps in reaching a consensus [Marq17]. Therefore, an edge component called Generic Shop-Floor Connector (GeSCo) is conceptualized to play the role of the facilitator that orchestrates the production execution. The designed architecture also includes identifying a set of modules in GeSCo with dedicated responsibilities. The production data is cached in GeSCo (refer Chapter 3 for more details).

Mere caching of production control data at the proximity of the shop-floor solves only a part of the problem. The static production data does not take into account the context of the shop-floor, and hence, is incapable of exploiting the dynamic conditions of the shop-floor. Furthermore, in the event of exceptions such as quality non-conformance and resource breakdown, the continuation of production execution requires guidance from cloud based MES. Therefore, the next step of the process is the flexible production orchestration by GeSCo which is possible only via automatic search, discovery and invocation of the manufacturing services offered by the OPC UA servers of the manufacturing resources. However, the information model of OPC UA alone is not sufficient to formally describe the semantics of the application-specific methods of an OPC UA server. The OPC UA collaborations [OPC 18a] seek to standardize the OPC UA servers into a common data model and common communication interface for a particular industry type. However, this exercise is still in its nascent stage, and also it is impractical to assume that every entity represented as OPC UA server in the manufacturing site shall adhere to an OPC UA collaboration. Additionally, the orchestration engine that needs to take care of pre- and post-conditions of a method execution cannot be part of an OPC UA collaboration. Such knowledge which encapsulates the circumstantial states of various elements of manufacturing facility only originates on per business case basis. Therefore, the application-specific methods of OPC UA that enable service oriented architecture in production automation must be described in a formal and vendor-neutral manner in order to describe all the aspects of production orchestration in a machine-interpretable format from a server perspective, and assist in discerning the meaning, reasoning and drawing inferences from this data automatically from a client perspective.

Description Logics (DL) provide the required degree of formalism and also, at the same time is decidable in finite time [Baad08]. Moreover, there are many widely known ontology frameworks such as OWL-S [Mart04], SAWSDL [Kope07] and WSMO [Roma05] that fortify web services with formal semantics. Instead of reinventing the wheel, this thesis uses case-based reasoning for self-description of application-specific methods of OPC UA resources. It adapts the existing semantic web service (SWS) standards for this purpose. The author proposes a modified OWL-S framework (top-down approach) [Katt18d] and SAWSDL (bottom-up approach) [Katt18a] standard to the OPC UA application methods. Further, a hybrid approach [Katt18c] which is a blend of the above-mentioned modified OWL-S and SAWSDL specifications is introduced. The integrated approach overcomes the downsides and yet retains the benefits of its constituents. All the three approaches are described in Chapter 4.

The above-presented semantic frameworks operate on the hypothesis that a production ontology is created by a human expert. This acts as a foundation for developing ontologies for OPC UA application methods. But production ontology encoding is a complicated and time-consuming overhead task that befalls on the manufacturer. Therefore, the thesis presents a pragmatic automation of an encoding of a primary and light weight production ontology based on the source code of MES (refer Chapter 5 for more details). To this end, the knowledge of OPC UA collaborations is also exploited during the creation of resource ontologies. This approach also eliminates the creation of ad-hoc ontologies at different levels of the automation pyramid and subsequent ontology alignment practices. The reverse-engineering transformation from source code implementation (imperative paradigm) to ontology representation (declarative paradigm) results in unavoidable loss of information. This information loss is compensated by the formulation of a ruleset using OWL based constructs and OWL based rule languages such as Semantic Web Rule Language (SWRL), Semantic Query-Enhanced Web Rule Language (SQWRL) and SPARQL Protocol and RDF Query Language (SPARQL) based on the feasibility and requirements of specific rules. The generated ontology and an abstract PO hooked with formulated rules are cached to the shop-floor network for consequent production control to enable smart edge production. An evaluation of the conceptualized solution is run on an intelligent key finder demonstrator (refer Chapter 6). GeSCo discovers appropriate manufacturing services for the given production order (PO) by looking up into its method discovery repository and subsequently, creates the orchestration plan. During production execution and control, pre- and post-conditions are executed/validated and accordingly the next production step is pursued. The average latency over the course of PO execution is reduced drastically due to localized decision support provided by the combination of semantic frameworks of OPC UA application methods and DL based rule validations. Chapter 7 discusses the scientific and industrial contributions of the thesis on a holistic level, and concludes with the recommendations for future work. Figure 1.4 shows

the structure of this thesis and also inter-dependencies between chapters.

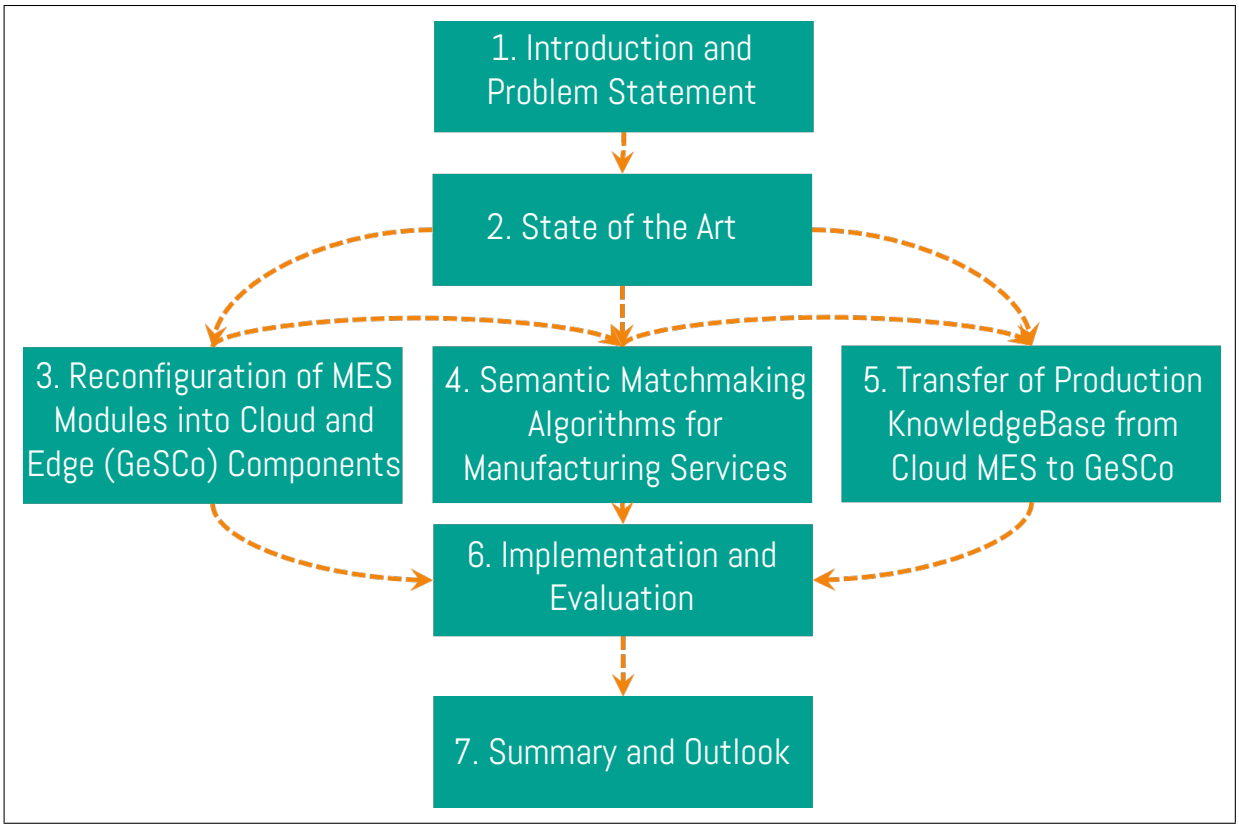

Figure 1.4: Thesis Structure and Interdependencies between Chapters

2 State of the Art

This chapter is divided into four sections. Section 2.1 motivates the usage of MES in the manufacturing, explains various MES deployment options, and lists related work in the area of cloud control in manufacturing. Section 2.2 describes how decentralized production strategies in manufacturing fare better than centralized control. Later, some important decentralized architectures and implementations are discussed. Section 2.3 briefly describes Reference Architecture Model for Industrie 4.0 and Asset Administration Shell (AAS). Focus is shifted to the OPC UA architecture which is a RAMI4.0 recommendation to implement communication layer of AAS, and later, the insufficiency of formal reasoning capabilities on its information model to implement decentralized production control is described. Last Section 2.4 emphasizes on semantic web and semantic web services, some well-known semantic web services frameworks, and ends with the description of previous work in this direction.

2.1 Manufacturing Execution Systems

In the context of manufacturing, ERP and PLM are most commonly used business software solutions. This section argues that these solutions are not capable of detailed shop-floor supervision, and MES, which forms layer 3 of automation pyramid fills this gap. Its objectives, various deployment options, and finally, the previous work in this area with a focus on the research topic are discussed in this section.

2.1.1 Enterprise Business Solutions in Manufacturing: Motivation for MES

Business management and logistics softwares such as ERP and PLM are promoted as integrated software applications based on central information systems. PLM is a technology solution which provides a shared platform for collaboration among product stakeholders, and streamlines the flow of information along all the stages of the product life cycle [Amer05]. Product information tied to a product design process increasingly resides within PLM systems, and on the other hand transaction-based manufacturing processes typically reside in ERP systems [Batc12]. Table 2.1 briefly summerizes differences between PLM and ERP.

Table 2.1: Distinction between PLM and ERP. Adapted from [Wu14]

Attribute	**PLM**	**ERP**
Domain	Engineering Design	Manufacturing
Goal	Product Management	Resource Planning
Driver	Projects	Orders
BoM Structure	Engineering-BoM	Manufacturing-BoM
Delivery	New Products	Ongoing Manufacturing

The ERP addresses all the core functions of an enterprise that include sales, marketing, manufacturing, distribution, personnel, supply chain management, and finance modules [Holl99]. From the perspective of manufacturing, the ERP empowers a production designer with an integrated real-time view of core business processes such as production, sales/shop-order processing and inventory management. This enhances the visibility of business processes and accuracy of information to the manufacturer [Seet15]. The resulting benefits from a broader perspective are informed decision-making which lower the production costs, alleviate product quality non-conformance, reduce wastages in terms of resources and time, improve production efficiency and achieve real-time deliverables. In the context of enterprise business solutions, many literature make loose references to real-time without the consideration of predictability and worst case execution time. Within this thesis, 'real-time'ness of enterprise business solutions is the ability to update information to all the stakeholders within and across the organization instantaneously without bounded time complexity.

The ERP transactions are executed at the point-of-use that operate under the perception that data changes with regards to the transaction are percolated to all the software modules participating in the manufacturing. However, the batch processes must be run to downstream the transaction data in order to make the changes visible to other modules. Similarly, the changes effected by other modules also need to be uploaded to the ERP. But modules in different layers of ISA 95 model operate on different temporal horizons ranging from days to weeks at the strategic level to seconds at the operational level (refer Figure 1.1). Therefore, there is a time lag between the actual occurance of shop-floor control data and its recognition by the management level in the ERP systems [Helo14]. [Huan02, Helo14] also claim that decisions of high-level plans of ERP hardly reach the lower-levels of operational systems. Even though ERP supports information flow across all the stakeholders, there are also concerns regarding the effectiveness of ERP capabilities in shop-floor supervision as it only stresses on managerial issues [Jeon17]. Hence, MES was introduced for managing shop-floor activities that integrates the high-level production schedule in the shop-floor according to the conditions of lower-level operational systems.

2.1.2 Objectives of MES

MES is a comprehensive automation software solution that coordinates all the responsibilities of modern production systems. The MES historically has been a vendor and industry specific solution and hence, is also called by other names such as Collaborative Production Management (CPM) and Manufacturing Operations Management (MOM) [IEC16]. Recent MES solutions comprise of former Computer Integrated Manufacturing (CIM) components such as Computer Aided Planning (CAP), Computer Aided Manufacturing (CAM), Computer Aided Quality Assurance (CAQ), Production Data Acquisition (PDA), Machine Data Acquisition (MDA) and Personnel Time Recording (PTR) [Arbe11]. Figure 2.1 illustrates the functionality overlap of MES with these former and current CIM components.

The CIM components were not thoroughly integrated into operations and processes as they operated as department specific standalone applications. Instead of supporting individual departments within an enterprise, MES considers the production as a holistic process within an organization.

The horizontal and vertical integration in manufacturing refines the operation flow, which in turn increases the productivity. Such productivity gains are significant to the manufacturing companies to compete in the future and stay relevant in their fields [Wang00]. The MES, which fundamentally is a centralized control structure, helps achieve this goal of the

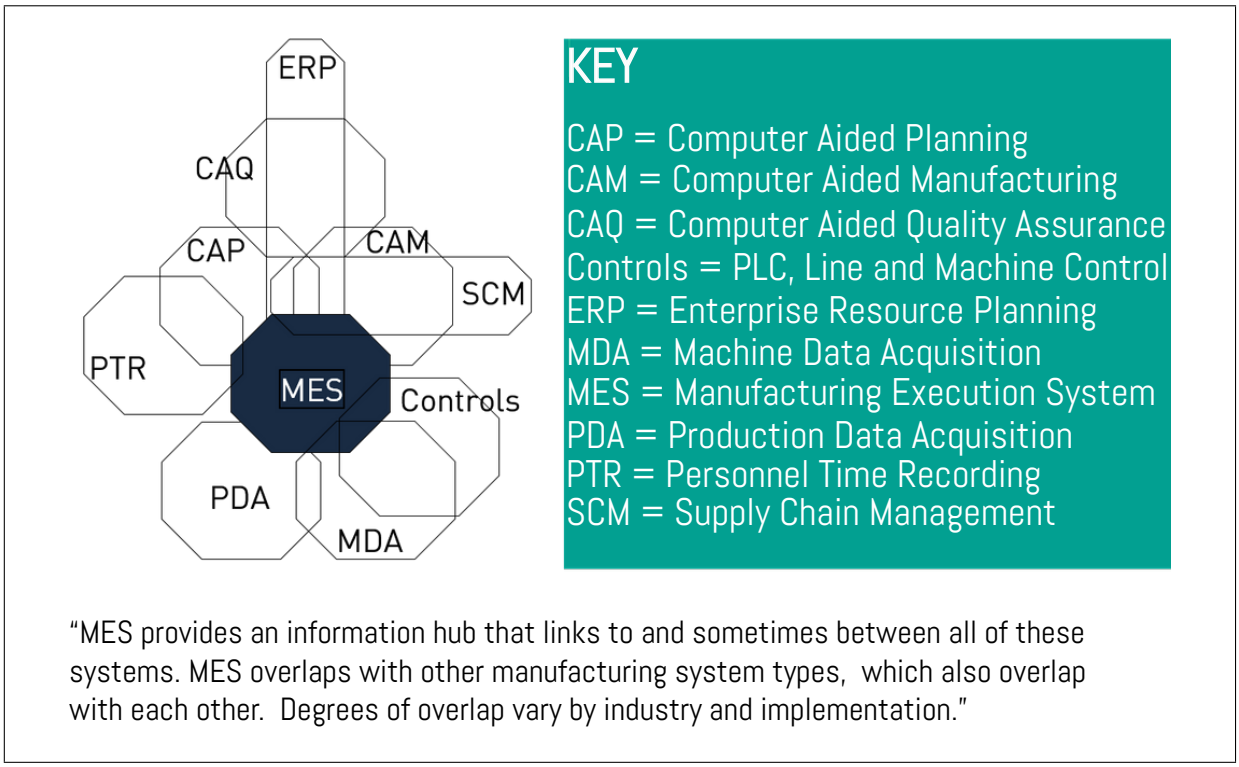

Figure 2.1: MES Information Context Model in Manufacturing Enterprise. Adapted from [MESA97]

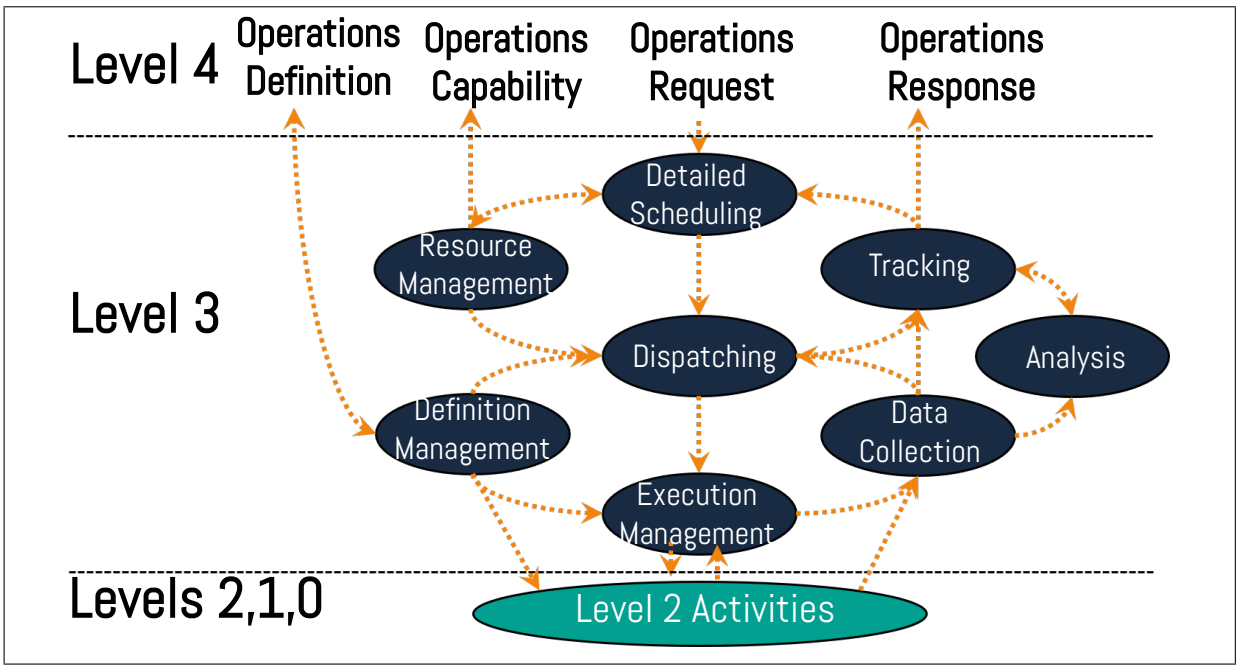

Figure 2.2: Generic activity model of MES [IEC16]

manufacturing companies by connecting the manufacturing resources and field devices to its centralized production control data. The MES also helps in better understanding the internal and external value chains in the manufacturing companies [Auto17]. The purposes of installation of MES are as follow [Arbe11]:

- Improve productivity while increasing production flexibility
- Enhanced process and product quality
- General process and cost optimization
- Shorten the production life cycles of the product

The IEC 62264-3:2016 standard, which is an extension of ANSI/ISA-95 model, defines activity models of MES in the view of enabling integration of enterprise system and the control systems. It divides the entire manufacturing operations management activities into the following four functional areas:

- Production operations management
- Maintenance operations management
- Quality operations management
- Inventory operations management

These different areas of the manufacturing operations management follow the generic activity model pattern as illustrated in Figure 2.2. Various levels in Figure 2.2 correspond to those levels in automation pyramid conceptualized in ISA-95 model (refer Figure 1.1). The generic

activity model can be broadly classified into four types of activities:

- Operations Definition - Specifications of operations and the required resources
- Operations Capability - Resource attributes
- Operations Request - Scheduling, Dispatching and Execution of operations
- Operations Response - Feedback and Results of the operations

The MES solution implements these activities and coordinates the information exchange between them.

Verein Deutscher Ingenieure (VDI) defines VDI guidelines 5600 [VERE07], which are based on the ANSI/ISA-95 model directed at discrete industry, for the German speaking community. It postulates that the MES should incorporate the following chronological aspects of the production process: Prognostic, real-time and historical. The prognostic aspect deals with production process planning; the real-time aspect involves management of the production process and the historical aspect involves analysis and reporting of production process. [VERE07, KARA14]

The standards such as IEC-62264, ANSI/ISA-95 and VDI 5600 provide only the guidelines and there is no standardization in MES implementation. It varies for different vendors, manufacturers and industries.

2.1.3 MES Deployment Options

A manufacturing facility can employ one of the several MES deployment options that fits its requirements, resources and budget. This section explores these MES flavors in a nutshell in terms of their competitive advantages and downfalls from a manufacturer point of view. Though, there is no universal solution for every set of requirements, it provides a general recommendation on the choice of MES based on economic and/or administrative rationality.

On-Premise MES Solutions

Historically, the MES have been mostly an on-premise software solution, i.e., they are nursed close to production sites.

Home-Grown MES The development task of such in-house systems is carried out by internal IT employees. This exercise involves direct contact between IT employees and back office personnel who utilize the system, and hence the system can be tailored to requirements. Hence, the majority of manufacturers choose this custom option in order to have an exact fit to their manufacturing processes.

Analysis: The manufacturer possesses complete control over the functionality, and a thorough knowledge of the system. At the same time, these self-owned MES are also difficult to enhance, in terms of business functionalities arising due to the growing trend of an increasing number of product variants and shortened product life cycles, in an efficient and time bound manner. As the implemented MES fits exactly to the requirements and is subjected to continuous development on an evolutionary basis, the software modules are tightly coupled to the manufacturing infrastructure. Both these factors are hindrances to the adoption of new technological innovations that enable complete horizontal and vertical integration.[Beac00, Katt18b, Schn10]

MES from Third-Party Vendor In this case, the manufacturer purchases the MES from third-party vendors. The manufacturing industry is complex and diverse, and hence, it is natural to find various mix and match approaches by the manufacturers to best fit their production requirements and budget constraints. A common subclassification in this category is a tailor-made version from the vendor who builds the MES as per the functional model specified by the manufacturer. Another little-known practice is the purchase of core modules of MES from the vendor by the manufacturer. This exercise further involves training and service from the vendor so that the manufacturer can take over the source code for additional custom development. This subclassification of vendor-provided MES provides all the benefits of an in-house MES without the huge investment of time.

The selection of an MES generally results in a long term relationship with the MES vendor in the interest of protection of investment [Beac00]. To that end, a detailed analysis of investment is necessary taking into the account the life cycle and cost of maintenance during the feasibility evaluation of an MES vendor [Wei07, Hedm14].

Analysis: The vendor provides a proven and off-the-shelf solution that incorporates industry best practices, along with professional support and training to the work force of the manufacturing organization. The vendor guarantees long term maintenance and further development of MES modules, and integrates the future customer requirements in the product design and development. However, the continuation of status-quo after successful installation of MES is expensive since it involves upgradation of hardware components and IT solutions owing to their short innovation cycles. Additional difficulties such as platform dependency, license model, costs to maintain the MES software and work force that needs to be trained also needs to be considered. Under these circumstances, the purchased system can also be inflexible as it is targeted to the whole class of a particular industry. [Iiva90, Ltd12, Prab15]

Proprietary MES The proprietary production control system that is part of the automation hardware is the third classification of MES. These production control systems convert

the ERP orders to technical production orders for the assembly lines [Arbe11]. However, such MES systems are increasingly discontinued due to the recent trend of provision of standardized interfaces across all the layers of automation.

Analysis: This complementary MES can be operated off-of-shelf, and also relatively economical compared to its other on-premise counterparts. However, when the hardware and subsequently the production control software is discontinued, the future manufacturing maintenance is not safeguarded. In such situations, the implementation of expansions and changed requirements at the production execution is highly impractical and expensive. This compromises the flexibility and future security of the plant. [Arbe11]

Cloud Based MES

The cloud based MES is a blend of various IT technologies such as distributed computing, internet technology, hardware virtualization and open source software. To be more precise, Internet of things (IoT), which enables perception, internet connection, acquisition and automatic control of various manufacturing resources and capabilities, is the core enabling technology for the implementation of cloud based MES [Tao14]. Cloud based solutions, in general, are best described as web based solutions that run on remote servers and accessed via internet on standard web browsers [Lena11]. The cloud MES solutions are offered as IaaS (Infrastructure as a service), PaaS (Platform as a service) and SaaS (Software as a service) layers in the cloud architecture that are demand driven and charged as per usage [Hwan11]. The services in cloud based MES are generated by virtualizing and encapsulating the available manufacturing resources and capabilities [Tao15].

Analysis: Instead of developing individual projects from scratch, these MES Solutions are mostly assembled from configurable software components. The generic set of functionalities is built as per the customers' requirements and typically, the functionalities provided by cloud based MES are richer than on-premise counterparts [Mars11] and are also simple, fast and cheap [Voor11]. Another main benefit of the cloud based MES is that it requires nearly no IT resource investment [Lena11], and therefore, lowers the entry costs for smaller firms that try to benefit from compute-intensive business analytics that were previously available only to the large corporations. This also lowers the IT barriers to innovation in the manufacturing processes [Mars11]. The cloud based MES helps smoothly face the challenge of unexpected peak production demand without additional investment on on-premise resources [Wood09]. This is made possible with the virtualization principle of cloud computing technology. The virtualization argument holds true in case of redundancy or upgrade costs of the on-premise resources. The cloud servers are run as per the necessity, licenses can be increased or decreased accordingly. This decision need not be made upfront.

The dearth of skilled resources that are acquainted with MES technology, achieving the returns on investment and technology compatibility are no longer the problems in the cloud scenario. Consequently, this version of comprehensive MES setup quickly adapts to the newer innovative technologies and also offers significant cost benefits to the manufacturer at the same time. These arguments infer that cloud based MES is a good substitute for traditional on-premise MES solutions.
Nonetheless, there are mainly two important downsides in adopting the cloud solutions in general, namely security and privacy concerns, and network latency [Apos13]. Only the latter is the focus of this thesis. Cloud based MES provides on-demand and scalable storage system, and computing power that can scale to digital manufacturing requirements. However, for the purpose of condition-monitoring of resources, emergency-response and other latency-sensitive applications, the round trip delay caused in transferring the data to and from the Cloud based MES is unacceptable. Furthermore, it is not an efficient approach to send large data to the cloud for storage and processing as it also saturates network bandwidth [Dast16]. Storage of large amounts of data in cloud based MES also makes it vulnerable to have high computation cost (refer to the description of the problem statement of the thesis in Section 1.1).

2.1.4 Previous Works in Cloud Manufacturing

Pioneering and Initial Works

Networking, autonomous systems and resources, and smart decision-making processes employing the emerging technologies from the information technology (IT) domain are the characteristics of digital factories of the future [Wahl12]. The cloud computing paradigm which has its roots in the IT world has emerged as a dominant force in industrial systems to offer automation functions as services. Among many utilities of cloud infrastructure in the context of manufacturing such as data analysis, digital twin and condition monitoring, and training statistical models of high complexity and computing power, only the works concerned with production control are considered in this section. [Xu12] proposed the *cloud manufacturing* business model where distributed resources were encapsulated into cloud services and managed in a centralized way. Manufacturers used these cloud services according to their requirements in a *pay-as-you-go* model. This work also extends the idea of cloud services to apply to all the stages of the product life cycle such as product design, manufacturing, testing and management. At the same time, [Tao11] proposes cloud manufacturing, called CMfg, combining cloud computing and the service oriented manufacturing model. It discusses as key enabling technologies four types of cloud service platforms: public, private, community, and hybrid, and advantages and challenges of implementation. The main objective of *Cloud*

Agile Manufacturing , presented in [Maci12], is to offer the industrial production systems as a service, also called Machine-as-a-Service (MaaS) via virtual representation in the cloud. It provides higher level business applications in cloud a direct access to the automation services such as access to quality metrics, process control, and status of resources, raw materials and processed products.

[Schl13] introduced a new edge-cloud architecture, based on *global information architecture for industrial automation* by [Voge13], consisting of two layers modeled as cones. The upper and lower layers represented business and technical processes, respectively. These two layers exchange information in a standardized manner using an information model sandwiched between them to enable vertical and horizontal integration. There are also some works that distinguish the control level into two parts: control functions that need highest performance for critical control loops, and other control functions that can operate with reduced performance. Accordingly, the latter functions that control the lower levels are migrated to the cloud [Bett13]. The architecture proposed by [Schl13], which distinguishes IT and automation technology (AT) clouds, was subsequently adapted by further integrating these clouds [Give13].

Literature related to network latency in Cloud Manufacturing

[Lang12, Verl14, Lang14] are one of the pilot projects to shift the runtime control level to the cloud via standard interfaces and semantics. The key objective of these works is to apply the ideas and methods from the world of IT to the world of automation in a systematic and structured manner.

[Gold15] introduces Control-as-a-Service (CaaS) in industrial automation which employs the cloud based *Soft PLC* which achieves a round-trip time of around 1000 ms in 99.72% of times while serving 30 parallel tenants/programs. The concept of caching machine data on the internet gateway in case of low bandwidth, and transferring this cached data to the cloud on reduction of the traffic congestion is proposed in this work. Another soft-real time Industrial Automation as a Cloud Service to manage many computing and communication systems was proposed in [Hega15]. This work also argued that cloud controllers can act as backup for physical controllers during upgradation or replacement. [Gazi15] is another approach to reduce the network latency in a typical industrial scenario. It proposes fog computing where after careful examination of resource data at the gateway, only necessary information that is required for the IoT application is transferred to a cloud. In an industrial automation setting, [Kane16] concentrates on availability by introducing redundancy in public cloud servers. In [Khan17], the authors recommend to increase the computational power of the gateway in order to run the sophisticated statistical models used for examining the data. [Lenn17]

Table 2.2: Requirements for some typical automation domains. Adapted from [Lenn17]

Types	**Cycle Time**	**Communication Technology**
Building Automation	Seconds	BACnet
Process Automation	10 - 1000 ms	DCOM, TCP/IP
Factory Automation	500 μs - 100 ms	RT Ethernet (Profinet)
Substation Automation	250 μs - 50 ms	Profinet IRT, Modbus
High Voltage DC control	10 - 100 μs	Not applicable

classifies the various application scenarios, the corresponding communication technologies and the cycle times (refer Table 2.2).

The white paper by Cisco [Cisc15] and [Khan17] suggest for *data aggregation* as one of the possible solutions to counter the network latency. They further recommend to send only essential pieces of information to the cloud for further analysis. [Abde17] proposes a delay compensation technique to counter the latency by reducing the problem statement to the one presented in [Besc12]. On similar lines, [Sang17, Mube17] also present delay compensation techniques to mitigate the network delays.

This thesis considers countering the problem of cloud network latency between the automation pyramid levels 1 and 3 (refer Figures 1.1 and 1.2). The goal is to achieve a soft real-time and average response times in the range of 100 ms.

Even though MES is a centralized system, it allows entities participating in manufacturing access its data in production runtime, and hence, enables logical decentralization.

2.2 Production Control Strategies

After discussing the centralized MES in Section 2.1, this section compares the centralized and decentralized systems, discusses some important decentralized production architectures, and makes general observations and recommendations regarding the control strategies suitable in manufacturing.

The argument of centralized versus decentralized decision-making has for long been the topic of research in a myriad of contexts, namely economics, game-theory, political science, biology and ecology [Zabo02]. These concepts are applicable to decision-making processes in an organization. The characteristic focus of these literature has been on trade-offs between the two opposite sides of delegation of control in production [Zabo02]. The same concepts have also been extrapolated to the manufacturing domain.

Early manufacturing system theory suggested cautious and rather pessimistic organizing

principles for systems exposed to substantial internal and external uncertainties. These principles profess that it is better to recognize ignorance than to presume knowledge. Ensuing research in this area led to an evolution of the system structure with distributed responsibilities, tasks and resources [MONO06]. Later investigations in this area classified control mechanisms into four types, namely, "centralized", "hierarchical", "modified hierarchical", and "heterarchical" system structures [DILT91]. An adapted version of the [DILT91] classification was introduced by [SCHO07] who groups the system structures into two broad classes, namely hierarchical and heterarchical controls. This work further classifies control structures into three categorizations: centralized, decentralized and autonomous control, where decentralized control spans over both hierarchical and heterarchical structures. This classification is described in Figure 2.3, and this thesis also follows it. Inspired by [DILT91], [TREN09] also constructs an architecture typology where three distributed control structures, called "fully hierarchical", "semi-heterarchcal" and "fully heterarchical" controls, are derived from a "centralized" control system. The following sub-section describes the broad categorization of centralized and decentralized paradigms in brief, and subsequently presents the prevailing conditions of modern industry in this direction.

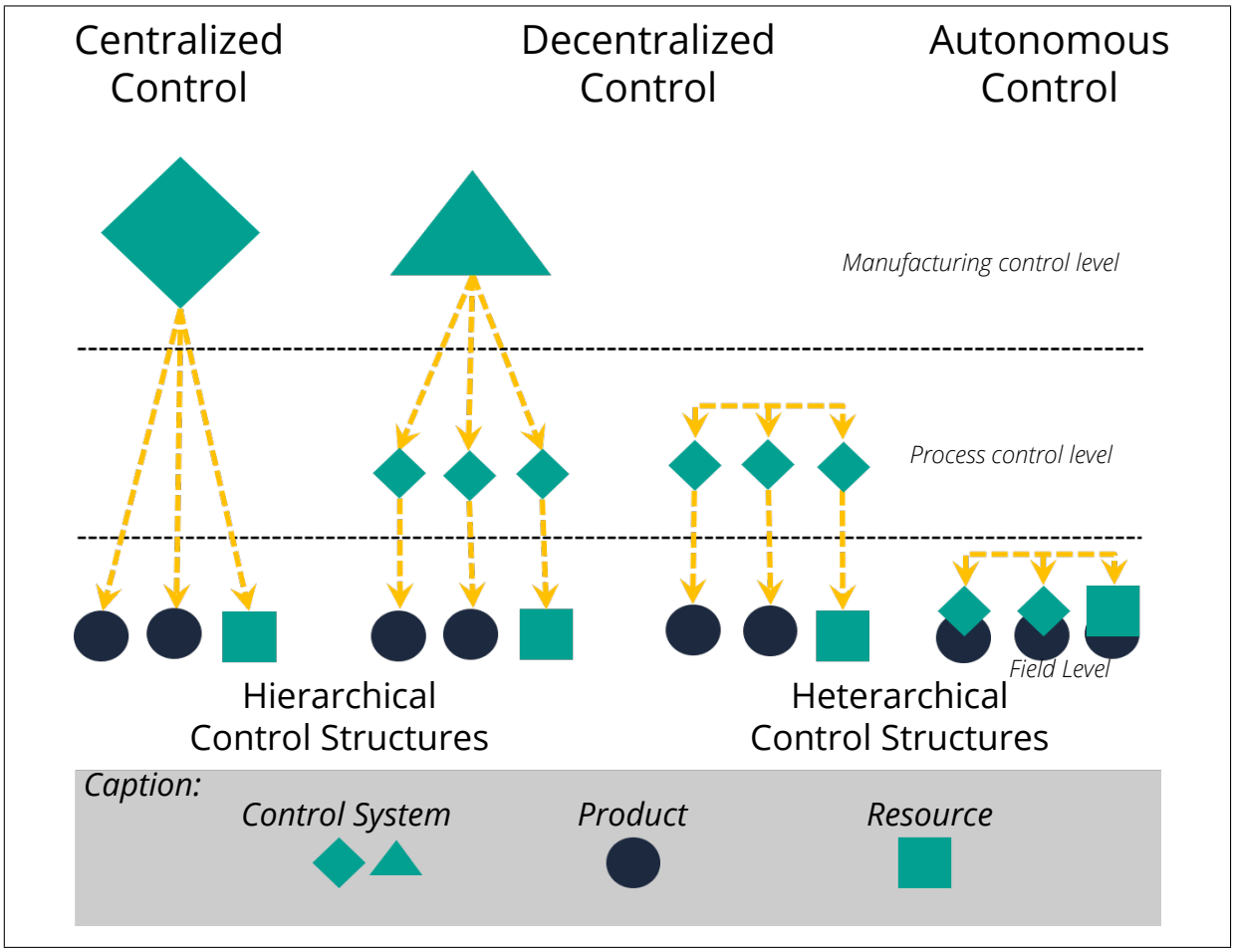

Figure 2.3: Classification of Production Control Systems [SCHO07, KOLB18].

2.2.1 Centralized and Decentralized Control Systems

Centralized control systems possess strict master-slave relationships between the components, where control decisions propagate in top-down fashion and the status is reported in bottom-up fashion [Vers06]. These control systems are characterized by complex solution algorithms, and an omnipresent processing unit, also called decision center in some literature [Marq16], and are tailored to solve a dedicated set of problems [Ande00]. This decision center is acquainted with all the system information. It is responsible for the entire system planning and hence, manages the operations at all stages of the manufacturing. This central entity also takes unbiased decisions that are aimed at optimization of the objectives of an entire organization [Fray01]. There is no direct communication between the local controllers at the bottom layer [Früh17], and they are helpless when cut off from their directing supervisors [Vers06]. The implementation of complex algorithms and analysis of large amounts of information set to obtain a globally optimal solution in centralized systems results in large computation time, and hence, the decision-making process slows down. Additionally, centralized systems have proven to be inflexible in cases of unexpected events and product customizations [Saha06, Leit09]. For example, from the planning and execution point of view, in the event of a manufacturing resource breakdown during production control, the corresponding information is fed back to the business systems such as ERP or MES. The production routing is adapted in the business systems, and the updated PO is pushed to the shop-floor to act against the prevailing contingency. The resulting response time is very long which renders the whole system inapplicable for high speed manufacturing scenarios. From the configuration perspective, if a product definition changes in PLM or ERP, it necessitates retooling of the entire system which is expensive in terms of both time and money. [Leit09] states that centralized control systems also do not efficiently support current manufacturing requirements, namely reconfigurability, robustness, agility and expansibility.

The manufacturing control is concerned with managing and controlling the physical activities on the shop-floor aimed at executing the manufacturing plans provided by the manufacturing planning activity, and to monitor the progress of the product as it is being processed, assembled, moved and inspected in the shop-floor. However, there is a big gap between the manufacturing planning and its actual control which is famously described as follows [Paru87]:

> "[...] In fact, it is proverbial among shop foremen that the schedules produced by the front office are out of date the moment they hit the floor. [...]"

Therefore, centralized control systems that are inflexible to adapt based on the conditions of the factory shop-floor are not ideal from a manufacturing control perspective. Instead **decentralized control systems** are preferred. The decision-making process is delegated

to the individual actors in the factory shop-floor in decentralized manufacturing. At each step of the manufacturing process, a decision-maker is identified out of all the participating entities in the production. Decentralized systems are based on distributed control in which individual machines react to local conditions in real time. These machine components are linked to neighboring components to form a network that display the desired self-organizing behavior. The behavior of each of the individual entities is defined with an express intent of optimizing its own objectives [Marq16]. The decision-maker also acts speculatively to arrive at a decision with an assumption of the decisions of other entities. But the extent of such assumption depends on the degree of collaboration of different entities [Marq17]. Furthermore, the decision variables of each entity are generally influenced by other entities' decisions [Hong08, Marq17]. In order for decentralized control to be effective and, more so, for the emergent behavior of the whole system to be consistent and predictable, the coordination of the supply chain is necessary where the operational decisions and activities are shared accurately and in time-bound manner with all the entities to avoid uncertainties.

Additionally, there is a third variety of extremely decentralized control structures, called **autonomous control systems**. Autonomous control methods are characterised by decentralised coordination of logistic objects in a heterarchically structured organisation (see Figure 2.3). These logistic objects are capable of processing information independently in order to take and execute decisions [Frei04].

Purely distributed approaches require a radical shift in the mindset to deal with the decision-making problems, which sometimes is difficult to apprehend and develop. The design of such distributed systems involves a high complexity of the system due to a high degree of interactions involved. The missing central component also causes some obstacles to industrial adoption. Therefore, in spite of emergence of these extremely distributed control structures a couple of decades ago, there are only a few research papers and much less industrial implementations. [Leit09, Jevt16, Kolb18]

2.2.2 Well-Known Decentralized Architectures for Production Control

Owing to the rigidity and low receptiveness to changes in manufacturing, centralized manufacturing practices are being replaced by decentralized manufacturing models. Decentralized models are dynamic in nature and demonstrate adaptive response to the changes in production orders - qualities that are key to competitiveness [Ueda04].

In the last few decades, various novel concepts were proposed to achieve decentralized production. Famous among them are Flexible Manufacturing Systems (FMS), Reconfigurable Manufacturing Systems (RMS), Holonic Manufacturing Systems (HMS) and Bionic Manu-

facturing Systems (BMS). The FMS and RMS are hardware-dominated architectures, and the HMS and BMS architectures are realized via software.

Hardware dominated Architectures

This section describes in brief the main architectures that heavily rely on physical system elements.

Dedicated Manufacturing Lines (DML) are a pre-requisite to understand and analyze the concepts of FMS and RMS. DML are based on fixed automation that produces a company's core products or parts over a long period and at high volume [Kore10]. The main driver of DML is the cost-effectiveness which is achieved through pre-planning and optimization [ElMa05].

FMS provide pre-planned flexibility for the production to react to both predicted and unpredicted changes [ElMa09]. The flexibility can be mainly categorized into two groups: routing and machine flexibility [Brow84]. Routing flexibility refers to the ability to change the order of operations performed on the product part, whereas resource flexibility refers to the ability to employ multiple manufacturing resources to perform the same operation on the product part. As per [ElMa05], "the objective of FMS is to cost-effectively manufacture several types of parts, within pre-defined product part families that can change over time, with minimum changeover cost, on the same system at the required volume and quality".

Typically, FMS consist of general-purpose computer-numerically-controlled (CNC) machines and other programmable forms of automation. Because CNC machines are characterized by single-tool operation, FMS throughput is much lower than that of dedicated lines. The combination of high equipment cost and low throughput makes the cost per part using FMS relatively high. Therefore, FMS production capacity is usually much lower than that of dedicated lines [Kore10].

In **RMS**, the system is designed for rapidly adjusting its production capacity and functionality through rearranging or changing its components to adapt to the changes of the manufacturing environment. This characteristic requires a reconfigurable hardware which enables to reconfigure both on the system structure level and on the machine level [Park12, Wien07, ElMa05]. Three features, namely capacity, functionality, and cost, differentiate the three types of manufacturing systems - RMS, DML and FMS. While FMS are usually constrained by the capacity-functionality factors, RMS are neither constrained by capacity nor by functionality, and are capable of changing over time in response to changing market circumstances [Kore10].

Software-Oriented Architectures

Long before the emergence of so-called Industrie 4.0 paradigm in Germany or Industrial Internet in USA, an international collaborative research program in manufacturing, called Intelligent Manufacturing Systems (IMS) [Comm96], was conceived in the early 1990s to advance manufacturing, help manufacturers compete on a global scale, and adopt next generation technologies in order to address the challenges arising from ever more demanding consumer market. Within this programme, several paradigms for the factories of the future were developed. The prominent among them are holonic, biological and fractal manufacturing systems [Leit09]. These theories exhibit similar concepts and characterisitcs in that they assume open, dynamic and reconfigurable systems where decisions are made and the production is carried out by independent and cooperative entities. But they are inspired from different sources: fractal manufacturing is grounded on mathematics [Thar96, Leit09], biological manufacturing paradigm is based on evolution and self-organization models of biological systems [Ueda97], and the holonic manufacturing paradigm is based on social organizations [Van 98].

Agent-Based Manufacturing Systems Agent technology is one of the important foundations in the field of artificial intelligence, and is also well-known and adopted in manufacturing applications. The study of agent technology began more than three decades ago, within the field of distributed artificial intelligence [Gire04]. The agent concept is applicable to a large range of domains, namely e-commerce, intelligent manufacturing, robotics and information systems. Nowadays, it is a very active area of research, and many commercial and industrial applications are based on it. Despite several definitions and interpretations, there is a general consensus regarding its three main abstractions [Mono06]:

- An agent is a computational system that is situated in a dynamic environment and is capable of exhibiting *autonomous* and *intelligent behavior*.
- An agent may have an environment that includes other agents. This community of agents *interact*, and as a whole, they operate as a *multi-agent system*.
- Agents are usually *adaptive* i.e., they are capable of tailoring their behavior to the changes in their environment in order to meet the set *objective*

Figure 2.4 illustrates the interaction of an agent with its sorrounding environment. The agent operates in an environment from which it is clearly separated. Hence, it makes unbiased observations about its environment with the aid of sensors. And finally, based on its priori knowledge and objectives, it initiates and executes actions using actuators to change the environment. Agents without physical embodiment receive inputs via either file contents, key strokes on keypad or network packets, and react appropriately in response [Russ16].

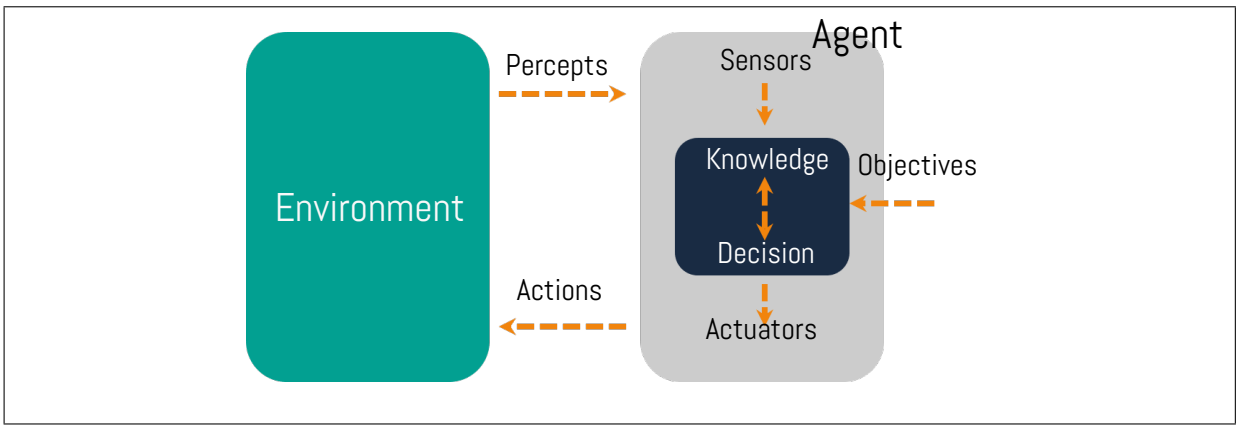

Figure 2.4: Interaction of an Agent with its Environment. Adapted from [Mono06, Russ16].

An agent can perform various tasks such as perception and execution, knowledge representation, reasoning, learning, coordination, and control, planning and scheduling [Shen03]. However, an agent need not incorporate all these functionalities. The design of an agent varies with requirements and application field.

A **Multi-Agent System (MAS)** is constituted by a network of agents which interact with each other. Within MAS, each agent owns only a partial model of sorrounding environment and follows its own objectives, which may differ from the objectives of other agents. However, due to the interaction of agents, MAS can provide solutions and emergent behaviors that are beyond the capabilities of individual agents [Mono06, Paru97b]. In the context of manufacturing, agent technology provides robustness, flexibility, reconfigurability, and plug and produce capability [Pěch08, Buss13, Merd09], and hence, is widely regarded as the enabling approach for future intelligent manufacturing systems [Jenn03, Buss13].

Early works in the field of manufacturing employing agent technology are AARIA [Paru97a] and MetaMorph [Matu99] architectures. Recent contributions in this area include the application of ontologies for advanced handling, exchanging and reasoning about knowledge in the area of industrial automation based on multi-agent technology [Vrba11].

HMS The holonic concept was first developed by the philosopher Arthur Koestler to explain the behavior of social systems [Koes67]. He postulated that these self-reliant social systems exist as both *wholes* and *parts*: almost everything is whole and part at once. These observations led Koestler to coin the word *holon* which is a composition of greek word *holos* meaning whole and greek suffix *on* meaning part [Koes67]. Koestler also observed two important characteristics of holons [Leit09]:

- *Autonomy*, where the stability of the holons result from their ability to act autonomously in case of unpredictable circumstances;
- *Co-operation*, which is the ability to have holons cooperating, transforming these holons into effective components of bigger wholes.

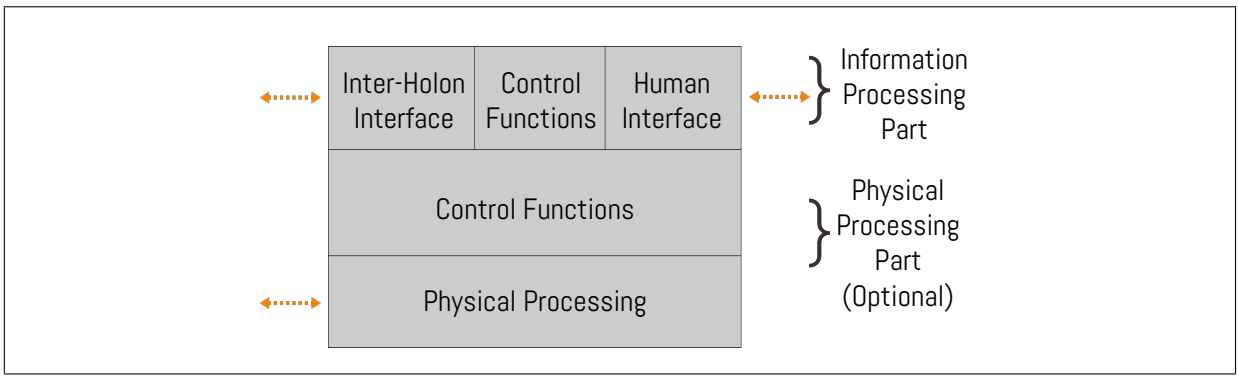

Figure 2.5: General Architecture of Holon [GIRE04].

Holonic manufacturing systems describe autonomous, self-reliant manufacturing units, called *holons*. Any manufacturing unit can be a holon, for example, factory, production line, manufacturing resource, product, conveyor belt, or production order. The holonic paradigm combines the high performance and predictability of centralized systems with the robustness of decentralized systems against disturbances. This is possible because a holon can function as a *whole*, which leads to increased robustness of the system. At the same time, it can also act as a *part* of a bigger *whole* forming a hierarchy with other holons for a certain period of time [VERS06].

A holon must be able to create and control the execution of its own plans and strategies. It contains an information processing part containing information about itself and its surrounding environment, and optionally, a physical processing part if it represents a physical device [LEIT09]. Figure 2.5 illustrates a general architecture of a physical holon. The physical interface of the holon connects to a device that performs the manufacturing operation based on control instructions originating from the holon.

A holarchy is defined as a system of holons, organized in a hierarchical structure, cooperating to achieve the system goals, by combining their individual skills and knowledge. Each holarchy has fixed rules and directives, and a holon can dynamically belong to multiple holarchies at the same time preserving its autonomy and individuality, which is an important difference to the traditional concept of hierarchies.[LEIT06a]

Several reference architectures have been proposed within the framework of HMS. A well-known holonic reference architecture for manufacturing systems, called Product-Resource-Order-Staff Architecture (PROSA) [VAN 98], was proposed within a project initiated at KU Leuven. It defines three types of basic holons: order holons, product holons and resource holons, which are structured using object-oriented concepts of aggregation and specialization. The product holon contains the process and product knowledge and describes the product to be manufactured; the order holon represents the production steps for manufacturing a concrete product; and the resource holon represents existing manufacturing resources on the shop-floor. ADAptive holonic COntrol aRchitecture (ADACOR) for distributed manufac-

turing systems is another familiar holonic manufacturing architecture that focuses on the job-shop production, characterized by concurrent and asynchronous processes with non-preemptive operations and alternative routings. The proposed adaptive architecture intends to be as decentralised as possible to tackle disruptions and as centralised as necessary for the sake of process optimization [Leit06b]. There are also numerous early works in this direction. [Ka98, Tsen97, Chir00] propose only two building blocks, namely order and resource holon, where order holon also integrates the functionalities of the product. More recently, an evolution to the ADACOR holonic control architecture, called $ADACOR^2$, was proposed insprired by biological theories [Barb15].

Distinction between Holon and Agent Though it appears that both the concepts have numerous similarities, there are also important distinctions. Agent technology has its roots in computer science (artificial intelligence namely), and holons in the CIM domain focusing on the problems associated with flexible manufacturing systems. Both holon and agent exhibit characteristics such as autonomy, cooperation, reorganization, reactivity, rationality, and learnability [Gire04]. In addition to these behavioral characteristics, holons also have some structural properties which agents lack. One of them is the property of *recursiveness* which allows to break-up an existing holon into several other holons, which in turn can be broken into further holons and so on until no further decomposition is possible or is a non-productive exercise. This feature allows the structural development of production control systems through the encapsulation of manufacturing services and components. However,

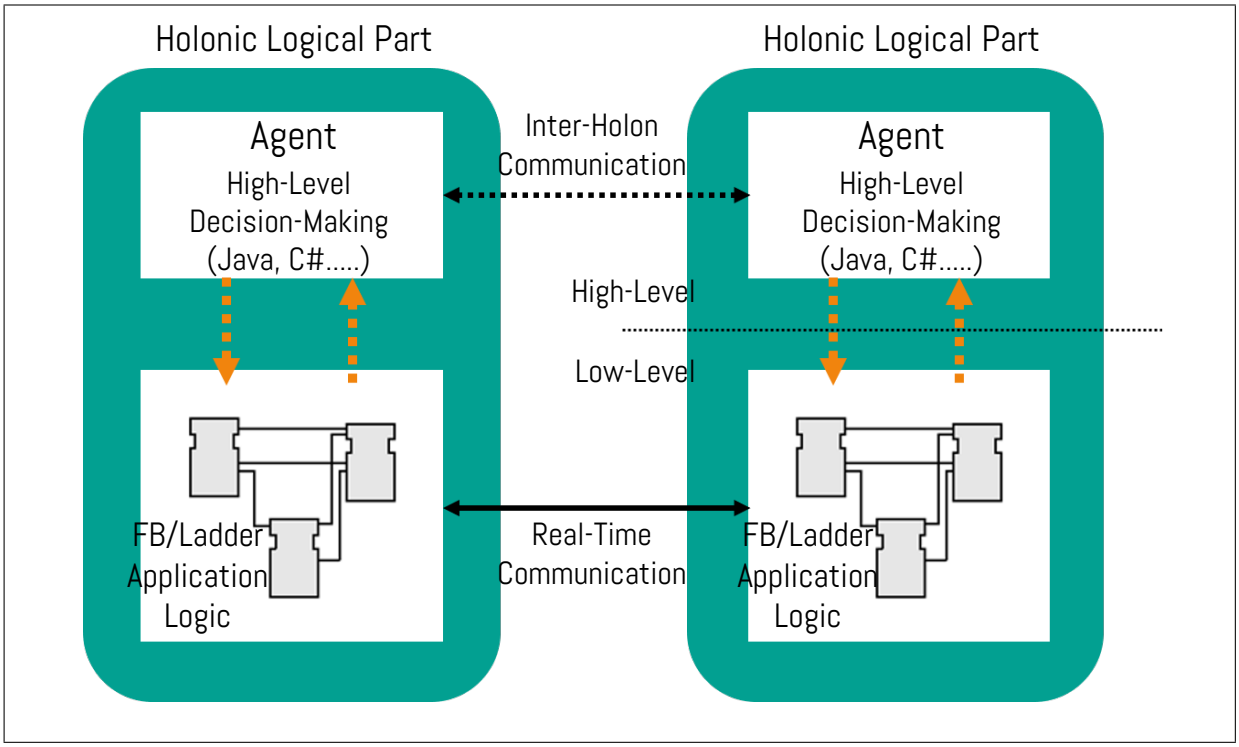

Figure 2.6: Implementation of Holon Information Processing Part using Agent Technology [Mari05].

when an agent represents a manufacturing resource, the focus is not on the integration of physical devices. Although both holon and agent contain an information processing part, an optional physical processing part of holon is missing in agent.

Additionally, a holon is a concept, and an agent is both a concept and a technology. Hence, the holonic concept can be implemented using agent technology to model behavioral characteristics such as modularity, decentralization, and reusability. The agent implements only the information processing part of a holon, and its physical processing part is connected to industrial PLCs to read data from sensors and send actions to actuators in real-time (refer Figure 2.6).

Market-Based Manufacturing Control using Agent Technology

Fundamentally, the task of deciding which manufacturing resource should be allocated to perform a specific manufacturing operation at a given time and context is viewed as manufacturing resource allocation problem. Here, the manufacturing resource is assumed to be modular and autonomous. Economics theories have influenced the manufacturing systems to decode the solution to this problem. Subsequently, it has resulted in a family of techniques known as *market-based manufacturing control* [Clea96] where resource allocation process is

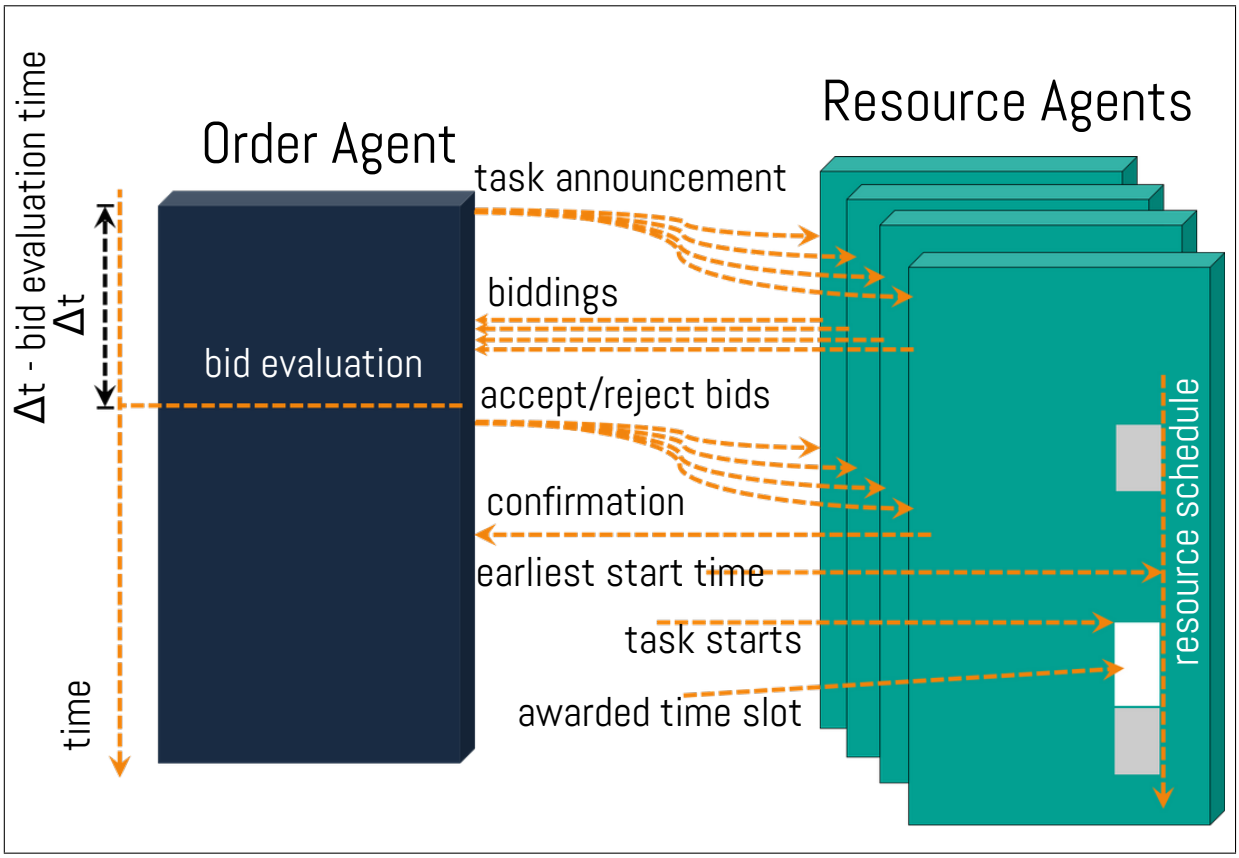

Figure 2.7: Market-Based Manufacturing Control [Mono06].

automated through software agents in a resource-constrained and market-like manufacturing environment. In a virtual economy, the software agents present the demand in the form of required manufacturing service and the matching manufacturing resources supply *resource time slots* [KUMA02] (refer Figure 2.7). The software agents are embedded with the custom bid evaluation algorithms to accept/reject bids. Upon confirmation of a bid and ensuing contractual agreements, the requestor and accepted supplier coordinate to agree upon a suitable timeframe for consumption/provision of the required manufacturing services, respectively.

Auctions are promising decentralized methods for agents-based scenario to allocate and reallocate manufacturing operations in dynamic, partially known and time-constrained domains with positive or negative synergies among operations. Auction-based coordination systems are also easy to understand, simple to implement and broadly applicable [KOEN10]. Due to this dynamic nature of the economy, the approach achieves higher levels of flexibility and scalability [MONO06]. This distributed scheduling and control helps to minimize network latency, maximize throughput and other system defined objectives, and at the same time adhere to constraints such as precedence and assignment restrictions.

2.3 DIN SPEC 91345: Reference Architecture Model for Industrie 4.0

This section briefly explains RAMI4.0, and shifts its focus to asset administration shell where communication is mandated to take place via OPC UA standard. Subsequently, the OPC UA standard specification and its information model is summarized. In status-quo evaluation, the inadequacy of current OPC UA specification to implement decentralized production control with regard to formal reasoning capabilities on its information model is described.

Introduction to the Architecture

In order to achieve a common understanding of standards and use cases which are necessary to realize the abstract concept of Industrie 4.0, it became necessary to develop an uniform architecture model as a reference, serving as a basis for the discussion of its interrelationships and details [ADOL15]. The Plattform Industrie 4.0 consisting of Zentralverband Elektrotechnik- und Elektronikindustrie e.V. (ZVEI), Verband Deutscher Maschinen- und Anlagenbau (VDMA), and Bundesverband Informationswirtschaft, Telekommunikation und neue Medien (BITKOM), has published a first version of a DIN SPEC 91345: Reference Architecture Model for Industrie 4.0 which provides a structured description of fundamental ideas and precisely describes Industrie 4.0-compliant production equipment (see Figure 2.8) [ADOL15].

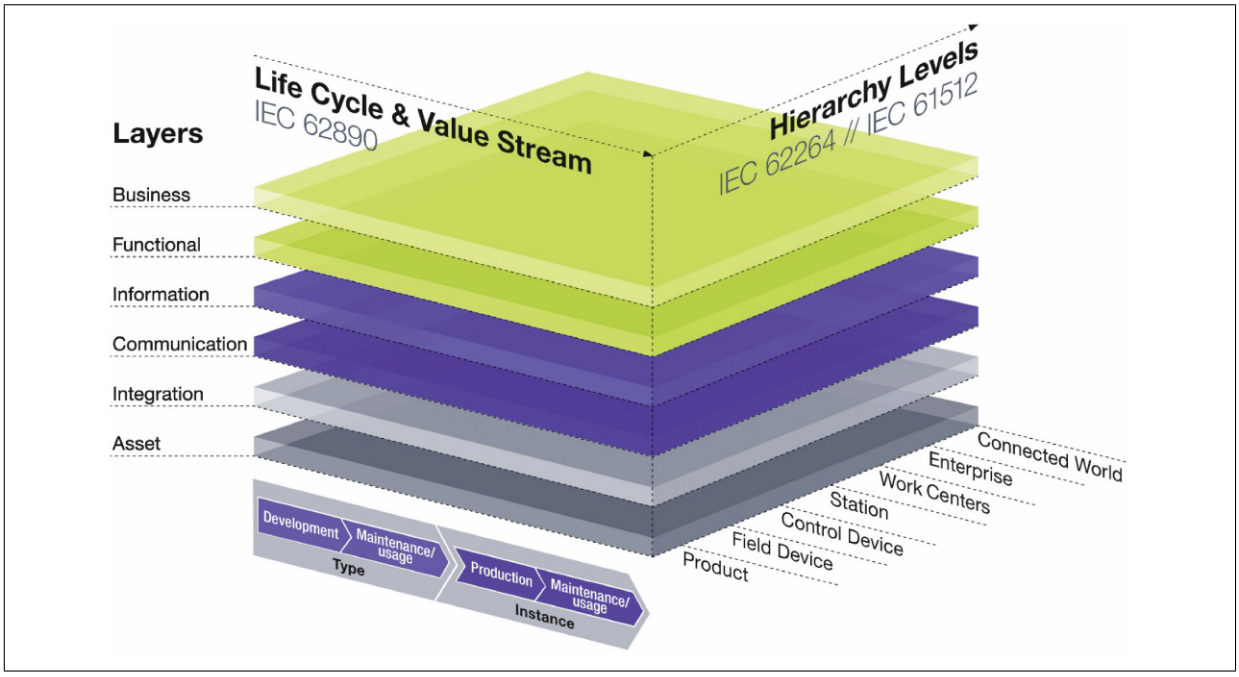

Figure 2.8: Reference architecture model for Industrie 4.0 [PLAT16].

RAMI4.0, which is represented in a three-dimensional coordinate system, describes all necessary aspects of a resource using a level model. The right horizontal axis shows the hierarchy levels defined in the IEC 62264 and IEC 61512 standards for enterprise IT and control systems. It also extends these hierarchical levels with the layers *Product* and *Connected World* that represent workpiece and inter-enterprise communication respectively [ADOL15]. In contrast, the IEC 62264 and IEC 61512 standards are intended for only intra-enterprise communication. The left horizontal axis represents the data acquisition along the entire life cycle of systems and products based on the IEC 62890 standard. RAMI4.0 makes a clear distinction between a type and an instance. A type becomes an instance when the design and prototyping is completed and the product is ready for being manufactured [ADOL15]. The vertical axis, also known as *architecture layers*, provides six layered representation of the information relevant to the role of the manufacturing resource. Description and implementation of highly flexible concepts can be realized by means of this reference architecture model. RAMI4.0 enables a step-by-step migration of the state of practice of present day factories to the vision of the Industrie 4.0 paradigm [ADOL15].

2.3.1 Industrie 4.0 Component (I4.0 Component)

Along with the RAMI4.0 layers, the Plattform Industrie 4.0 also defined the concept of an I4.0 Component in the DIN SPEC 91345. DIN SPEC 91345 defines an I4.0 Component as a "production system, an individual machine or unit, or a module within a machine". Despite that, a machine or machine unit cannot be automatically considered as an I4.0 Component unless it has at least a passive communication capability and has been equipped with an *asset*

administration shell (AAS) (refer Figure 2.9). An asset administration shell contains relevant information to represent an asset and its technical functionality. It shares the information regarding the asset with the outside world in a structure conforming to the guidelines laid down in RAMI4.0. Apart from providing with service capabilities, the AAS also provides semantic information of the managed asset. An equivocal semantics is required for seamless exchange of information between the I4.0 components. Additionally, an I4.0 component can be represented as a configuration of two or more I4.0 components using "the principle of recursive description of I4.0 components". This implies that any configuration of an asset can be digitally represented to any degree of granularity by describing structured assets, and combinations thereof, using the concept of I4.0 component. [ADOL15]

Asset Administration Shell (AAS)

AAS, or simply Administration Shell, is the root node of the I4.0 component postulated in DIN 91345. It is also refered to as *virtual entity* in [BEDE17] to better align within the concept of cyber physical systems (CPS) which have a physical and a virtual entity. The author believes the idea of AAS has its roots in the concept of holon which consists of an information processing part and a physical processing part.

With regards to the information processing part, the structure of an AAS is divided into two parts: a header and a body. The specification of a header is mandatory for an AAS. The header only consists of minimal information to identify the encapsulating AAS and its managed assets. Pointers to the security related aspects such as key sizes, authentication et cetera also form part of the header. A client can deduce the type and the instance information of asset(s) managed by an AAS by means of this header information.

The body of an AAS is a container for properties, supported views, services and references.

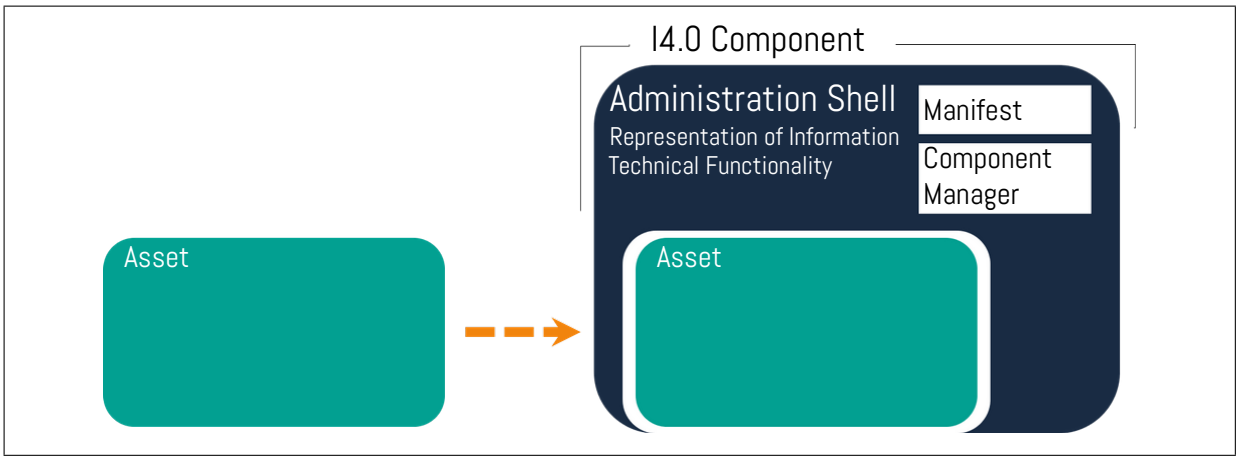

Figure 2.9: Description of I4.0 component containing an asset and its Administration Shell [PLAT16].

It contains all the information and functions to perform various operations on the managed assets. Static and dynamic **properties** of managed assets are represented in a key-value pair where the key is a globally unique identifier enabling semantics on data element look up, and the value characterizes the corresponding value of the property. In order to make exploration of these properties convenient for the client, related properties are grouped together. **Views** provide the means to filter the contents of an AAS so that only relevant data is shown as per roles and/or interest groups. The **service set** of an AAS can be invoked to execute certain functionality on or of the managed asset. Each service provides an interface to asset-specific functionality such as *close valve*, *drill hole* et cetera. Administrative services retrieve historical data or alarm conditions. Similar to properties, related service sets are also grouped together.

2.3.2 RAMI4.0 Communication Layer: IEC 62541 OPC UA Standard

OPC Unified Architecture (OPC UA) is a platform and vendor independent communication technology for a secure and reliable data exchange over the different levels of the industrial automation pyramid. In addition, the information models of the OPC UA standard provide the foundation for semantic interoperability [OPC 19]. OPC UA is the successor to the well-known and adopted OPC architecture originally designed by the OPC Foundation. OPC UA provides a number of benefits in comparison to its predecessor, namely platform independence, provision of internet and IP based communication protocols, built in security features, generic object model, extensible type system and scalability through profiles [CÂND10].

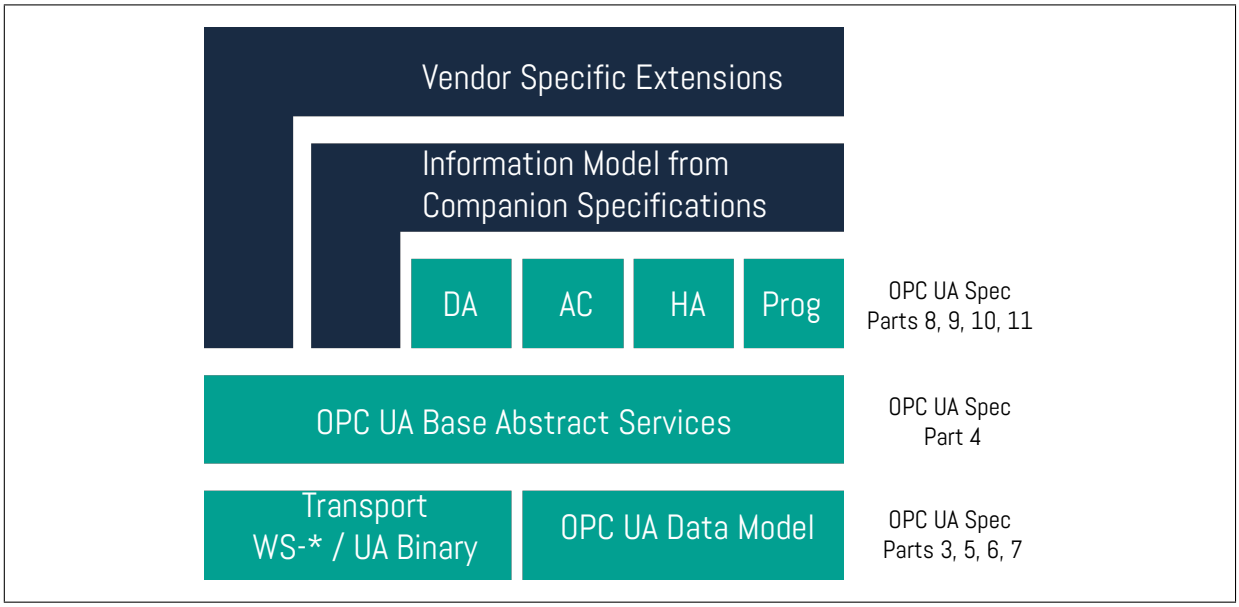

Figure 2.10: OPC UA Specification Stack [OPC 19].

The OPC UA specification is standardized as the IEC 62541 standard series containing 14 parts [OPC 19]. The first 7 parts are associated with the *core specifications*, namely, overview, security model, address space model, services, information model, service mappings and profiles. Parts 8 to 11 are called *access type specification parts*, and they provide backward compatibility to the classic OPC implementations with respect to data access, alarms and events, and historical access in the form of built-in *base information models*. OPC UA programs are *stateful* complex functions representing any level of functionality in an OPC UA server that can be invoked and managed by a client. Parts 12 to 14 are called *utility type specification parts*, and they are optional to set up an OPC UA platform. Parts 12 and 13 provide specification with regards to local OPC UA server discovery and OPC UA server aggregation functionalities respectively. An aggregated OPC UA server aggregates one or more OPC UA server(s) to provide relevant and compact information in its address space so that the client need not access several servers. OPC UA was originally designed as a client-server architecture, and recently the standard was extended to accommodate part 14 which provides a specification for the publish-subscribe messaging pattern. Figure 2.10 illustrates the multi-layered OPC UA architecture. The so-called *Companion specifications* or *OPC UA collaborations* also occupy a place in this OPC UA specification stack. These specifications are the result of an agreement of a consortium of industry partners to adopt a common information model extending the standard information model of the OPC UA server. In addition to the *Companion specifications*, some manufacturers also include custom add-ins into the server in order to differentiate from the market.

Cross-vendor data exchange is necessary for communication between machines, or between machines and workpieces. This requires unified semantics including a common syntax for data. Currently, the RAMI 4.0 recommends only the IEC standard 62541 OPC UA for implementing the *communication layer* (refer Figure 2.8) [Plat16, Adol15, Pauk16]. Moreover, Plattform Industrie 4.0 published a checklist for advertising and classifying Industrie 4.0 products into one of the following: *basic*, *ready* and *full*. To comply with the *Industrie 4.0 communication* and qualify for even the lowest *basic* criterions, the products and resources in the factory shop-floor must be addressable over the network via TCP/UDP or IP, and at least integrate the OPC UA information model [Hopp17, Adol15]. Besides, the information modeling capability, security, and scalability of OPC UA is unmatched to any existing similar technology solutions. Therefore, this thesis considers shop-floor communication exploiting OPC UA technology.

Information Modelling in OPC UA

Unlike the classic OPC DA, where the address space is built around folder objects and variables, OPC UA introduces the Object Oriented Paradigm (OOP) to the address space, where

objects can be defined with members such as attributes and methods, and instantiated objects can invoke these members (Refer Figure 2.11). It also allows to express the relationship between the object types. [OPC 17b]

Every object in the OPC UA is represented as a node in the address space. It uses a tree-based hierarchical representation where references are used to associate different nodes of the tree, thus providing a full-meshed network of nodes. A node can be categorized into eight different node classes which are specializations of the base node class. These include *Object*, *Variable*, *Method*, *View*, *ObjectType*, *VariableType*, *ReferenceType* and *DataType*. The address space model which is defined in part 3 of the specification series is the meta model of OPC UA.

Application-Agnostic Standard Services and Application-Specific Methods

With regards to management of this address space, OPC UA specification part 4 defines a fixed set of protocol and technology independent base services with parameters and behavior definition. All OPC UA servers should implement these services. The genericity of these services ensures interoperability. These so-called base service sets are divided according to functionality such as server discovery, server address space browsing, reading and writing node values, reading history of data and events, invocation of methods, subscription for data changes and events, and management of nodes and references. An OPC UA service itself is defined by its request and response messages, and thus, it is on the same level as an *operation* in a WSDL [Stop09].

Inside these wrapper application-agnostic services and the address space, application-specific

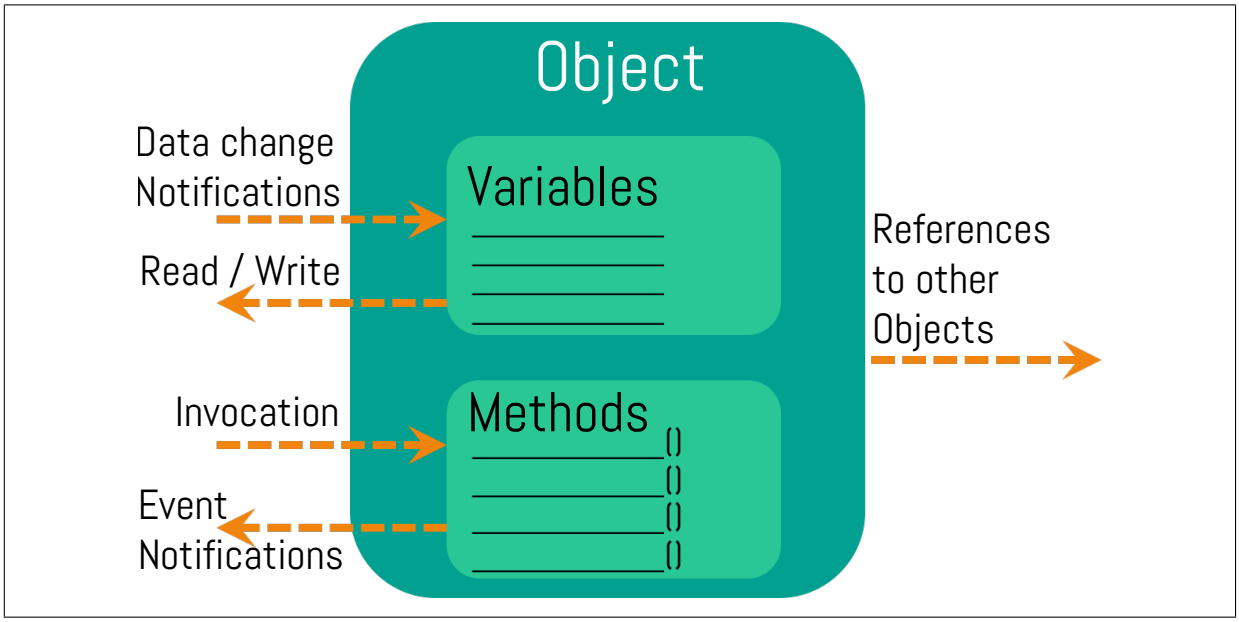

Figure 2.11: OPC UA Object Model. [OPC 17b].

methods are defined in the OPC UA server. These application-specific methods are the topic of interest of this thesis work. These methods are represented as *method* nodes which inherit from base *node* class, and they provide interfaces to perform specific application-related actions. A application-specific method, which is similar to a method of a class in an object-oriented programming, is a stateless entity, and invoked by an OPC UA client, proceed to completion on the server to return the result to the client. The OPC UA client can discover these methods defined in the server by browsing the owning objects references that identify their supported methods. A method node contains the metadata that identifies varying number of input and output arguments of the method, and such a method can be invoked by using the *call* service defined in part 4 of the specification series. [OPC 17b]

2.3.3 Status-Quo Evaluation

RAMI4.0 creates a collective understanding of norms, standards and case studies to realize the concept of industrie 4.0. In the event of non-availability of a suitable standard, it helps identify those gaps [Econ17]. To that end, the AAS of a device/resource can be characterized using the information model of OPC UA. Currently, the OPC UA architecture possesses only the base information model, and this vocabulary can be enhanced in the form of companion specifications. This informal semantics cannot enforce strict information modeling rules. In other words, these virtual resource representations should possess reasoning capabilities in order to make runtime decisions in decentralized manufacturing control. The information modeling in OPC UA relies on two fundamental concepts: node and reference. Based on the utility of the node, it can belong to one of the node classes, namely object, variable, method, referenceType, ObjectType, VariableType, datatype and view. These nodes are connected to one another with an OPC UA defined reference type(s). The OPC UA information model helps represent contextual information by linking a node in question to a related target concept node with the defined (and/or extended) reference type(s). In order to meet custom requirements, this linkage relation of objects can also be extended. The node that contains the reference is called the source node and the node that is referenced is called target node. A combination of the source and target nodes together with the reference type is used to uniquely identify references in OPC UA service requests. The target node can also be located in a different OPC UA server, which is referenced as the concatenation of the server name and the node ID. This concept of interconnecting the OPC UA objects to another object via reference provides a good alternative to the classical "Subject-Verb-Object" data representation of semantic technologies. OPC UA address space can express complex RDF triples like graphs with the concept of interconnection of fully qualified nodes with binary relationships tailored to the needs of the customer. The OPC UA information model provides a vocabulary that enables to model the hierarchical and non-hierarchical

relationships between the nodes. The flexibility provided by the OPC UA allows to design a scalable information model of the entities of the shop floor that participate in the product life cycle.

Notwithstanding the benefits of the rich information model of OPC UA described above, the OPC UA semantic vocabularies, in particular the hierarchical relationships, are only reasoned based on *dictionary of common understanding*, and therefore, do not enforce strict information modeling rules. Furthermore, the non-hierarchical relations in the OPC UA information model are not governed by any reasoning. Though OPC UA working groups have covered the bases with regards to most aspects of semantic information modeling, a reasoning engine that recognizes conflicting semantic definitions and infers logical consequences using the principles of first order predicate logic or description logics is still missing in OPC UA.

At present, the OPC UA, which is a standard intersection of IT and automation, does not possess reasoning capabilities which allow to make numerical and logical calculations that consequently assist in the design of consistency-check rules, and derive logical inferencing. This *intelligence* is required to reason on the services provided by manufacturing resources, assess the resource conditions for better coordination in the production, and evaluate the pre- and post-conditions of an OPC UA method execution in the PO orchestration.

Figure 1.3 provides an example of the above explanation where the OPC UA with its current capabilities can only answer questions with green tick bullets, but not crossed in red. Such conditions that not only involve connecting the mere references but also involve logical and reasoning expressions need to be represented in the information model at design time, and need to be processed at runtime to assess contextual information at factory shop-floor, and it is not possible with the current OPC UA specification.

The above-mentioned shortcoming in OPC UA can also be viewed from the perspective of the services offered by the AAS in the *communication layer* of RAMI 4.0. In order to interact with services offered by the AAS, in and out parameters, exceptional behaviors and interaction paradigm needs to be defined. These service signatures are still under discussion. The expert committee of Gesellschaft Mess- und Automatisierungstechnik 7.21 (GMA 7.21) has proposed a number of application-agnostic basis services. However, it agrees that, "Interactions with industrial assets are today often based on vendor-specific means. There exist no standards for application-specific services of AAS, nor respective service catalogs are known." It also muses, "it is still unclear whether a generic AAS model for all use cases is achievable and practical. Application-specific variants streamlined for certain resource constraints and communication needs may need to be investigated." [Bede17]

As per RAMI 4.0 suggestion, the OPC UA technology stack is not only employed in communication, but also to define an asset virtual representation [Bede17, Plat16, Pauk16]. Unlike

informal semantic descriptions of application-agnostic services of OPC UA/AAS which remain the same for all the servers, the application-specific methods of OPC UA/AAS are required to be reasoned about before invocation and hence, a formal definition is necessary. Formal representation allows the client to understand the context of manufacturing, the manufacturing service provided by the method, the required in/out parameters, and pre- and post-conditions of the method execution. This information, in turn, assists in creating adaptable manufacturing orchestration in the factory shop-floor.

2.4 Service Oriented Manufacturing and Semantic Web Technologies

This section describes service oriented architecture from the perspective of manufacturing and later, the concept of semantic web is explained. These concepts are used as a basis to introduce the theory of semantic web services that enable automatic manufacturing service discovery, and subsequent adaptive production orchestration, which is followed by the description of well-known semantic web service frameworks. The section ends with description of relevant previous work regarding application of software services in manufacturing.

2.4.1 Service Oriented Architecture

Starting at the turn of 21th century, researchers in the field of intelligent manufacturing shifted their attention to general IT technologies, namely service oriented architectures (SOA) [Shaw96] and semantic web technologies [Bern01]. SOA is an architectural paradigm for designing software in the form of reusable, loosely coupled and inter-operable software components, called services. SOA aims at defining interfaces, protocols, and data formats for

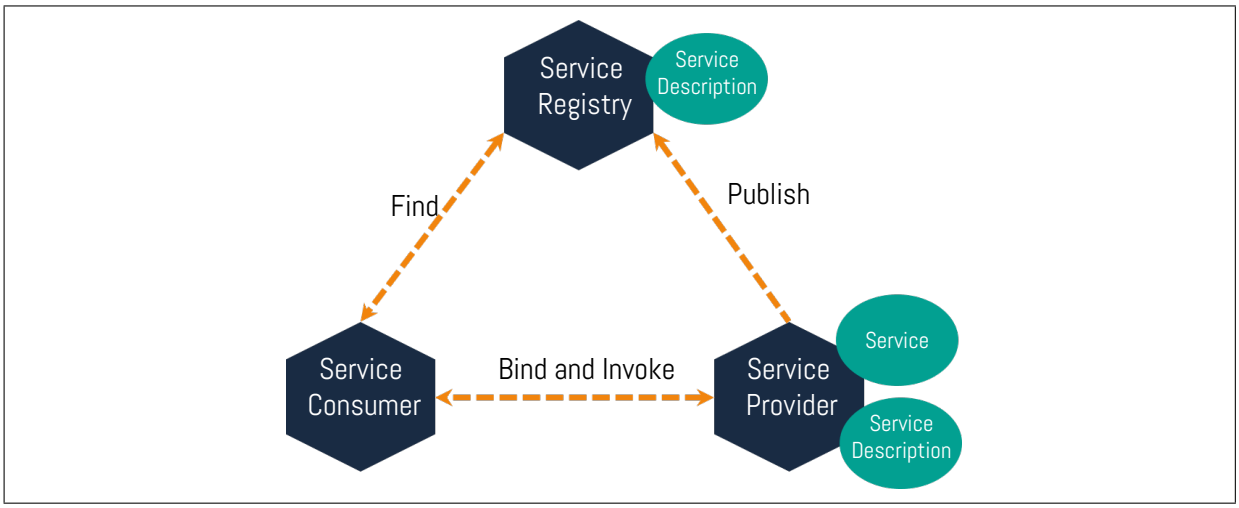

Figure 2.12: Interactions in Service-Oriented Architecture [Endr04].

accessing services in order to allow the developer to combine various services into a final application.

It involves the "publish, find, bind and invoke" collaboration cycle where a service consumer searches a service by querying the service registry with match criteria. In case of a match, the service registry provides the web service invocation details such as the web service interface contract and the end point address to the client (refer Figure 2.12). Some of the key characteristics of web Services are that they are inherently open and standard based, composable, language independent and interoperable [Endr04].

Evidently, MAS and SOA are based on the same principles. Alternatively, SOA is one of the enabler technologies to implement agents. In case of both MAS and SOA implementations, the application is composed of self-contained entities collaborating in a networked environment [Leit13]. For example, [Zhan11] does not distinguish between web services and agents. However, this work encapsulates the functionalities of an agent in web services.

2.4.2 Semantic Web

The term *Semantic Web* was coined by Tim Berners-Lee in [Bern01]. The Semantic Web is an extension of the *World Wide Web* through standards by the *World Wide Web Consortium* (W3C) [Brat07], and it is a collaborative effort led by the W3C with participation from a large number of researchers and industrial partners. The objective of these standards is the promotion of common data formats and exchange protocols on the web. It provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries.[Hawk12]

[Hitz10] identified three main topics as core concepts of the Semantic Web:

- *Building models* in order to describe knowledge about the world in abstract terms,
- *Calculating with knowledge* in order to use these knowledge models and their encoded information with reasoning machines that can draw meaningful conclusions,
- *exchanging complex information* in order to distribute, interlink, and reconcile knowledge on a global scale.

Semantic Web technology stack

The semantic web annotates machine processable data about documents and related entities on the web, and their relationship. The standards, technologies, rule languages such as eXtensible Markup Language (XML) [Bray97], Resource Description Framework (RDF)

[Lass99], RDF Schema (RDFS) [Bric04], Web Ontology Language (OWL) [OWL18a], Semantic Web Rule Language (SWRL) [SWR18a], SPARQL Protocol And RDF Query Language (SPARQL) [SPA18b] et cetera are the means to construct such an infrastructure. The semantic web architecture has XML at the bottom to make sure that it follows the common syntax and hence, enables data interoperability among the software applications. The semantics is provided by the upper layers of the architecture. **RDF** provides the core data representation structure by representing the information about real world objects in the form of graphs. RDF represents the information in the form of *subject-predicate-object* (SPO) format in pursuit of representing the graph data. The RDF Schema (RDF-S) language formalizes the RDF taxonomy by providing a set of classes, sub-classes, properties, sub-properties, domain and range restrictions in order to create a lightweight ontology. For example, RDF-S can express a class or a property as a sub-type of another more general type. Ideally, RDF-S is suitable to handle large amounts of data with a less expressive formalism. A more evolved and detailed ontology can only be created in **OWL**, which is derived in description logics (DL) [Baad08]. OWL is syntactically embedded into RDF and it provides additional standard vocabulary by offering more constructs over RDF-S. In order to express unconstrained knowledge representation and support the corresponding reasoning and calculations within finite time, the OWL language provides three increasingly expressive versions: OWL Lite (minimal classification hierarchy and simple constraint features),

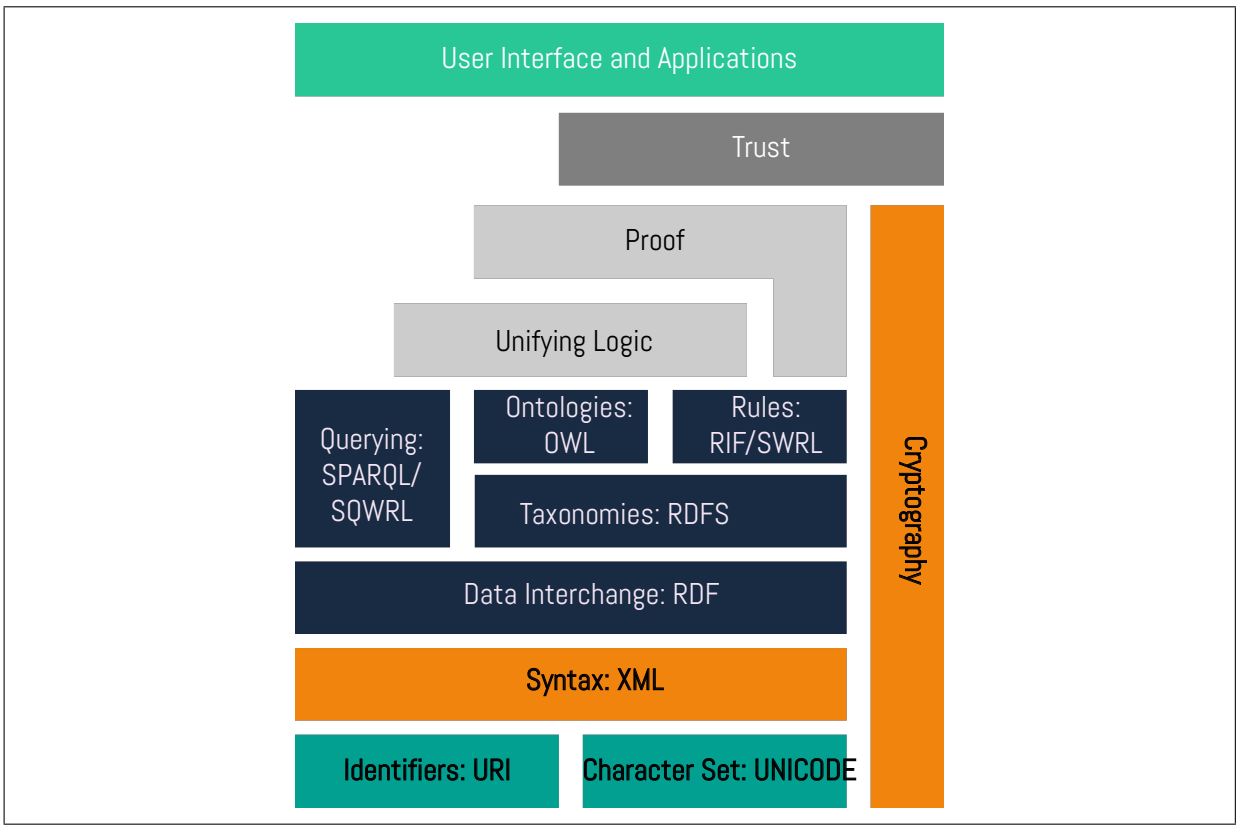

Figure 2.13: The Semantic Web technology stack [Horr05, Stev07].

OWL DL (expressivity with computational completeness and decidability) and OWL Full (maximum expressiveness with no computational guarantees). OWL 2 [OWL18a] which is a reengineered version of OWL also has two major dialects OWL 2 Full and OWL 2 DL whose underlying motivations are the same as in their earlier counterparts. However, OWL Lite is replced by three versions of OWL 2 profiles: OWL 2 EL, OWL 2 QL and OWL 2 RL (refer [OWL18b]), and each of these versions has various target use-cases. However, the increase in efficiency of these profiles comes at an expense of restricted expressivity. **SWRL** is a syntactic extension of OWL; it enhances the OWL-DL expressivity with its ability to formulate rules. **SPARQL** is a RDF query language for diverse data sources stored natively as RDF or viewed as RDF via middleware. All these components of the semantic web along with their relation to one another are represented in the so-called Semantic Web layer cake or Semantic Web technology stack (see Figure 2.13).

The semantic Web has numerous application fields which span financial, public and governmental institutions, e-Commerce, automotive, health care, life sciences, oil and gas, pharmaceutical and telecommunications industries. The application areas are data integration, semantic annotation, natural language interface, text mining, domain modeling, improved searching, content discovery and management, service integration and management, and schema mapping.[Bake12, Lége05, Camb12]

2.4.3 Semantic Web Services (SWS)

Web services were formally designed for human interpretation and utilization. At the onset, Business-to-business (B2B) and e-Commerce applications increasingly employed web services for data exchange. The web service search and the decision of its qualification to fulfil the requirements was made at design time by the humans. In most cases, the web services interoperation was achieved via hard-coded information-extraction code to locate and extract content. This resulted in tight coupling between the web client applications and service interface, even though the server implementation itself was decoupled from the clients in SOA [McIl01, Haki]. This scenario faces multiple difficulties [Bary10, Fens11b, McIl01]:

- Increasing size of web service repository
- Various web services designed and implemented from different points of view and technologies introduce challenges during service composition in terms of mismatches with regards to functionality, in/out parameters, and other side-effects
- Dealing with evolving information such as revisions to the existing web services in terms of both functional and non-functional characteristics due to corresponding changes to the service contract

- Unpredictability of web services characteristics such as downtime and replacement of old services

The hand-written code on the client side either broke or needed to be changed in each of the above-listed scenarios.

Early implementation of web services was based on the SOAP messaging protocol which is grounded in XML. The Web Service Description Language (WSDL) [Chri01] is a W3C recommendation which provides a formal, syntactic and machine readable description of SOAP-based web services. However, the description of the service interface and its capabilities in WSDL is merely syntactic in nature. As a result, there is a semantic gap between the syntactic description of a service and the underlying meaning. This semantic gap makes an automated discovery, selection and orchestration of appropriate services almost impossible [Losk11], and makes them susceptible to human intervention.

Fundamental to having computer programs or agents implement reliable, large-scale interoperation of web services is the need to make such services computer interpretable. To this end, properties, capabilities, interfaces, and effects must be encoded in an unambiguous and machine-understandable form [McIl01]. This led to the development of SWS which are the convergence of semantic web concepts with service oriented computing (refer Figure 2.14). SWS address the major challenge of automated, interoperable and meaningful coordination of web services to be carried out by intelligent software agents [Klus08].

Even in the context of manufacturing, the semantic description of manufacturing services enables dynamic discovery, efficient reuse and (semi-)automatic orchestration of manufacturing services to build higher value production processes [Losk11]. In this context, SWS also realize the automation of market-based manufacturing control described in Section 2.2.2.

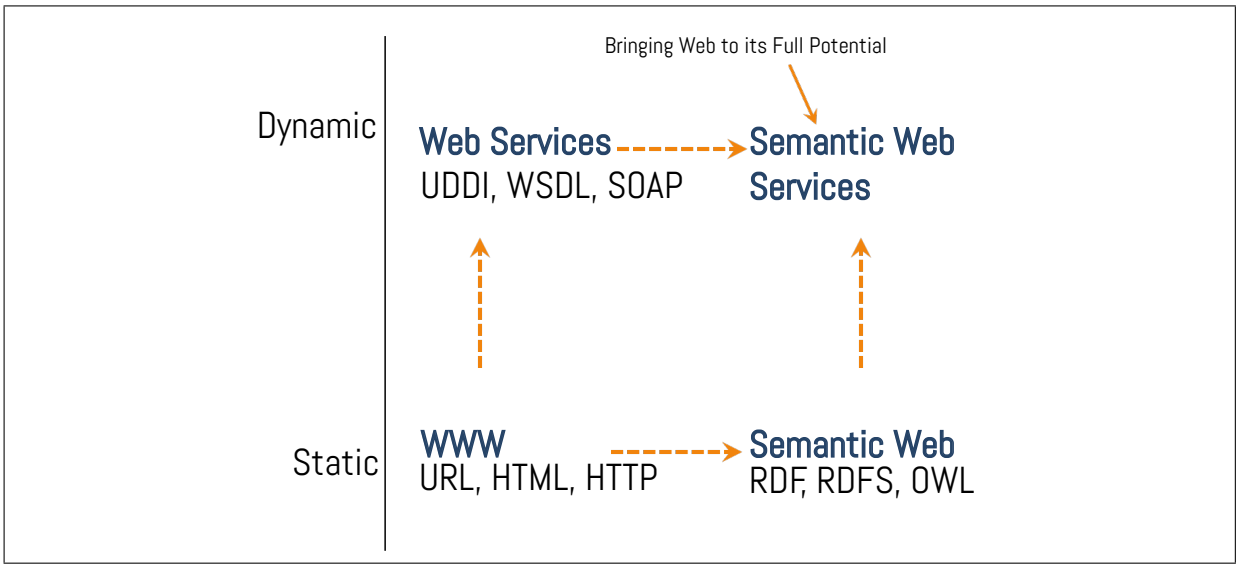

Figure 2.14: Vision from WWW to Semantic-Web Services [Fens11b, Fens18].

Formal Reasoning in Semantic Service Descriptions SWS combine different technological concepts such as web services, semantic web, and automated logical reasoning. The formal description of web services opens new challenges: first, the identification of SWS that meet the requirements by web clients, and second, dynamic composition of SWS to construct services of higher complexity. In order to address these challenges, the web services should be described in terms of their capabilities, and necessary frameworks need to conceptualized containing algorithms that match the service description to the service request. The basic idea of formally grounded descriptions of web services is to enable the client to understand the functional and non-functional semantics through appropriate logic-based reasoning [Klus08]. The concept expressions used to describe the service input and output parameters are assumed to build up from the basic concepts of the common ontologies referred by both the service provider and the requester.

SWS Frameworks

The focus of SWS is to transform web services from an interface-oriented description into a meaning-oriented description which supports automatic discovery, composition, invocation and interoperation of services [Varg05]. W3C recommends the use of software agents to automate the service discovery tasks with the aid of formal specifications using ontologies containing domain operational knowledge. Technologies and frameworks are needed to support human experts to augment the web service descriptions with such formal semantic ontologies. This led to an investigation in both academia and industry to describe semantic web services and the corresponding operational environment. There are numerous research initiatives in this direction such as OWL-S, WSMO, SAWSDL and SWSF [Batt05]. The following sub-sections provide a brief overview of important SWS frameworks.

OWL-S is an OWL-based ontology framework of the semantic web to describe SWS, and it enables the agent-based framework to discover, orchestrate and invoke the SWS. It comprises three main sub-ontologies, namely service profile, process model and service grounding (refer Figure 2.15). The service profile advertises the service functionality, the process model provides the detailed description of the service and the service grounding provides concrete details to communicate with the service instance. Specifically, the process model also describes the service composition (both orchestration and choreography) of one or more services. The process corresponding to the process model in OWL-S can be atomic, simple or composite. An atomic process corresponds to an instance of service operation, whereas a composite process is a composition of various atomic processes constructed with various control flow operations like if-then-else, while-until, sequence, split and split-join. A simple process is a service or process abstraction which has to be realized by an atomic process. [Mart04]

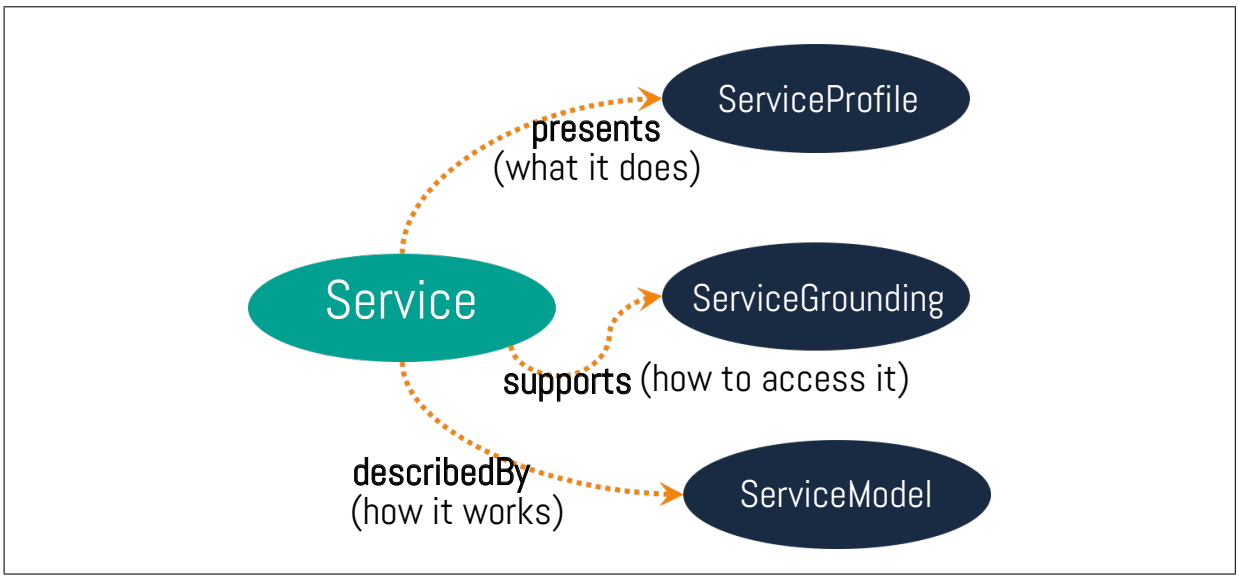

Figure 2.15: Architecture of OWL-S Ontology [Mart04].

SAWSDL Semantic Annotations for WSDL (SAWSDL) [Kope07] is an incremental bottom-up mechanism of modeling SWS and is a W3C recommendation. However, it does not specify an ontology language for semantic service description, but only specifies mechanisms to semantically annotate the ontologies defined either within or outside the scope of WSDL to the WSDL description elements. These annotations are attached to the WSDL using WSDL extension elements. The SAWSDL specification defines two types of extension attributes on the web service concepts, namely modelReference and schemaMapping. The former correlates a WSDL element with a concept in the semantic model, and the latter extension attribute is employed for bi-directional XML and semantic data transformation. [Kope07]

The semantic annotation that points to semantic concepts can be applied on interface, operation, faults, input and output parameters, and binding levels. The schema lowering mechanism maps the semantic model to the XML that facilitates service invocation, whereas schema lifting does the reverse transformation. These formal concepts enable in automatic service discovery, and subsequent dynamic orchestration of web services. Listing 2.1 illus-

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

Listing 2.1: Example of SAWSDL Annotations on a Web Service Operation and its Parameters.

trates an example of semantic annotations on a manufacturing operation called *welding*.

The SAWSDL specification neither expresses a precise semantic model nor does it direct the ways of exploiting the semantically annotated XML data. It is left as the prerogative of the user or the software agent to make an informed decision. Moreover, the lack of specification to state the preconditions and effects of services in SAWSDL hinders its ability to compose services.

Web Service Modeling Ontology (WSMO) WSMO is an evolution of the Web Service Modeling Framework (WSMF) which is a result of various European Commission funded research projects in the domain of SWS [Klus08]. WSMO provides the conceptual underpinning and a formal language for semantically describing all relevant aspects of Web services in order to facilitate the automatization of discovering, combining and invoking electronic services over the Web. Although both OWL-S and WSMO provide orchestration capabilities [Losk13], WSMO has a different approach of mediation where it tries to resolve potential mismatches in the representation of source and target ontologies.

The overall structure of WSMO is described by its four main components (refer Figure 2.16) which are described briefly in the following [Roma05]:

- *Ontologies* provide terminology to be used by other components to describe the relevant aspects of the domains of discourse.
- *Web Services* represent computational entities that, in turn, provide access to the some value added services. Web service descriptions comprise the capabilities, interfaces and

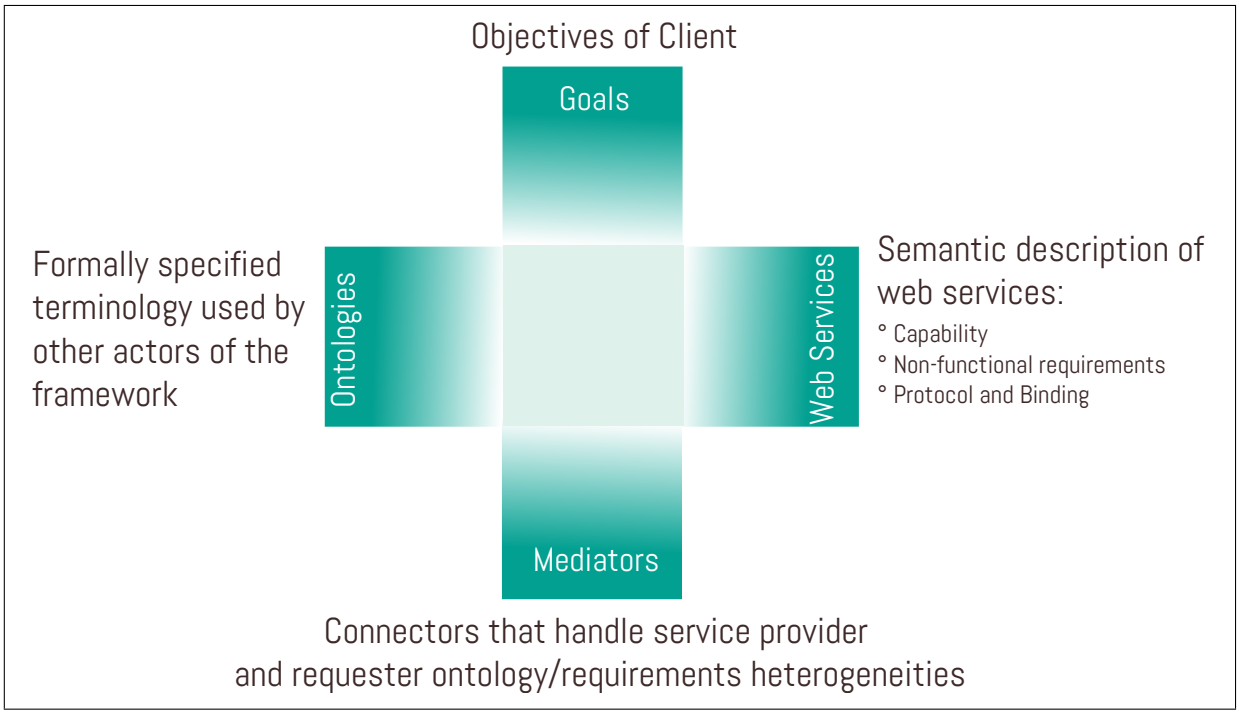

Figure 2.16: Top level elements of WSMO. Adapted from [Roma05]

internal workings of the service using the terminology defined in the ontologies.

- *Goals* represent the requirements of service requesters.
- *Mediators* deal with interoperability problems on the data (different terminology), process (composition of web services) and protocol (communication between the web services) level among the WSMO components.

The WSMO framework also provides a formal language called web services modeling language (WSML) for the definition of logical statements in the framework. WSMO framework also developed a reference architecture, called Web Service Execution Environment (WSMX). Besides the explicit modelling of mediators, the separate modelling of offered and required web services is one of the central characteristics of WSMO [Losk13]. However, the core problem of the approach is the definition of an own ontology and web service description language which diverges from standards adopted by the W3C such as OWL and WSDL [Klus08, Losk13].

2.4.4 Previous Works

This section briefly discusses the past project undertakings focusing on industrial service oriented architectures.

The Service Infrastructure for Real-time Embedded Networked Applications (**SIRENA**) project [Bohn06] leverages Service Oriented Architectures to seamlessly interconnect embedded devices within and between the industrial, telecommunication, automotive and home automation domains. In order to assure interoperability and extensibility of embedded devices, a core framework was developed as a part of this project, called **Devices Profile for Web Services** (DPWS) [Orga09]. Since its adoption as OASIS standard, it is a common web service middleware and profiles for devices [Zeeb07]. DPWS proposed the usage of WS-* protocols for device networking. It establishes two fundamental elements, namely, the device and the hosted services on it. The device participates in meta data exchange and hence, enables its discovery, and the services provide the functional behavior of the device. Later, DPWS became underpinning for subsequent service oriented architectures in industrial automation. The Service-Oriented Device and Delivery Architectures (**SODA**) project [Deug06] also works on the same theme of *modeling embedded devices as web services*. **SOCRADES** [SOC09] is one of the earlier and well-known projects that focused on the development of industrial systems based on the SOA paradigm both at the field device and the application level. The main features of this project are direct access to devices, service discovery, legacy device integration, middleware historian, security and formal semantics support, and service composition [De S08]. With respect to SCADA systems, the **IMC-AESOP** project [IMC14] implemented a set of monitoring and control functions ex-

posed as *services* in automation clouds. This project marks a change in interactions among the different systems, applications and users through a fusion of SOA, cloud computing and cyber physical systems [Colo14].

The rapid adoption of service oriented approaches also resulted in the increasing popularity of SWS in manufacturing. Enriched by semantics that are capable of being processed by machines, SWS gained instant traction since they could efficiently exploit the services on the web without human intervention. SWS assist in services registration, exploitation of contextual information, evaluation, discovery, and dynamic service composition [Losk13]. To date, a huge volume of literature has been published on the subject of applying SWS in the manufacturing domain. The concept of introducing ontologies in manufacturing as the state of the art was reviewed in [Borg04]. The facilitation of dynamic orchestration of operational processes in the shop floor by SWS is the reason for its widespread adoption in factory automation. There are also several research papers, for example, [Izag11] that focus on purely syntactical level orchestration which is suitable only for static workflow requirements. However, these approaches fail where adaptive process planning is the key requirement of the production. The idea of replacing the low-level programming of sensors and actuators with the high-level programming of the manufacturing resources with the application of SWS was recognized in [Lobo09]. [Losk13] presented adaptive orchestration of manufacturing processes using the OWL-S framework. Recently, [Chen17] showed that a high degree of customization and reconfiguration of the system is possible through ontology-based web services. [Rohj11] enhanced abstract services covered in part 4 of the OPC UA specification with semantics for discovery of servers based on location in the field of smart energy grids. This work employed the WSMO framework for service discovery. Embedded Multi-Core Systems for Mixed Criticality Applications in Dynamic and Changeable Real-Time Environments (**EMC²**) [Emb14] is an Artemis [Art09] project with focus on safe and cost-efficient cyber physical systems using modern microelectronics [Webe16]. Its aim is to develop an innovative and sustainable service-oriented architecture which is suitable to handle the following scenarios [Scho17, Webe16]:

- Dynamic adaptability in open systems, scalability and utmost flexibility,
- Utilization of expensive system features only as service-on-demand in order to reduce the overall system cost,
- Handling of mixed criticality applications under real-time conditions,
- Full scale deployment and management of integrated tool chains

The **RACE** project introduces a centralized platform computer (CPC) which is inspired by the approach adopted in the avionics and automation domains [Beck15]. The CPC establishes a generic safety-critical execution environment for applications, providing interfaces for

test, verification and reliable communication infrastructure to smart sensors and actuators. The CPC also significantly reduces the complexity of integration and verification of applications, and enables the support for the Plug and Produce feature. The above-mentioned projects and research works demonstrate the feasibility of embedment of web services at device level, and integration of these devices with business information systems at upper levels of the enterprise architecture.

The most recent development in this area is matrix production which is based on categorized, standardized and connected production cells. The principle of matrix production is the separation of logistics and production which ensures smooth execution of wide variety of processes related to different product variants. The main features of such a production facility are modularity, autonomy and flexibility, and the manufacturers claim that such a system systematically fulfils the requirements of Industrie 4.0 in industrial manufacturing [Smar18].

3 System Architecture

This chapter rearranges the deployment locations of the functional modules of cloud MES based on the task criterion. The modules related to design time activities of production are retained in the cloud infrastructure, whereas the production runtime tasks related modules are moved to the production edge layer. Section 3.1.2 discusses the various production scenarios the edge component should address. Finally, representation of such a realignment of basic cloud MES modules is illustrated in Section 3.2.

3.1 Caching of Production Scheduling and Control Information

Given the possibilities of IT technologies, this thesis reconsiders current monolithic structure of the software modules of MES in order to reduce the network latency of shop-floor communication with the cloud based MES, and subsequently, realize decentralized decision-making in the production. It recommends breaking down the monolithic MES system into simpler autonomous components, and deploying those software components to the proximity of the production site. A subsequent requirement is the smooth interaction between these distributed partial systems of MES. From an implementation perspective, though this extension of cloud based MES to the production network involves additional communication layers between the MES modules, at the same time the exercise leads to the development of optimally tailored decision support systems which derive conclusions based on contextual information of the factory shop-floor. To that end, this research proposes introducing an edge layer, called Generic Shop-Floor Connector (GeSCo), which is an extension of cloud MES at the edge of a production network layer.

The MES in the cloud is not guaranteed to be close to the site of production. As explained in Section 1.1, the network latency is directly proportional to the geographic distance. Due to the physical proximity of GeSCo and shop-floor, and subsequent local area network (LAN) communication, the network latency is short as data packets need not cross multiple routers. A small amount of data corresponding to one to few PO(s) in GeSCo also effectively nullifies the computation time penalty witnessed in the cloud MES platform. Additionally, GeSCo also alleviates the problem of latency introduced by the virtualization layer of the cloud

infrastructure. Consequently, GeSCo is an ideal place to store the cached data.

3.1.1 Characteristics of GeSCo

GeSCo should be close to, but not tightly coupled to the shop-floor. In its basic conception, it should control the production processes, and collect the data to and from the shop-floor and enterprise business software. Therefore, it must provide web service capabilities to send/receive the information from business software systems (vertical integration), and should support a wide variety of industrial communication protocols in order to connect to a wide variety of industry specific data sources of diverse manufacturers, such as OPC UA, classical OPC and http based web services (horizontal integration) (refer Figure 3.1).

The cached data constitutes an abstract production schedule created in the cloud MES infrastructure which facilitates in production execution and control data of part of/complete/multiple PO(s). GeSCo collaborates with enterprise software and diverse industrial data sources to execute this cached PO by performing division of labor in the shop-floor. That is, it autonomously evaluates and chooses the concrete manufacturing services offered by the resources, and allocates the production operations to the corresponding resources on the shop floor at run-time based on the production recipe. Such information empowers GeSCo to take decisions with regard to production control without consulting the centralized cloud MES, and hence, it facilitates the implementation of decentralized control in the production execution. Besides, the introduction of GeSCo in the shop-floor is not to take over the role of SCADA. Instead, GeSCo should just serve as a thin client to the centralized cloud

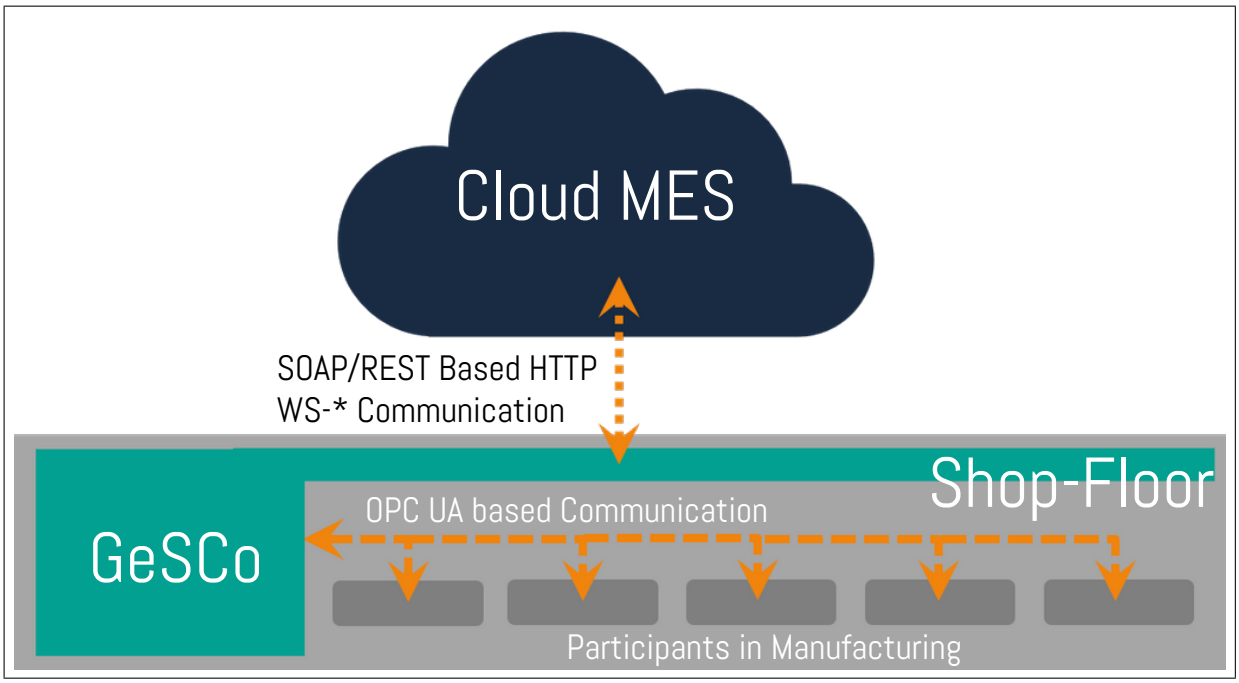

Figure 3.1: GeSCo Communication Landscape.

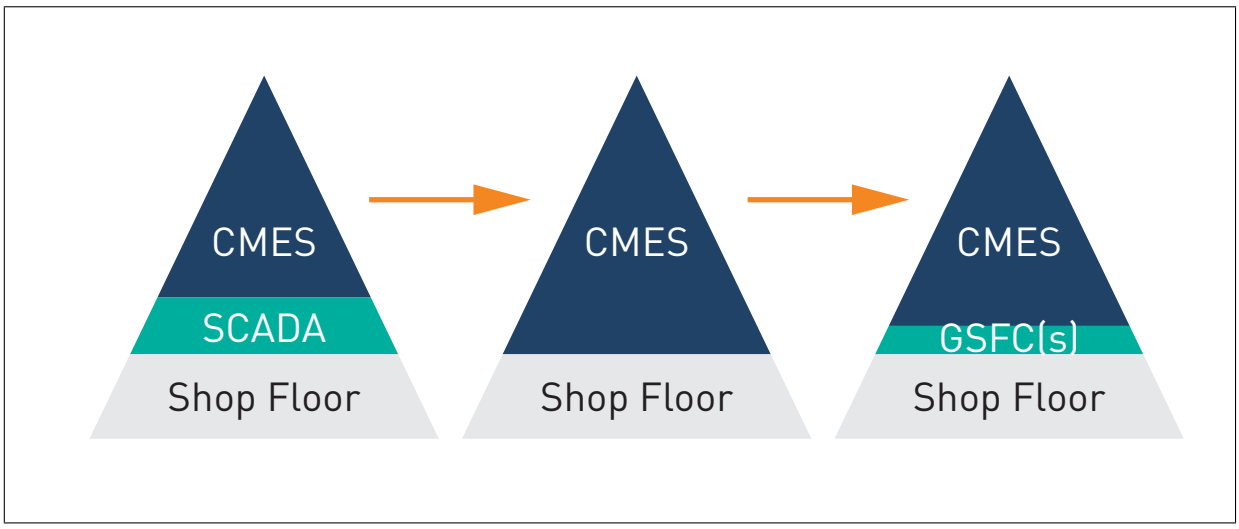

Figure 3.2: Evolution of Cloud MES - Shop Floor Connectivity [Katt18b].

MES server. Based on these arguments, the cloud MES and the shop-floor communication evolution can be illustrated as in Figure 3.2.

3.1.2 Anticipated challenges due to caching

Following the relocation of modules of cloud MES related to runtime production tasks to production edge, and subsequent caching of the production control data, the intention is to reduce the communication between the GeSCo and cloud MES as far as possible. Several exceptional situations may arise in the shop-floor while the GeSCo is in control of the production execution. Manufacturing resource breakdown is one such case in point which is a highly disruptive occurrence in an automated production environment. Even as preventive maintenance or repair is a preferable way to increase the system reliability and significantly reduce the system cost, [Chiu10] claims that, machine breakdowns are inevitable in real-life manufacturing systems. The GeSCo should anticipate such an eventuality and must be well equipped to take an appropriate course of action.

The current manufacturing operation cannot be swapped to another manufacturing resource when there are no alternative manufacturing resources in the shop floor cell. In such a case, the GeSCo should preempt all the other steps of the routing and retain its state. Under such an abort/resume policy in case of random manufacturing resource breakdown, production should resume with the processing of the preempted step of the routing after the breakdown is fixed. Even when GeSCo has started the execution of another PO of a different product variant with no dependency on the resource which has broken down, it should resume the execution of the aborted PO after completion of the current PO.

In a job shop environment, the presence of multi purpose manufacturing resources enables to execute multiple operations on several alternative resources. In such a scenario, GeSCo

must reschedule the production routing by replacing the disrupted resource with an alternative resource. In the event of manufacturing resource replacement, the new manufacturing resource should be introduced easily and quickly into the manufacturing system without reprogramming or reconfiguration of the production setup. The GeSCo should be resilient to such *plug and produce* concepts.

When the PO is changed in the ERP during execution, the cloud MES should deliver the necessary information promptly to GeSCo. The GeSCo should check the feasibility of the changed PO, take appropriate measures and convey the same to the upper layers of the automation pyramid. Under normal circumstances, the GeSCO should adopt the First-In-First-Out (FIFO) policy for the execution of a PO. However, when the production routing consists of manufacturing operations of different lead times, the priority order in the pipeline should finish the execution earliest. Therefore, the provision should be made in GeSCo by defining a priority policy to put a non-priority order on *hold* state in order to expedite the execution of the priority order.

The traditional MES creates static production routing where manufacturing resources and operations are coupled together, and pushed down to shop-floor execution. This approach does not allow the edge component the freedom to make decisions at the shop-floor. In case of deviation from the production planning, the edge component seeks directions from cloud MES to recover from the path of deviation to successfully perform all the activities of production. In very dynamic scenarios such as high-speed manufacturing, more autonomy of the edge layer is desired in order to enable it to react to unforeseen events. However, in order to provide more autonomy to the GeSCo, the cloud MES should only create the abstract production planning without tying the manufacturing operations to resources. This process should be performed in GeSCo. The GeSCo should possess local intelligence during the dispatch of manufacturing operations to manufacturing resources. In addition to their *reachable property*, modern manufacturing resources also known as Cyber Physical Production Systems (CPPS), have more computing power to complement a large number of embedded sensors and actuators. These resources can track their state, PO buffer, and are aware of their various configurations to manufacture products with unique characteristics. In state-of-the-art factories, the shop-floor is considered to be a service market place where different manufacturing requirements are matched against the corresponding services offered by the resources to produce a tailored product defined by the customer.

It should be the responsibility of GeSCo to associate each operation of a PO to a particular manufacturing resource, in order to process a semi-finished assembly, also called CPS. Thus, the dispatched operation characterizes the logical binding between the CPS and CPPS. The changes brought about by this combination of the CPS, CPPS and GeSCo that has the relevant contextual information of the current POs drive changes in manufacturing produc-

tion and control, and actuate the remodeling of centralized to truly decentralized production decision-making systems.

The idea is not to store complete informational and operational technology information in the GeSCo to make these runtime decisions. The provision should be made where the manufacturing resources publish their capabilities to the GeSCo. The GeSCo should utilize this information to assign a routing step to one of the manufacturing resources. The dispatcher module of cloud MES should also push the required abstract services in case of quality non-conformance along with the non-conformance codes. In the event of quality non-conformance, the GeSCo only looks up the non-conformance code, and seeks the corresponding services from the manufacturing resources. Another major challenge is to make feasible decisions taking into account the physical configuration of the cells of the shop-floor.

In all the above-mentioned exceptional situations, the GeSCo should either resolve or find an alternative course of actions. The objective of this exercise is the successful completion of the production execution. The cloud MES should support this goal by sending meaningful data at the right time.

Challenges of Integration of GeSCo: A Survey

The GeSCo should assume the role of the cloud MES after the PO is transferred to its cache. The transfer of production control to the GeSCo is smooth under normal circumstances when the production encounters no problems. However, the system should be designed such that it should be robust against production fluctuations, and should mitigate or solve the problems that may arise under exceptional circumstances.

In order to determine which responsibilities such a system must fulfill, several experts in the field of manufacturing were asked to prioritize the challenges for GeSCo during the execution of shop orders. The results of this survey in the descending order of their weighted average are as following:

1. Determination of next routing step since business rules that govern the routing decisions are present in the cloud MES
2. Semantic translation of data arriving from cloud MES to a technology and business agnostic solution such as GeSCo
3. Adaptation in GeSCo in the event of change of the data model in the centralized cloud MES
4. Determination of suitable resources to perform the current operation
5. Routing-path substitution in the event of machine breakdown

6. Dealing with the change of the PO
7. Handling the POs of high priority
8. Course of action in the event of quality defects
9. Resumption of production after a disruption due to unforeseen circumstances
10. Course of action in the event of unavailability of raw materials
11. Distributed manufacturing where components are being manufactured at different sites

3.2 Proposed System Architecture

The production management consists of activities such as planning, scheduling, dispatching, production execution management, data collection, and tracking. These tasks are segregated into design time and production runtime activities. Design time activities are retained in centralized cloud systems, and the functional modules that are responsible for production runtime are moved to the edge of the production network layer to react to the local conditions on the factory shop-floor. The solution architecture also takes into account the challenges mentioned in Section 3.1.2. It should enable the cloud MES to exercise control over the production process while at the same time ensure a smooth integration of the GeSCo for providing flexibility in exceptional cases. Hence, both centralized and decentralized aspects should be incorporated into the architecture.

MES Components in the Cloud

This section describes the set of basic building blocks and services that are required in the cloud from the view point of the production management (refer Figure 3.3).

Production Planning System This application layer enables the human production planner to plan the production sequence with generic requirements. To this end, it has different user interfaces that help define and maintain the plant and product definition, operation planning, and production execution aspects. This master data facilitates the design of BoM and the shop-floor routing for a product variant. This unit also enables the human to create and release the PO to the edge component (GeSCo) on the shop-floor.

Manufacturing Resource Model and Servitization Remote resource sharing and management is a challenge to the cloud MES since it is geographically separated from the shop-floor. The resource virtualization is the key idea behind building the cloud services in the context of manufacturing. The resource model is the transformation of a real manufacturing

resource to a virtual or logical resource. Each manufacturing resource is modeled formally with a set of inputs and outputs according to its main functionality. The functional and non-functional capabilities of the resource can be semantically modeled. The model is then subjected to *one-to-one*, *one-to-many* or *many-to-one* real-to-virtual mapping methods to map to a logical resource [Ren12].

The concept of enriching digital plant models by making virtual copies of the manufacturing resources with near real time data from sensors also makes the information flow more transparent. Virtual resource servitization is the transformation of abstract concepts of capabilities provided by these resources into formal services that are understandable by the cloud platform [Ren12]. This process involves several aspects such as definition of the service model, message model, ports and protocols. The service model includes the template for the service offered by the cloud platform. The reception of inputs and generation of outputs of the service is defined in the message modeling process. The port modeling involves the definition of functional operation port used to accomplish the operation target. The protocol binding specifies the different protocols that are supported by the service.

This service interface of the virtual resource enables GeSCo to store the resource relevant data in a realistic resource model, also called resource digital twin [Rose15]. The GeSCo periodically collects the data from manufacturing resource, and pushes it to the resource model in the cloud MES. This assists in real time monitoring of the manufacturing resource for the purpose of tracking the status, understanding its behavior with regard to its interaction with other manufacturing systems, and also in calculation of the equipment effectiveness. Further, the data is archived and the aggregated historical data is fed to a predictive analytics tool to derive insights into the resource behavior.

Dispatcher The PO created and released by the production planner is digitized and transferred from the cloud MES to the manufacturing facilities by the dispatcher. The algorithm for transferring the priority order(s) should also be pre-loaded into the dispatcher. The parameters that expedite the release and subsequent transfer to the shop-floor are production end date, priority customer, and inventory and manufacturing resource availability. The GeSCo, introduced in this research work, is a technology and business agnostic solution. Therefore, the dispatcher should send unambiguous data, for example, a collaborative product definition and operations semantic model to the GeSCo. The GeSCo translates this information to its compatible data model for further processing.

Data mining and predictive analytics Instead of relying on human expertise alone, there is an increasing inclination towards aggregating and processing a large amount of data at the shop-floor, which in turn enables MES to train better models for classification, clustering and prediction. This component analyzes the current and past semi-structured or unstructured

data, extracts useful patterns and transfers this knowledge to GeSCo. This knowledge of past experience is then helpful for GeSCo to take run-time decisions that solve or mitigate the problems arising in the shop-floor during production. This information is also helpful to achieve optimization of the production processes in the shop-floor.

Information Systems This constituent stores the digitized version of production (shop) orders with a detailed scheduling plan of all the products created by the human production designer. It also stores the product genealogy including complete work instructions, components and phantom assemblies, operation flow and routing, manufacturing resources and work centers employed, bill of materials, activities on the shop-floor, rework instructions and the discrepancies. One possible methodology of realization of product genealogy was proposed in [Haup13], called Digital Object Memory (DOMe), which maintains all the information about a product instance over its production lifecycle, where each product is identified and tracked using an RFID tag that contains the unique shop-floor control number. Since a *DOMe* is centrally accessible to all the involved entities of production, it enables production coordination among these entities, compilation of historic manufacturing reports, quality investigations and process improvements.

Components of GeSCo

The production management tasks that can be decentrally organized are incorporated into modules which are part of GeSCo. To overcome the problems of network latency and connectivity associated with cloud MES, the production control should be delegated to the manufacturing edge layer and hence, the argument of decentralization of the decision-making process is even more applicable to this narrative. To this end, an edge layer that caches the production execution and control data is designed, and a comprehensive architecture is designed to integrate this edge layer with the cloud MES. GeSCo consists of the following components with dedicated responsibilities:

Manufacturing Resource Perception Layer To achieve harmonization among various manufacturing resources, they need to be coupled together. The perception layer undertakes this responsibility of loose coupling of different resources on the shop-floor. The different manufacturing resources at the site also register themselves to this layer. The registration can take place via publishing either the resource meta-data or the resource endpoint. In case of the latter, the resource endpoint permits the perception layer to browse the resource data structures to extract the meta-data of the resource. To this end, this GeSCo module has an internal sub-module known as Capability Discovery Repository (CDR) (sometimes also referred as Generic Method Discovery Repository (GMDR) in the thesis) which stores the capabilities of the various manufacturing resources. The manufacturing resources are also al-

lowed to directly announce all their capabilities semantically to the CDR. However, a formal explicit specification of shared concepts and relationship among those concepts, also called ontology, needs to be modeled at the organization level in order to realize the semantic publishing of the capabilities. A static service, which provides access to the created ontologies, should enable referencing and dereferencing of the semantic concepts. The decentralization facilitator exploits this semantic information from the CDR to arrive at the decisions at runtime. Since this thesis considers OPC UA for industrial communication, the authors propose to adopt the semantic web services frameworks to the OPC UA application specific methods in order to automate the process of method discovery and subsequent method composition. Details are provided in Chapter 4.

Purely from a communication perspective, the perception layer should support the standard industrial communication protocols, such as OPC UA, classic OPC and HTTP based web service stack. These protocols are employed to perceive different manufacturing resources with an intent to enable intelligent identification, detection, communication, tracking, monitoring and management. The effectiveness of this exercise depends on the ability of this layer to extract the key information from the real resources.

Production Control Data Cache This component stores the data delivered by the cloud MES. It contains the blueprint of the production execution on the shop-floor, which is the detailed routing information in the case of discrete manufacturing. Various entities of GeSCo such as decentralization facilitator and production engine base their decisions and actions on this cached production scheduling and control data.

Decentralization Facilitator This entity enables the decentralization in manufacturing by coordinating with various manufacturing resource models and cloud MES, and thus helps address the challenge of determining the suitable resources for a particular operation. Specifically, this layer refers to the collection of virtual manufacturing resources maintained by the method discovery repository for the run-time classification of resources. This, in turn, aids in on-demand resource capability matching. The virtual resource management helps the decentralization facilitator identify capabilities intelligently by semantically searching for suitable services and the corresponding manufacturing resources on the shop-floor to meet the production requirements. Detailed theory and implementation aspects of method discovery and ensuing method orchestration are described in Section 6.3.

Common Semantic Model A homogeneous production ontology generated in the cloud based MES must be distributed to the all the entities participating in the manufacturing including GeSCo. This ontology must include a fact-oriented and formal conceptual data model and the corresponding instances. When different distributed applications model their information model based on this reference ontology, the meaning behind the data ex-

change between the applications can be discerned and consequently, this interpretation of data guarantees interoperability of these applications. The Common Semantic Model (CSM) component stores a copy of such semantic model.

In line with the above explanation, the metadata of resources on the shop-floor that expresses their manufacturing capabilities should be modeled using these common semantics. In the same manner, the PO should be created with abstract manufacturing operations requirements without tying it to concrete resources. The CSM component assists the decentralization facilitator in determining the suitable resources for specific production steps taking into account also the contextual information at production runtime.

Exception Handler This block of the GeSCo is accountable for overcoming shortcomings that arise in the production environment. These shortcomings are explained in Section 3.1.2, numbers 5 to 9. The exception handler either attempts to find an alternate course of action by local coordination, or seeks further instructions from the centralized MES which has a global picture of the system.

Production Engine and Work-In-Progress Monitor The production engine is the heart of the GeSCo that collaborates with all the other components of GeSCo to achieve the end goal of successful completion of the PO. It fetches the PO information and routing details from the production control data cache, and delegates the responsibility of matching the manufacturing resources for the given operation to the decentralization facilitator. After identification of suitable services and the corresponding participating manufacturing resources to carry out production sub-steps, the production engine refers the ordered sets of operations and creates manufacturing service composition accordingly. It then delegates each of the jobs of the composition plan to the perception layer that assigns the operations to the real resources after the necessary configuration. The production engine also assigns the unique PO identifier to the smart product or the product carrier at the start of the PO so that the carrier can be identified and tracked any time during production. During the dispatch of each routing step of a PO, the manufacturing operation harnesses the unique CPS identifier, and binds the product to the manufacturing resource. The PO is put on hold in the event of non-availability of default and alternate resources, and is only resumed after the required resource registers to the perception layer.

To ensure the production is running as expected, it is necessary to monitor the run-time status and respond to changes. In case of changes and exceptions, this layer coordinates with decentralization facilitator and exception handler to solve or mitigate the contingency. The production engine also has the intelligence to recognize the situations where GeSCo cannot take the optimal decision based on local information. In such scenarios, it seeks the master data, the singular source of truth, stored in centralized cloud MES.

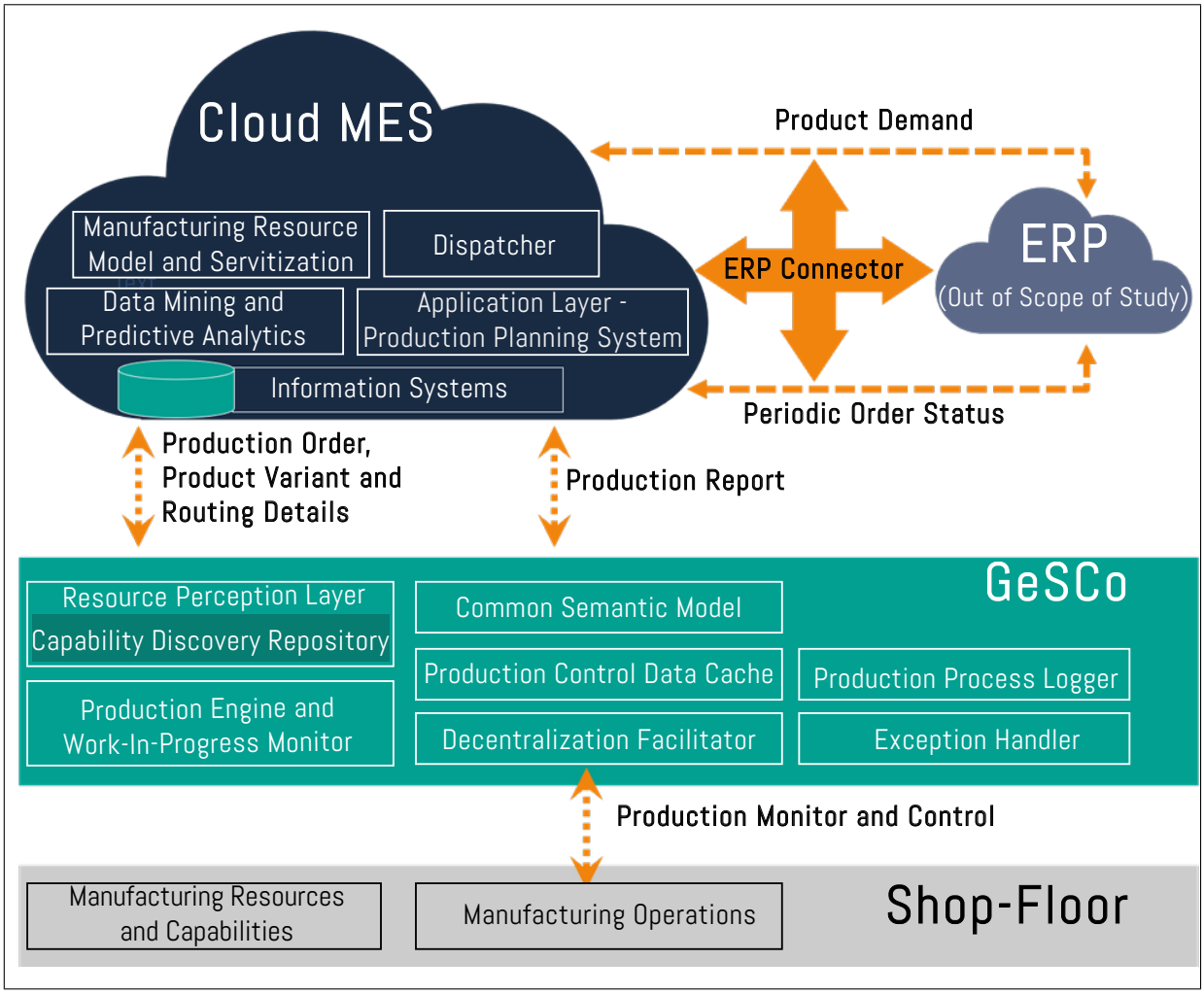

Figure 3.3: Integration of GeSCo with cloud MES [Katt18b].

Production Process Logger This component uploads the data corresponding to a variety of production processes or manufacturing steps it gathers during the production onto the cloud MES. This unstructured data is subjected to analysis, and an effort is made by the cloud MES to find patterns in previous behaviors of resources, and transform it into a structured data. This knowledge in turn can be channeled as feedback to the MES-GeSCo closed loop system in order to optimize the production in the long run. The data is also stored for retrospective tracking of product quality and evidence purposes.

Based on this discussion, the entire landscape can be constructed as illustrated in Figure 3.3. However, in order to simulate the manufacturing conditions of distributed manufacturing where components are manufactured at different sites requires multiple test beds, and logistic support such as automatic guided vehicles. The non-availability of such multiple manufacturing islands coupled with the lower priority of the corresponding requirement from the survey necessitated to the omission of the last requirement presented in Section 3.1.2. With the exception of this last requirement, the research work tries to find the solution with the above-defined solution architecture. Caching of abstract PO(s) and production

ontology at GeSCo, coupled with the introduction of a semantic framework for OPC UA application specific methods discovery and composition empowers GeSCo to make local production scheduling and control decisions at runtime, and thereby also effectively addresses the problem of network latency experienced in pure cloud based manufacturing.

4 Semantic Modeling

In today's factories, the field device level services are abstracted by composition and integration into higher level services. Hence, the intricate details of lowest layer of automation are hidden employing standard communication interfaces and functional encapsulation of field devices. This characteristic implies that the services offered by the manufacturing resources can be directly invoked without the burden of understanding complex Programmable Logic Controller (PLC) programming. From GeSCo point of view, this encapsulation necessitates application specific programming only at the last layer of abstraction as illustrated in Figure 4.1.

In the context of the thesis, there are two classes of application specific methods in OPC UA, namely methods that perform business or manufacturing tasks, and methods that retrieve the status information of resources on a number of aspects such as their current temperature, pressure, energy consumption, PO queue et cetera. These two types of OPC UA application methods are composed by GeSCo according to the PO requirements in order to form complex operation sequences of higher order.

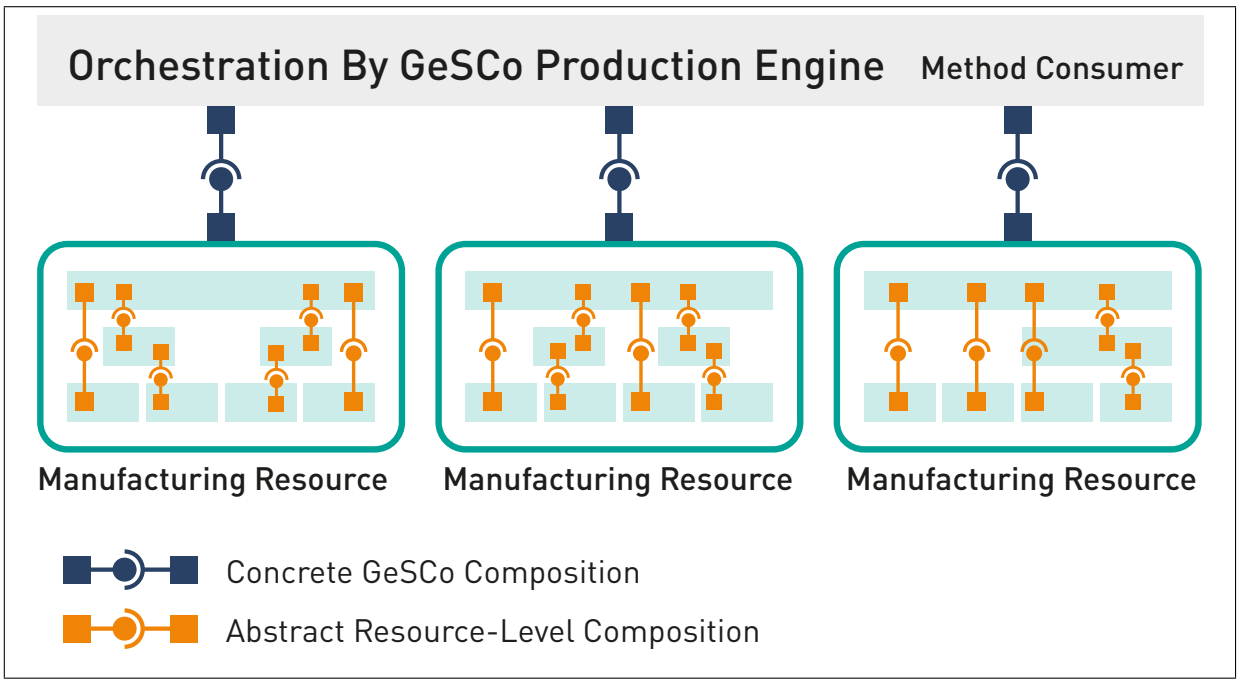

Figure 4.1: Service Orientation in Manufacturing Resources [KATT18d].

This chapter is organized into four sections. Section 4 builds a case to integrate formal ontologies into the GeSCo functional modules in general, and the OPC UA information model in particular. Sections 4.1 and 4.2 describe the adoption of the well-known OWL-S and SAWSDL semantic web service frameworks to application specific methods of OPC UA server, respectively. Subsequently, Section 4.3 presents an improved hybrid version of the above-mentioned semantic web service frameworks. The author attempts to overcome the insufficiency of OPC UA (refer Section 2.3.3) by augmenting it with a reasoning engine based on description logics. Additionally, the service oriented architecture provided by OPC UA is decoupled from the actual implementation details in order that the vendors can choose a communication protocol of their preference. Hence, an effort is made to develop a semantic OPC UA solution that is independent of the underlying communication protocol details. With the introduction of these novel semantic OPC UA approaches, it is possible to generate a flexible orchestration plan of manufacturing operations which can be changed in case of unforeseen events in production with well-founded semantic reasoning of the OPC UA methods. Additionally, the intelligent planning algorithms introduced by the formal description of manufacturing services using semantic OPC UA infrastructure facilitates the implementation of loosely coupled production systems, and thereby provides plug and produce and smooth reconfigurability features.

Ontologies in Manufacturing

The touch points of this research work range from production design to production planning, execution and control. Though these three stages of manufacturing are distinct in theory, in reality the boundaries of each of these activities come into contact with each other. In order to achieve higher efficiency, integration of these stages is essential. Ontologies make it possible by making the data interoperable across different stages. The development of ontologies which allow to configure the complete manufacturing system using a model-based engineering approach is the step in the right direction for the advancement of domain and contextual knowledge. At the same time, when they are designed in certain specific ontology construction languages such as Web Ontology Language (OWL) and Resource Description Framework (RDF), they also permit effective assimilation of such knowledge in software and agent-based automation systems. Additionally, the deductive reasoning provided by the inference engine of OWL ontology can be better than human interpretation and transform the facts into a source of smart data [Fort14]. Accordingly, this research work proposes to model the manufacturing and internal logistics system structure into an ontology which is machine processable, and subsequently enables integration of knowledge within automated systems.

A decentralized decision-making process permits quick response to production requirements

and is also resilient to changing manufacturing environments [Zhan17] (refer Section 2.2.1 for brief overview). To achieve such a smooth production monitoring and control, GeSCo should be equipped to handle the following scenarios:

- Decentralized production line control
- Decentralized material flow control

This results in circumstance-based supervision and regulation of manufacturing resources and product (including basic and work-in-progress sub-assemblies) resulting in dynamic synergy between these participating manufacturing systems. To this end, the production engine component of GeSCo which is in charge of the manufacturing process orchestration needs to structure and manage the evolving domain knowledge and track the real time process data to take time bound decisions during the control of the industrial cyber physical systems. To summarize, the author employs the ontology in this work for the following purposes:

- Automation of data assimilation among different software units
- Formal representation of relationships between various concepts of the production information model, and creation of relevant rules that need to be adhered to in the production
- Gain contextual awareness
- Assist GeSCo in decision-making activities to generate an on-the-fly configuration of the production processes that consists of dynamic orchestration of manufacturing services provided by resources
- Decentralized manufacturing planning and control without an upfront knowledge of the factory layout

As described in Section 2.3.2, this thesis considers the OPC UA technology for resource-resource, and GeSCo-resource communications (refer Figure 3.1). The next sub-section briefly describes the shortcomings of the OPC UA specification with regards to the provision of formal semantics, and a solution is proposed to describe the semantic model of OPC UA servers using a DL-based ontology language such as OWL.

Formal Semantics with regards to Dynamic Method Invocation

Information exchange and usage between disparate manufacturing resources is possible if common data formats and exchange protocols constituted by semantic technologies are employed. In a service-oriented manufacturing paradigm, OPC UA methods of manufacturing resources are invoked to consume their manufacturing services. Therefore, the concept of semantic information modeling of OPC UA should also be extended to the methods of the

OPC UA server.

The following enumeration lists the aspects related to automated discovery, composition and invocation of an OPC UA method from a software agent point of view:

1. Application methods of OPC UA servers of all the relevant manufacturing resources in the shop-floor should provide the semantic description of capabilities that can be interpreted by a computer.
2. In order to invoke the method in an OPC UA address space, the client needs to know the browse path of the method node. This information is more relevant in OPC UA methods than in web service operations. The web service operations are invoked by operation names. In contrast, in OPC UA application specific methods, the client requires the method node ID, the parent node ID of the method and method arguments as per the OPC UA specification.
3. During and between the method invocation(s), the pre- and post-conditions that represent the current and the subsequent state of the method execution context must be modeled with strict definition enforcement. From a production runtime perspective, before dispatching the routing step of a PO to a manufacturing resource, GeSCo, which is in charge of production orchestration in the shop-floor, should query the manufacturing resource for its state, PO capacity, current PO queue at its work station and product-specific business conditions in order to evaluate its feasibility to the PO task at hand.
4. The endpoint URL, in conjunction with the server security policy and message security mode, of each of the OPC UA servers is also one of the requirements for the automatic invocation of an application method.

This thesis focuses on OPC UA aspects that need further research with regards to its information model. For example, above-listed item 2 deals with automatic method invocation, and items 1 and 3 deal with formal semantic capabilities for the purpose of autonomous discovery and composition of manufacturing services. The item 4 refers security concepts which are outside the purview of this research work. The complex information arriving from the multitude of manufacturing entities must be aggregated and normalized by subjecting to numerical and/or logical operations before interpretation. In order to enable the OPC UA client to access the above information at manufacturing runtime, the semantic model of the OPC UA server should be described, for example, in a computational logic-based language such as Web Ontology Language (OWL) that can be subjected to consistency checks and inference. The ontological language that is grounded on description logics enables to create a set of axioms that provide explicit logical assertions about the concepts (class), individuals, and object and data properties. Additional rules on the properties of class instances (indi-

viduals) can be modeled using a Rule Markup Language, for instance, SWRL, SQWRL and SPARQL. When the reasoner engine takes over the modeled ontology and generated rules in SWRL, it infers other implicit logical assertions contained in the ontology. Such a rule based ontology capacitates GeSCo to compute the values and states of all the participating entities, and thereby, have a holistic view of the production to make smart decisions.

The application-specific methods of the OPC UA server which are on the same level as operations of WSDL are the focus of this thesis. The introduction of a new framework for semantic augmentation of these application-specific methods of an OPC UA server presents barriers for industry adoption. To overcome this problem, the well-known OWL-S framework and SAWSDL specification from the domain of semantic web services are incorporated to OPC UA methods. Subsequently, a hybrid approach which is a combination of OWL-S and SAWSDL specifications is proposed. All three proposed semantic approaches enable market-like demand-supply synergy described in Section 2.2.2. These semantic approaches enable classical walrasian-like markets [Vega97] during production runtime where auctions are placed centrally so that agents make local decisions [Clea96], and thus consolidate global declarative and local imperative knowledge. Such frameworks also make sure the emergent behavior due to local interactions conforms to global convergence. Subsequently, the autonomous discovery of manufacturing services of abstract PO leads to dynamic orchestration of the manufacturing processes using OWL-S constructs or BPMN tools.

4.1 Assimilation of OWL-S Framework in OPC UA

Based on the principle that semantics should be separated from the underlying details, the OWL-S approach follows a top-down design where service semantics is described independent of actual realization. It is only later grounded to concrete service descriptions such as WSDL.

The OWL-S specification is built on top of OWL. It consists of the main ontology called service that has three sub-ontologies, i.e., the profile, the process model and the grounding. Apart from semantic service description and software interface for service invocation, the OWL-S specification also allows to specify the preconditions and results of a process execution to model the initial and subsequent states of the world, respectively. The preconditions and results of a method invocation can be set on the states of the manufacturing resources and products that are part of the common semantic model. The OWL-S constructs such as if-then-else, while-until, sequence, split and split-join take into account the various outcomes of a process. This facilitates an adaptive orchestration plan that considers the states of the world before and after the execution of the process. Although the OWL-S ontology was conceived for web services, the framework is applicable to describe services in general. This thesis incorporates the OWL-S standard into OPC UA application specific methods.

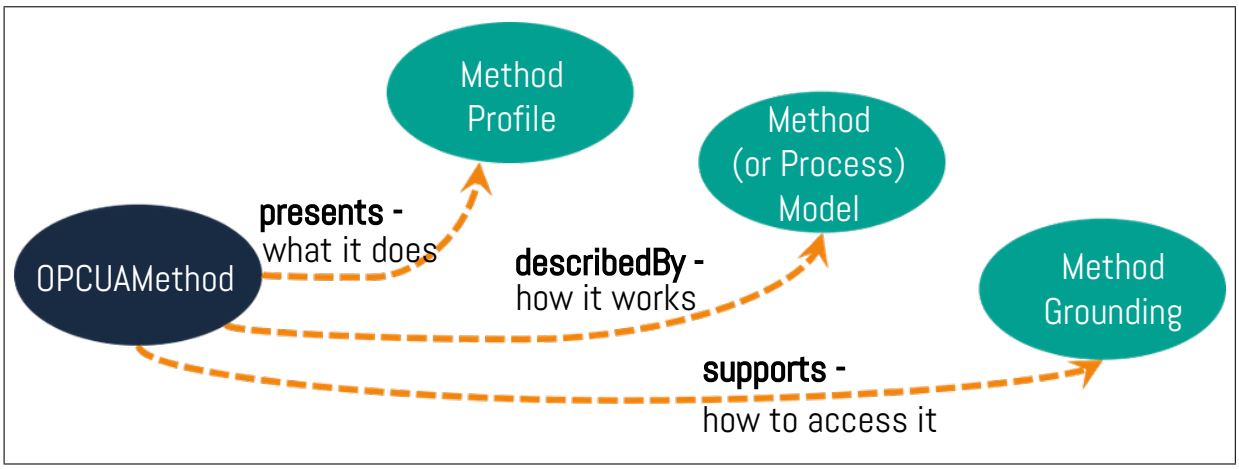

Figure 4.2: OPC UA Compliant OWL-S Ontology [KATT18d].

This approach is named *SemOPC UA*. The details are described with regards to a concrete example.

4.1.1 Modified OWL-S Sub-Ontologies

The main *OPCUAMethod* class and its sub-ontologies, namely, *MethodProfile*, *MethodModel* and *MethodGrounding* provide the means of organizing the description parts of a web service. Sometimes, method model and process model are used interchangeably, as both terms signify the same meaning. These ontological concepts are partially relevant with reference to the OPC UA application methods. Therefore, these ontological components must be enhanced/modified to incorporate the concepts related to OPC UA. The remodeling of the OWL-S ontologies also involved adapting the terminology to make it relevant and compatible with OPC UA. Figure 4.2 illustrates the renamed top ontologies of OWL-S. The modified ontologies are imported into the method metadata file (see Listing 4.1).

For the purpose of brevity, this thesis only refers to the important object and data properties of the sub-ontologies relevant to method discovery and invocation. The current section describes the modified sub-ontologies that define the metadata of an OPC UA application specific method, and a concrete example is provided with regards to an imaginary *welding* method example.

methodProfile Ontology The profile class is *presentedBy* the OPC UA method to provide the introductory information such as method name and textual description. It also provides a functional description regarding information transformation and changes to the state of the world when the method is executed by specifying the input, output parameters, and preconditions and post conditions respectively. The result is the combination of method output and the changes to the domain caused by method execution. In order to avoid confusion, the thesis classifies the state of the world, with the exception of output parameters, post

```
xmlns:methodProfile="http://www.rescom.org/owls/methodProfile#"
xmlns:processModel="http://www.rescom.org/owls/processModel#"
xmlns:methodGrounding="http://www.rescom.org/owls/methodGrounding#"
xmlns:Node="http://www.rescom.org/owls/OPCUANode#"
xmlns:WeldParams="http://www.rescom.org/owls/WeldParams#"
xmlns:Expr="http://www.rescom.org/owls/Expression#"
xml:base="http://www.example.org/opcuamethod.owl">
<owl:Ontology rdf:about="http://www.rescom.org/ABox_Owls">
<owl:imports rdf:resource="file:/C:/workspace/MethodGrounding.owl"
    />
<owl:imports rdf:resource="file:/C:/workspace/MethodProfile.owl"/>
<owl:imports rdf:resource="file:/C:/workspace/Node.owl"/>
<owl:imports rdf:resource="file:/C:/workspace/WeldParams.owl"/>
<owl:imports rdf:resource="file:/C:/workspace/Expression.owl"/>
</owl:Ontology>
<method:Method rdf:ID="WeldingMethod">
<method:presents>
<profile:Profile rdf:ID="WeldingProfile"/>
</method:presents>
<method:describedBy>
<process:AtomicProcess rdf:ID="WeldingModel"/>
</method:describedBy>
<method:supports>
<grounding:Grounding rdf:ID="WeldingGrounding"/>
</method:supports>
</method:Method>
```

Listing 4.1: Imported Modified Ontologies.

invocation of an OPC UA method into *post-conditions*, as against the usage of *results* and *effects* terminology of original OWL-S ontology framework. The profile class only introduces the method, it provides no description. Hence, it is recommended that the method parameters in the profile class should point to the process model class that points to the concrete ontological concepts.

The functionality of a web service in the IT world is determined by its algorithm that transforms the data in terms of input and output. In manufacturing, the functionality of a machine method is its ability to transform the workpiece from one state to another. In this context, the method functionality cannot be represented by the in-out parameters alone, and hence, a task-oriented description which supplies additional information is also needed. To that end, the thesis recommends to link the concrete manufacturing service provided by a manufacturing resource to the method category class. Concrete method selection parameters such as execution time and manufacturing cost are also introduced as data properties and

```
<profile:Profile rdf:about="#WeldingProfile">
<method:presentedBy rdf:resource="#WeldingMethod"/>
<profile:OPCUAMethodName>Welding</profile:OPCUAMethodName>
<profile:textDescription>Welding task</profile:textDescription>
<profile:hasInput rdf:resource="#X-CoOrdinate"/>
<profile:hasInput rdf:resource="#Y-CoOrdinate"/>
<profile:hasOutput rdf:resource="#Acknowledgement"/>
<profile:hasPrecondition rdf:resource="Expr#StatusStart"/>
<profile:hasResult rdf:resource="Expr#StatusWelded"/>
<profile:methodCategory rdf:resource="WeldParams#Welding"/>
<profile:ExecutionTime rdf:resource="abc"/>
<profile:ExecutionCost rdf:resource="xyz"/>
</profile:Profile>
```

Listing 4.2: OWL-S Profile concept applied to OPC UA Method.

linked to the method category class in order to make the method discovery relevant to manufacturing services.

The method profile is utilized by the client side agent during the method discovery stage. The discovery is successful if the method invoker satisfies the constraints put forward by the server method, and the server method returns the arguments expected by the client. In other words, if $Client_{\text{inArgs}}$ is the list of input arguments of a client request template and $Server_{\text{inArgs}}$ is the list of input arguments of a server side method, then the discovery function $D(x, y)$ is successful if following condition holds true: $\forall x \in Server_{\text{inArgs}}$, $\exists y \in Client_{\text{inArgs}}$, $D(x, y)$. In the same vein, if $Client_{\text{outArgs}}$ is the list of output arguments of a client request template and $Server_{\text{outArgs}}$ is the list of output arguments of a server side method, in order to match the following relation should hold true: $\forall x \in Client_{\text{outArgs}}$, $\exists y \in Server_{\text{outArgs}}$, $D(x, y)$. In order to find a suitable method, this relation should hold true over preconditions, postconditions and method category properties. Various semantic matching algorithms can be applied on the attributes of the method profile ontology in order to find a degree of match.

Although backward compatible, the deprecated properties and ontological elements of the OWL 1.0 and 1.1 release profile class are not discussed. The changes to the method Profile ontology are only cosmetic. The object property *serviceParameter* which is used as a construct to define a supplementary property, and a few data properties, namely, *serviceProduct*, *taxonomy*, *code* and *value* are omitted on the grounds of simplicity and inapplicability. Listing 4.2 provides an example of the profile class of an OPC UA method offered by the welding machine in terms of method name, category, input and output parameters, and preconditions and result.

methodModel Ontology The method model ontology describes the method specification with regards to transformation of data: in terms of input and output parameters, and transformation of the world: in terms of pre- and post-conditions. Though the method profile plays an active role during initial method discovery by the client agent, and the method model ontology is involved during method invocation, they are essentially two different representations of the same OPC UA method. Naturally, the input, output, pre- and post-conditions appear in both these ontologies. After a preliminary investigation of the method profile, if the method is found suitable for the client's requirements, the method discovery agent is redirected to the method model instance ontology using the *hasProcess* object property. From this point on, the method model controls the interaction with the method [Mart04]. The method model can have any number of inputs (including zero), and can produce any number of outputs after successful method invocation. However, the pre-condition(s) of a specific method should hold true to invoke it, and the successful invocation produces a *change in the world*, called post-condition of the method. Through these attributes, the method model ontology provides a declarative specification on the objectives of the method and guidance to the clients for interaction with the server. In a nutshell, the profile ontology of the *welding* method describes the service provided by the method, and the method model ontology declares that in order to invoke the method the client should pass the welding co-ordinates on the work piece to the server.

As OWL-S does not have a construct for grouping several processes, each method model corresponds to a single OPC UA method. Therefore, the OPC UA server is required to generate a separate OWL-S method metadata file for each of its methods. However, composite processes which are hierarchically defined workflows made up of atomic, simple and other composite processes can be constructed using a number of control flow operators.

There is no syntactic differentiation between a method in OPC UA and an operation of a web service. Hence, the original service model ontology can be used *as-is* to represent an OPC UA method signature. The parameter, which is a subclass of a variable defined in Semantic Web Rule Language (SWRL), and parameterType are renamed to *Argument* and *ArgumentType* respectively in order to reflect the terminology relevant to arguments of the OPC UA specification.

The pre- and post-conditions of the OPC UA method can be represented as SWRL rules. Thus, the functional properties of an application specific method are evenly distributed in both the method profile and method model sub-ontologies. However, in order to advertise the same method to different perspectives of clients, the same method model ontology can be tied to different method profile ontologies. For example, the method representing the manufacturing service provided by a versatile milling machine can have different method profile ontologies corresponding to its *drilling*, *boring*, *gear-cutting*, and *slots production* profiles.

```
<process:AtomicProcess rdf:about="#WeldingProcess">
<method:describes rdf:resource="#WeldingMethod"/>
<process:hasInput><process:Input rdf:ID="X-CoOrdinate">
<process:ArgumentType rdf:resource="WeldParams#X-CoOrdinate"/>
</process:Input></process:hasInput>
<process:hasInput><process:Input rdf:ID="Y-CoOrdinate">
<process:ArgumentType rdf:resource="WeldParams#Y-CoOrdinate"/>
</process:Input></process:hasInput>
<process:hasOutput><process:Output rdf:ID="Acknowledgement">
<process:ArgumentType rdf:resource="WeldParams#ACK"/>
</process:Output></process:hasOutput>
<process:hasPreCondition><process:PreCondition rdf:ID="StatusStart">
<process:Condition rdf:resource="Expr#StatusStart"/>
</process:PreCondition></process:hasPreCondition>
<process:hasPostCondition><process:PostCondition rdf:ID="StatusEnd">
<process:Condition rdf:resource="Expr#StatusWelded"/>
</process:PostCondition></process:hasPostCondition>
</process:AtomicProcess>
```

Listing 4.3: OWL-S Process Model Sub-Ontology Adapted to OPC UA Method.

Listing 4.3 illustrates the method model of the welding method where the arguments, pre- and post-conditions refer to concrete ontological concepts.

methodGrounding Ontology While the profile and process model ontologies describe the capabilities of the method, the grounding ontology specifies the details of method invocation. It maps the abstract method specification to the concrete implementation details. The original OWL-S grounding ontology was modeled for facilitating the execution of web service operation. The ontology exemplified to pragmatically connect the ontology to the prevalent web service standard of WSDL. Therefore, the method grounding sub-ontology has to undergo major changes as the mode of method invocation is inherently different in OPC UA compared to a web service.

```
<grounding:Grounding rdf:about="#WeldingGrounding">
  <method:supportedBy rdf:resource="#WeldingMethod"/>
  <grounding:hasAtomicProcessGrounding>
    <grounding:AtomicProcessGrounding rdf:ID="
    WeldingAtomicProcessGrounding"/>
  </grounding:hasAtomicProcessGrounding>
</grounding:Grounding>
<grounding:AtomicProcessGrounding rdf:about="#
    WeldingAtomicProcessGrounding">
  <grounding:owlsProcess rdf:resource="#WeldingProcess"/>
  <grounding:EndpointURL rdf:datatype="http://www.w3.org/2001/
```

```
        XMLSchema#anyURI">
10    opc.tcp://localhost:58711/PCoUaServer</grounding:EndpointURL>
11    <process:methodInfo>
12    <grounding:MethodBrowseInfo rdf:ID="N1">
13    </grounding:MethodBrowseInfo>
14    </process:methodInfo>
15    <process:hasInputArgument><grounding:Argument rdf:resource="#X-
        CoOrdinate">
16    <Node:NodeID>ns=2;s=50bcabac-623b-43ea-8f69-17b12d533166-1</Node:
        NodeID>
17    </grounding:Argument></process:hasInputArgument>
18    <process:hasInputArgument><grounding:Argument rdf:resource="#Y-
        CoOrdinate">
19    <Node:NodeID>ns=2;s=50bcabac-623b-43ea-8f69-17b12d533166-2</Node:
        NodeID>
20    </grounding:Argument></process:hasInputArgument>
21    <process:hasOutputArgument><grounding:Argument rdf:resource="#
        Acknowledgement">
22    <Node:NodeID>ns=2;s=50bcabac-623b-43ea-8f69-17b12d533166-3</Node:
        NodeID>
23    </grounding:Argument></process:hasOutputArgument>
24  </grounding:AtomicProcessGrounding>
25  <grounding:MethodBrowseInfo rdf:ID="N1" rdf:resource="Node#Node">
26    <Node:ReferenceType>Organizes</Node:ReferenceType>
27    <Node:NodeID>ns=2;s=a72e725d-6be7-4a17-bcd4-0be67b6cbfbe</Node:
        NodeID>
28    <Node:BrowseDirection>Forward</Node:BrowseDirection>
29    <Node:NodeClassMask>Object|Method|Variable</Node:NodeClassMask>
30    <Node:ResultMask>All</Node:ResultMask>
31    <grounding:MethodBrowseInfo rdf:resource="#N2" />
32    </grounding:MethodBrowseInfo>
33    <grounding:MethodBrowseInfo rdf:ID="N2" rdf:resource="Node#Node"
        >
34    <Node:ReferenceType>HasComponent</Node:ReferenceType>
35    <Node:NodeID>ns=2;s=77786eba-f095-4996-9dcb-4013f3d7df04</Node:
        NodeID>
36    <Node:BrowseDirection>Forward</Node:BrowseDirection>
37    <Node:NodeClassMask>Object|Method|Variable</Node:NodeClassMask>
38    <Node:ResultMask>All</Node:ResultMask>
39  </grounding:MethodBrowseInfo>
```

Listing 4.4: OWL-S Grounding Sub-Ontology Adapted to OPC UA Method.

Lines 1-6 in Listing 4.4 declare the grounding for the welding atomic process defined in the

method model, whereas the definition of the atomic process grounding takes place in lines 7-39. Since the *wsdlDocument* data property of OWL-S is not applicable in this context, it has been replaced with the *EndpointURL* data property. An OPC UA client requires this endpoint URL to connect and subsequently create a session with the OPC UA server. The object property *MethodBrowseInfo* provides the root node of the OPC UA server. The nested object property, called *hasNodeReference*, provides the browse information from the default or the root node of the server to reach the method node.

The input parameters of the OPC UA *call* service which invokes the required application method are the ID of the method node, and the ID of the object that provides the method node along with input arguments. Although the *hasNodeReference* is essentially recursive, in effect, the two innermost nodes are adequate. In order to create a request instance to browse the references of a node, the *Node* has several data properties, namely, *NodeID*, *ReferenceType*, *BrowseDirection*, *NodeClassMask* and *resultMask*. The property has the predicate *ReferenceType* to represent the reference type of the node. The predicates *NodeID* and *BrowseDirection* represent the OPC UA node ID and the direction of the returned browsed nodes respectively. The predicate *NodeClassMask* expresses which node classes should be returned by the *browse* service, while the predicate *ResultMask* conveys the types of fields that the *browse* service should return. Furthermore, this modified OWL-S specification also provides the means to access the input and output argument nodes with the *Argument* predicate.

It is a strategic decision to always opt for the TCP-UA or UA native binary protocol since the performance is better in comparison to the WS-* based SOAP/HTTP(S) OPC UA implementation [Arch18]. Moreover, the OPC Foundation guidelines suggest the implementation of the UA native binary protocol as a mandatory procedure to set up an OPC UA server. Implementations of other communication protocol bindings are optional. Hence, the *OPC UA transport* description which includes protocol, message formats, serialization, transport and addressing is not part of this method metadata OWL-S file. The complete modified OWL-S ontology is illustrated in Figure 4.3.

4.1.2 Working Principle of OWL-S Augmented OPC UA

The manufacturing resource providing an OPC UA application specific method transmits the *method profile*, *method model* and *method grounding* ontologies either in different OWL-S files or in a single OWL-S file to the GeSCo GMDR with regards to its manufacturing service capability. These OWL-S files are centrally accessible to all the entities of the production as a static ontology service. During production runtime, the decentralization facilitator component of GeSCo parses these metadata files from its GMDR with the aid of common semantic

model, and matches the required manufacturing services of PO to the capabilities offered by resources on the shop-floor. With the OWL-S concepts, it is possible to generate a flexible orchestration plan which can be changed in case of unforeseen events in the production. It also facilitates the implementation of loosely coupled production systems, and thereby provides plug and produce, and smooth reconfigurability features.

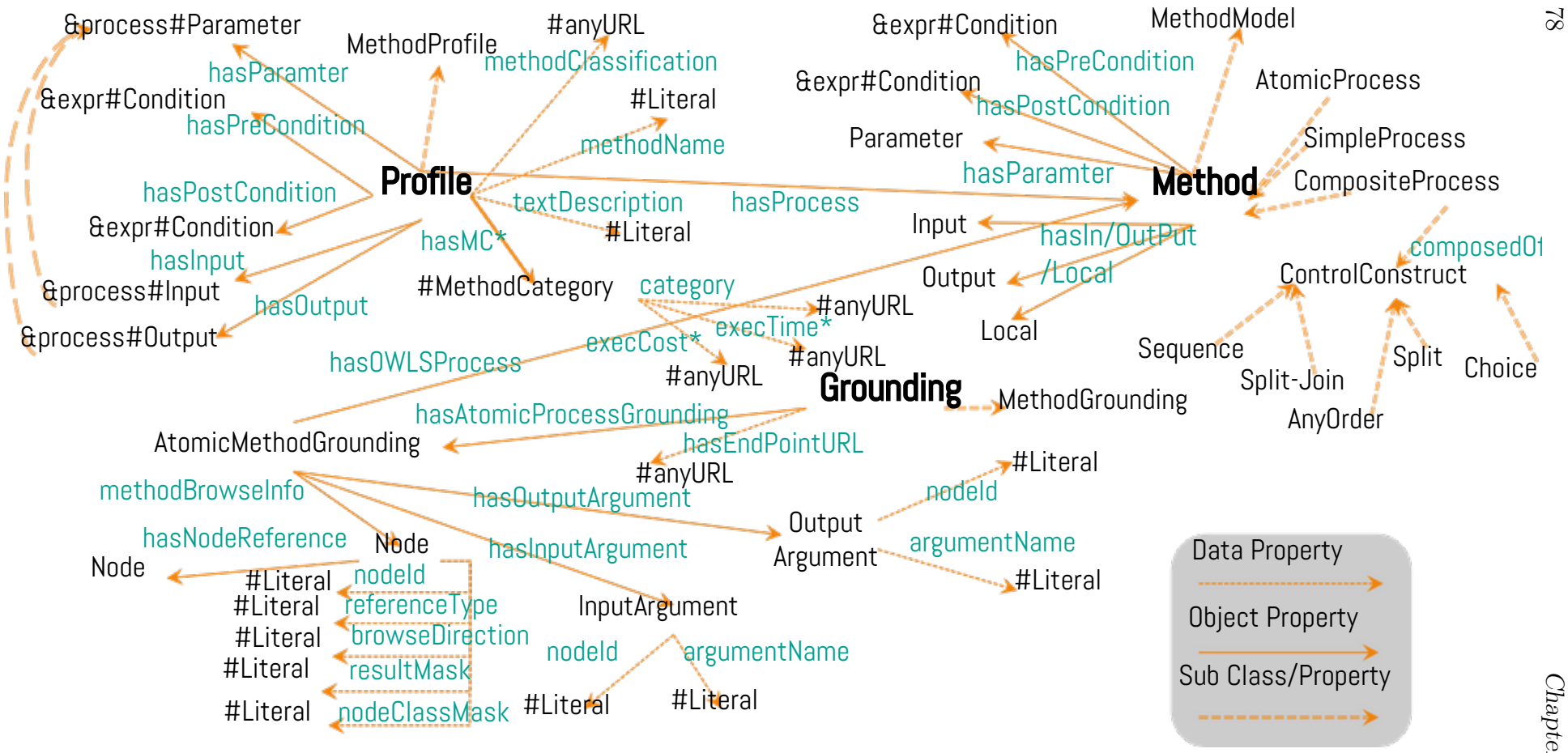

Figure 4.3: Complete OWL-S Ontology with regard to OPC UA Application Method.

4.2 Assimilation of SAWSDL Specification in OPC UA

The previous Section 4.1 proposed SemOPC UA by incorporating prominent OWL-S concepts to OPC UA application specific methods in the manufacturing domain to enable factory automation. This section continues the pursuit of enriching the OPC UA with semantics, and to that end, extends the idea of the industry-neutral SAWSDL specification that does not enforce the usage of a particular ontology modeling language, to OPC UA server side application specific methods. This approach is named as SA-OPC UA. The intentional simplicity and easy implementation of SAWSDL concepts are the factors that lower the entrance barrier by smoothening the learning curve for adopters of formal semantics in OPC UA.

The following subsections lay the groundwork by drawing the parallels between WSDL elements and OPC UA information model elements with regards to OPC UA application methods. With this analogy, the research work later extends the annotation concepts of SAWSDL to OPC UA applications methods.

WSDL and SAWSDL Concepts

The Web Service Description Language (WSDL) is a W3C recommendation which provides a formal and machine readable description of SOAP based web services. WSDL provides information with respect to the functionality offered by the web service. It provides both an abstract and concrete aspect of a web service by describing the messages exchanged during a web service operation, and binding specific information such as transport and wire format details for one or more interfaces. However, the web service description using WSDL is merely of syntactic nature. It neither expresses the goals of the web service nor the meanings of its input and output arguments. As a consequence, there is a semantic gap between the syntactic description of web service and its underlying meaning. Semantic Annotations for WSDL (SAWSDL), a W3C recommendation, is an incremental bottom-up approach of modeling Semantic Web Services (SWS). It is a mechanism where elements of WSDL are decorated with extensible attributes to attach semantic annotations. These semantically annotated WSDL elements consist of references to concepts in a semantic model, for example classes of an OWL ontology. SAWSDL is ontology modeling language agnostic, i.e., it does not command the usage of a specific ontology modeling language. This choice is left to the end-user. However, from the perspective of SAWSDL, references of semantic descriptions must be Unique Resource Identifiers (URI) that must point to semantic concepts defined either within or outside the scope of WSDL.

The SAWSDL specification defines two types of extension attributes on the web service concepts, namely modelReference and schemaMapping. The schemaMapping attributes are

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

Listing 4.5: Example of SAWSDL Annotations on a Web Service Operation and its Parameters.

further classified into liftingSchemaMapping and loweringSchemaMapping attributes. Listing 4.5 illustrates an example usage of these attributes on a web service operation and its parameters. The modelReference that is set to an URI indicates the correlation between the WSDL or XML schema component, and a concept of the semantic model. This attribute is annotated over WSDL interfaces, web service operations and its messages, XML schema type definitions and element declarations. WSDL fault element(s) which declare and describe the runtime exceptions that may occur during an operation execution can also be semantically annotated to provide a high level semantic description of the fault. The model references are employed to arbitrate between the client requirements and the service capabilities. After the match is found, incongruities might arise between the parameters of the client and the server during the operation invocation. A mechanism is required to translate the semantic model into a request message on the client side to match the expected format on the server side. Another requirement is the transformation of a service output represented in a suitable data model to a corresponding semantic model. The former technique where semi-structured XML data is translated to semantic data is known as lowering transformation and the latter is known as lifting transformation. In general, these are extension attributes applicable in case the structure of an instance data does not have a direct relationship with the organization of the semantic data. The SAWSDL specification recommends that these extension attributes point to the transformation documents such as XSL (eXtensible Stylesheet Language) transformation (XSLT) that understand and resolve the mismatches in the event the web service request structure fed by the client does not satisfy the server expectation (refer [Fens11a] for details).

4.2.1 Organization of Application-Specific Method Nodes in OPC UA Server Address Space

This section proposes to classify the application specific methods pertaining to a OPC UA server of a manufacturing resource in order to organize them in a defined structure instead of scattering across the server. This structure also streamlines the browsing of the OPC UA

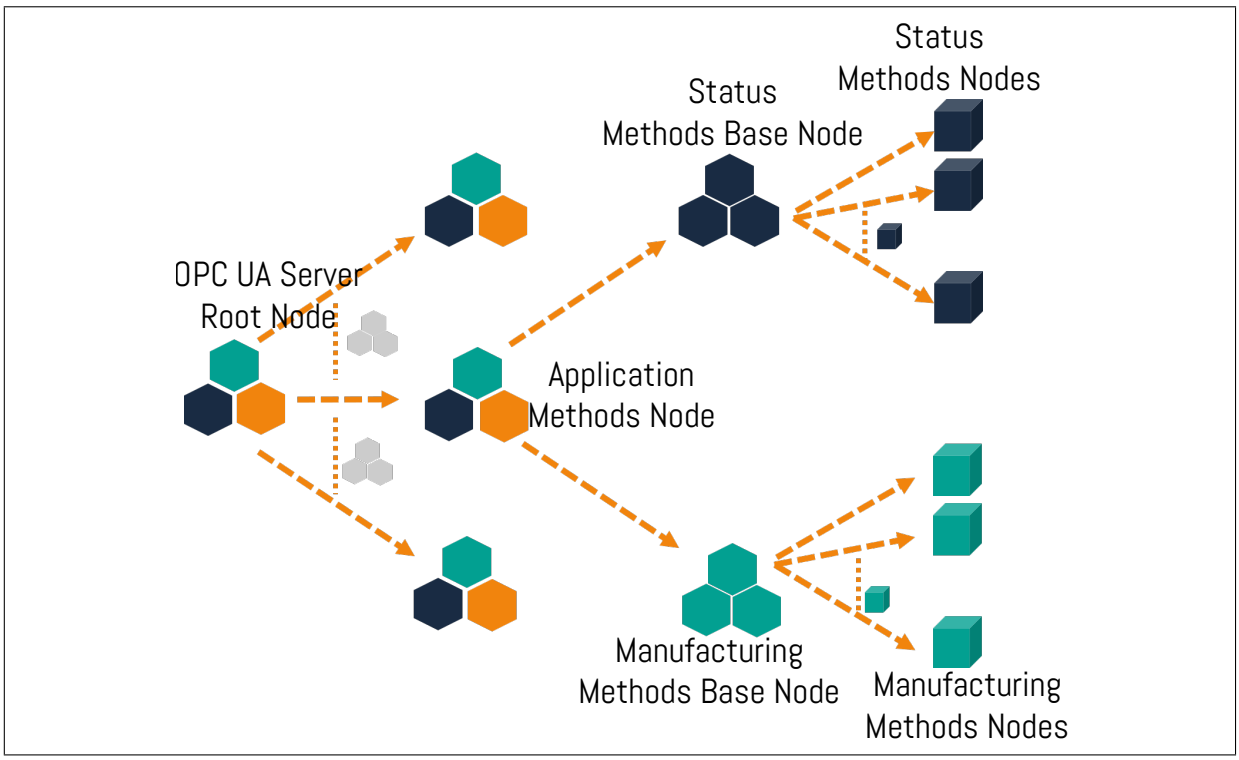

Figure 4.4: Organization of Application Specific Methods in OPC UA Server [KATT18a].

server during method discovery. The application specific methods of the OPC UA server should be classified into two categorizations: status methods and manufacturing methods. The status methods of a manufacturing resource provide information about various physical aspects of the manufacturing resource, status of the assigned PO and miscellaneous data like PO queue at its work station. Manufacturing methods when invoked should deliver the corresponding manufacturing service. Based on the business tasks they perform, each of the methods belonging to a particular categorization should be grouped under a common parent node. The thesis names these two parent nodes as *status methods base node* and *manufacturing methods base node* respectively. These two base nodes should further be grouped under a single node, called *Application Methods Node*, as illustrated in Figure 4.4. In addition to the provision of streamlined method browsing for human users, this organization of methods also plays a role in method discovery by software agents.

4.2.2 Exploitation of OPC UA Node structure

The WSDL is the underlying enabling technology of SAWSDL. SAWSDL enables semantic annotations using extension attributes of WSDL on WSDL elements such as interface, operation, messages, fault and binding. Unlike WSDL, which describes the syntax of the web services and its operations, there exists no such equivalent facility in OPC UA to describe the syntactical behavior of the OPC UA server. The UA node set file which describes the information model of the OPC UA server is a distant match to WSDL of web services, how-

Table 4.1: Drawing Parallel between WSDL Attributes and OPC UA Application Specific Method Nodes

WSDL Elements	OPC UA Types related to Server Methods
Interface (or PortType)	Status or Manufacturing Methods Base Node
Operation	Status or Manufacturing Method Node
Message (Input)	Argument []
Message (Output)	Argument []
Input Message Part	Argument
Output Message Part	Argument
Fault	StatusCode

ever, it contains neither grounding information, nor is it extensible. In addition, there is also no provision to add extension elements to methods, input and output parameters in the OPC UA address space.

As described in Section 4.2.1, an OPC UA server is a collection of clusters of application methods segregated by type of functionality provided by them. This logical segregation is equivalent to implementations of different interfaces in web services to achieve the design principle of separation of concerns. Moreover, an application method in OPC UA has the following components, namely, Method Node class, Input and Output Arguments and Service Result Node class. It is necessary to make the distinction between the WSDL elements and the corresponding OPC UA terms to correlate the SAWSDL concepts to the OPC UA specification. Based on the organization of application-specific method nodes described in Subsection 4.2.1, Table 4.1 provides equivalent mappings between WSDL elements and OPC UA application method specific elements.

All the OPC UA types listed in Table 4.1 inherit from the base Node class which has an

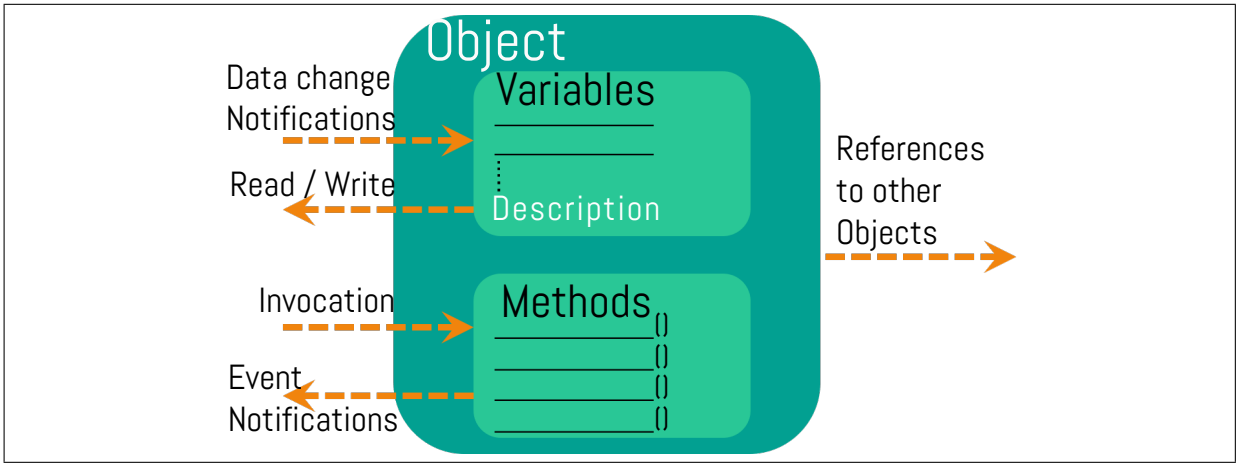

Figure 4.5: SAWSDL Style Semantic Annotation in the *Description* Field of OPC UA Object Model.

optional attribute called *Description* that is reserved for localized textual description of the node (see Figure 4.5). Since this attribute is optional, the effective usage of the field is entirely at the discretion of the OPC UA server programmer. Hence, this field is subjected to arbitrary usage. The author proposes that this free flow textual field should semantically describe the node when annotated with the ontological concept. As such, this attribute serves as a hook-point for attaching the shared knowledge base. As all the concepts that are essential to describe a method have the provision for semantic annotation, the concepts of SAWSDL can be smoothly extended to OPC UA application specific methods.

The capabilities of a manufacturing resource can be divided into two granularities: server level and method level. From a client perspective, server level capability discovery is sufficient when a manufacturing resource provides a single dedicated functionality. However, in case a resource offers multiple manufacturing services, the corresponding OPC UA server exposes multiple stand-alone methods in its address space. In this context, a client is usually interested in only one or two of the methods rather than all the methods offered by the server, and instead of server discovery, the semantic client might need to do the finer grained method discovery. The annotation on the server method using model reference pointers is intended only to provide a rough approximation of method capability which is used as an initial indication to gauge the suitability for a particular requirement, and thereby filter out large number of incompatible methods. Further, the annotations on the inputs and output arguments of the method convey additional behavioral aspects and thus, assist zeroing in on the required OPC UA method.

During the discovery phase of a method, the OPC UA client searches for a suitable server method that accomplishes a specific set of goals. In the broader perspective, the qualified methods participate in method composition which is subjected to orchestration to complete the PO. However, the method capability along with the input and output compatibility alone are not the yardstick to gauge the suitability of the method to fit into method composition. The pre- and post-conditions represent the current and the subsequent state of the method execution context such as resource status, product state et cetera. At runtime, the pre-conditions of a method should hold true to invoke it. The output parameters plus post-conditions together express the outcome of the method. A formal specification of the method outcome reveals whether a method accomplishes the manufacturing objectives of the OPC UA client. During the production orchestration which is essentially an execution of methods in succession, these conditions act as bridges between the methods. To that end, SAWSDL is subjected to a general criticism that it is incapable of specifying the pre- and post-conditions of a web service operation as it only provides a single place holder to attach semantics at the operation attribute of WSDL (for example, refer [Ploc11, Losk13]). The OPC UA information model also suffers such a drawback as each node only has one Description field for semantic annotation. Since the concepts of pre- and post-conditions of OPC UA

```
<owl:Class rdf:ID="OPCUAMethod">
<rdfs:subClassOf><owl:Restriction>
<owl:onProperty rdf:resource="#manufacServiceCategoryProperty"/>
<owl:qualifiedCardinality rdf:datatype="xsd#nonNegativeInteger">1
</owl:qualifiedCardinality><owl:onClass rdf:resource="#
    ManufacServiceCategory"/>
</owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
<owl:onProperty rdf:resource="#manufacServiceProperty"/>
<owl:qualifiedCardinality rdf:datatype="xsd#nonNegativeInteger">1
</owl:qualifiedCardinality><owl:onClass rdf:resource="#ManufacService
    "/>
</owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
<owl:onProperty rdf:resource="#postConditionObjectProperty"/>
<owl:qualifiedCardinality rdf:datatype="xsd#nonNegativeInteger">1
</owl:qualifiedCardinality><owl:onClass rdf:resource="#
    MethodPostCondition"/>
</owl:Restriction></rdfs:subClassOf>
<rdfs:subClassOf><owl:Restriction>
<owl:onProperty rdf:resource="#preConditionObjectProperty"/>
<owl:qualifiedCardinality rdf:datatype="xsd#nonNegativeInteger">1
</owl:qualifiedCardinality><owl:onClass rdf:resource="#
    MethodPreCondition"/>
</owl:Restriction></rdfs:subClassOf></owl:Class>

<owl:ObjectProperty rdf:ID="manufacServiceProperty">
<rdfs:domain rdf:resource="#OPCUAMethod"/>
<rdfs:range rdf:resource="#ManufacService"/></owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="manufacServiceCategoryProperty">
<rdfs:domain rdf:resource="#OPCUAMethod"/>
<rdfs:range rdf:resource="#ManufacServiceCategory"/></owl:
    ObjectProperty>
<owl:ObjectProperty rdf:ID="postConditionObjectProperty">
<rdfs:domain rdf:resource="#OPCUAMethod"/>
<rdfs:range rdf:resource="#MethodPostCondition"/></owl:ObjectProperty
    >
<owl:ObjectProperty rdf:ID="preConditionObjectProperty">
<rdfs:domain rdf:resource="#OPCUAMethod"/>
<rdfs:range rdf:resource="#MethodPreCondition"/></owl:ObjectProperty>
```

Listing 4.6: Additional Object Properties Definition in OPC UA Method.

method execution, which are also deciding factors to check whether a given method meets the manufacturing requirement, cannot be modeled with such a simple approach, the method composition is not possible. Consequently, at manufacturing runtime the orchestration of the production process that relies on method composition is also not possible. Based on this reasoning, SAWSDL augmented OPC UA server only allows to discover the individual methods that serve the PO manufacturing services by dynamic discovery of capabilities of registered manufacturing resources in the shop-floor.

In order to overcome this problem, this thesis proposes the following: the ontological concept that is attached to the OPC UA application method should describe not only its capability but also pre- and post-conditions. In essence, any additional behavioral constraints like method category can also be associated with the nodes of method, and the concept can be

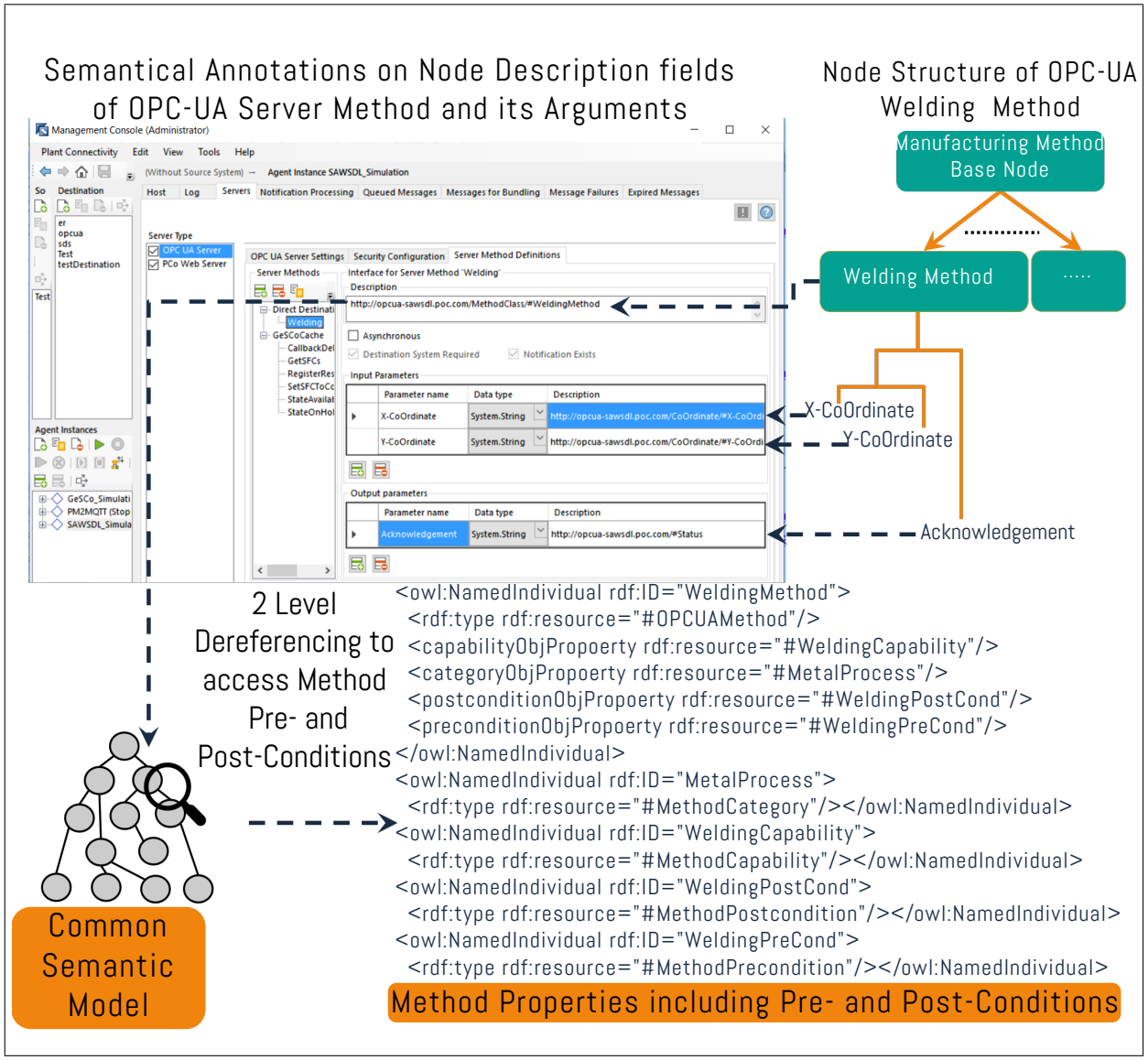

Figure 4.6: Semantic Annotations on Description Fields Welding Method and its Arguments from Running Use-Case Example of SA-OPC UA Server [Katt18a].

extended to even other nodes such as input and arguments of method. This is illustrated in Listing 4.6 where *OPCUAMethod* concept defines additional object properties, namely *preconditionObjectProperty* and *postconditionObjectProperty*.

From our running use case, Figure 4.6 shows an example of semantic concepts associated with the welding method which is an instance of *OPCUAMethod*. During method discovery, the OPC UA client runs a query on this ontological concept hooked to the method against the central ontological repository named common semantic model, to find the additional object and data properties of the method. This simple solution overcomes the problems of both the non-mandate of SAWSDL to define pre- and post-conditions constructs, and the limitation of the OPC UA method node to attach additional semantic attributes.

4.2.3 Assumptions, conventions and Scope Definition

Taking into account the constraints of simplicity, scope of thesis research work and annotation limitations on the OPC UA object model, the author makes the following assumptions and conventions:

1. OPC UA has a finite number of numeric codes that describe the result of the service or operation known as Status Code that is described in part 4 of the OPC UA specifications. It is a 32 bit number whose higher-word represents the error or the condition, and the lower-word represents the additional flags that shed light on the meaning of the status code. This limited sample space of status codes allows the OPC UA server to do away with the annotation on the status code of the application specific method. However, the status code can be used freely to model pre- and post-conditions of methods.
2. The evaluation set up and the thesis problem statement revolve around intra-enterprise manufacturing where access and communication with the different OPC UA servers of manufacturing resources on a technical level is presumed to be uniform. It is a strategic decision to always opt for the signed and encrypted UA native binary protocol since the performance is better in comparison to the WS-* based SOAP/HTTP(S) OPC UA implementation. Security is not the research focus of this thesis. Therefore, the corresponding grounding details such as security mode, security policy and encoding algorithms of the OPC UA protocol are not discussed as part of this thesis.
3. Though, in theory, any conceivable XML and RDF transformations (refer Subsection 4.2) can be programmed using sophisticated XPath queries in XSLT which is a Turing-complete language, an XPath expression applied to a particular RDF representation might not work for another equivalent representation (refer [Fens11a, Akht08] for details). The hybrid solutions such as XSPARQL [Akht08] which is a combination

of XQuery and SPARQL might be an alternative. However, no XML data exchange takes place during and between method invocation(s) as research is carried out on the premise that UA native binary is the protocol of choice. Hence, schema mapping transformations are not relevant for the research use-case.

4. The OPC UA node has only one attribute, namely the *Description* field to associate the semantic relationship in OPC UA. Therefore, a convention needs to be arrived at in order to accommodate both model reference and schema mapping attributes of SAWSDL. To that end, the *Description* field has no length constraints. Therefore, this thesis recommends to employ a double semicolon as the delimiter to differentiate model reference, lowering schema mapping and lifting schema mapping extension attributes. The schema mapping extension attributes are pertinent only in case of WS-* based SOAP/HTTP(S) OPC UA server which deals with XML request and response messages.

5. The model reference attribute can also have multiple URIs to accommodate the diverse heterogeneous ontologies represented in different semantic languages. When an OPC UA method related node is annotated with multiple URIs, each of the URI concepts is applicable to the node. However, there need not be a logical relationship between them. The multiple URIs pointing to the same model reference are separated by white spaces. This approach of annotation/extension of the nodes is non-intrusive in a way that does not invalidate the objectives of the OPC UA specification nor the manner it was previously handled.

4.2.4 Publishing OPC UA Methods Grounding Metadata

The OPC UA client accesses and browses the OPC UA server in order to find the offered application specific methods. These application methods can potentially be many and it is a tedious job to discover the required method either for a human browser or an automated OPC UA client. Therefore, different application specific method characterizations such as status methods and manufacturing service methods should be attached to the respective base node classes (see Figure 4.4), to create a logical segregation. The client, then, can unambiguously browse the specific base node depending on the current requirement.

Following the recommendations of SAWSDL, a semantically enriched WSDL document of a web service is published in the web service discovery registry which is accessible to the prospective web clients. However, the OPC UA server does not provide such a distributable and extensible metadata document for the benefit of the clients in order to discover the offered methods. To close this gap between SAWSDL and SA-OPC UA, the server should publish its endpoint URL along with the node IDs of the base node classes to the RPL component

```
<owl:NamedIndividual rdf:ID="SAOPCUA_ServerMetadata">
<rdf:type rdf:resource="#ServerMetaData"/>
<endpointURLDataPropety rdf:datatype="xsd#anyURI">
opc.tcp://WDFN32202381A:58710/PCoUaServer</endpointURLDataPropety>
<StatusMethodsBaseNodeDataPropety rdf:datatype="xsd#string">
e40222b8-b9ed-4054-9d76-df355cfafcf9</
    StatusMethodsBaseNodeDataPropety>
<ManufacturingMethodsBaseNodeDataPropety rdf:datatype="xsd#string">
94b2286d-5fd1-4126-b649-e7d32f67b23b</
    ManufacturingMethodsBaseNodeDataPropety>
</owl:NamedIndividual>
```

Listing 4.7: WSDL Equivalent Metadata published by SA-OPC UA Server of Manufacturing Resource to GeSCo [Katt18a].

of GeSCo. Listing 4.7 illustrates an example of such a metadata of an OPC UA server. However, there is a subtle difference between SAWSDL and SA-OPC UA conceptualizations. The abstract definitions of WSDL are annotated in the former, while concrete methods and parameters are annotated in the latter. The methods that belong to a particular methods base node (either status or manufacturing methods base node) in OPC UA are equivalent to the WSDL operations of the web server belonging to a particular port/interface type. In effect, the SA-OPC UA server publishes a compact interface that roughly translates to a semantically annotated WSDL with an aim of assisting the semantic clients in automated method discovery, composition and invocation. Another feature of this *grounding metadata* with regards to SemOPC UA method grounding discussed in Section 4.1.1 is that the method grounding of a SA-OPC UA server is applicable to all the methods of the server, while the SemOPC UA server defines method grounding for each of its application methods.

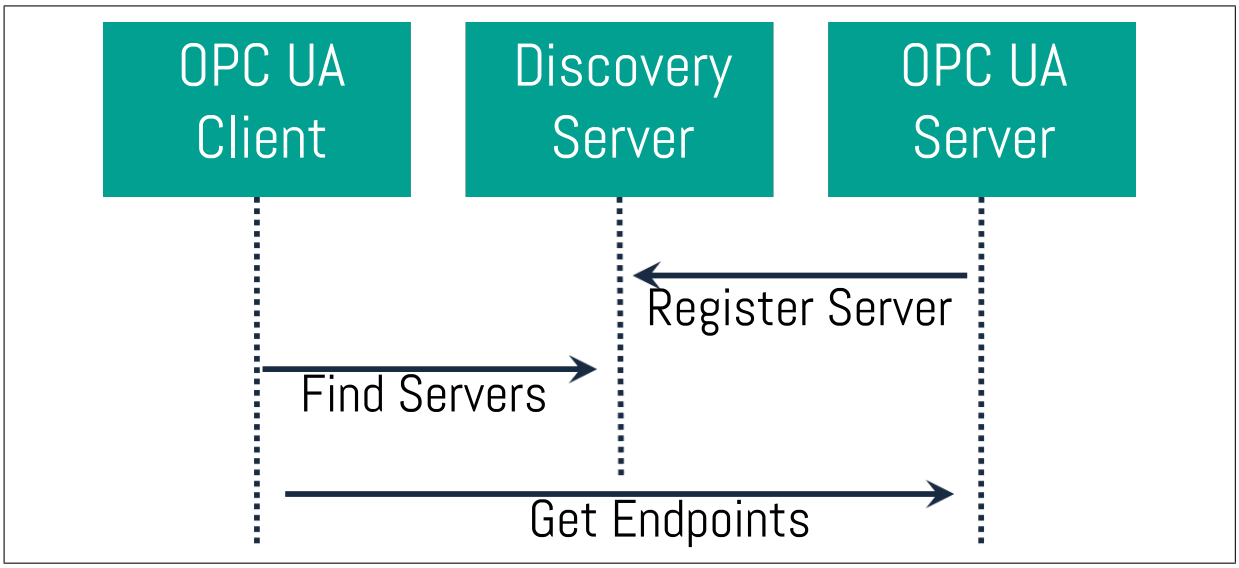

Figure 4.7: Use of Local Discovery Server [OPC 18b].

The client session that is required to browse the server is created using the standard practice of Local Discovery Server (LDS) where the OPC UA servers register themselves as prescribed in part 4 and 12 of the OPC UA specification. This procedure of OPC UA server discovery is illustrated in Figure 4.7. Therefore, the published endpoint URL is not required to create a client session with the server. Instead, this endpoint URL is merely a unique identifier of an OPC UA server to the client.

4.2.5 Working Principle of SAWSDL Augmented OPC UA

As described in [Katt18b], the manufacturing resource should either publish its capabilities meta data or permit its client to browse its internal structure to extract the metadata. The SemOPC UA server follows the former approach. The latter is the case in point of SA-OPC UA server. The sequence of events in method invocation and subsequent method composition in a SA-OPC UA server is shown in Figure 4.8. The first step is the encoding of the common semantic model which is relevant for both cloud based MES and GeSCo. This step involves the creation of formal expressions of the production participants and all relevant concepts. The contextual conditions are represented using formal rule based languages. The ensuing step is the annotation of application methods, and their in-out arguments of the SA-OPC UA server with concept expressions created in the previous step. Following publishing

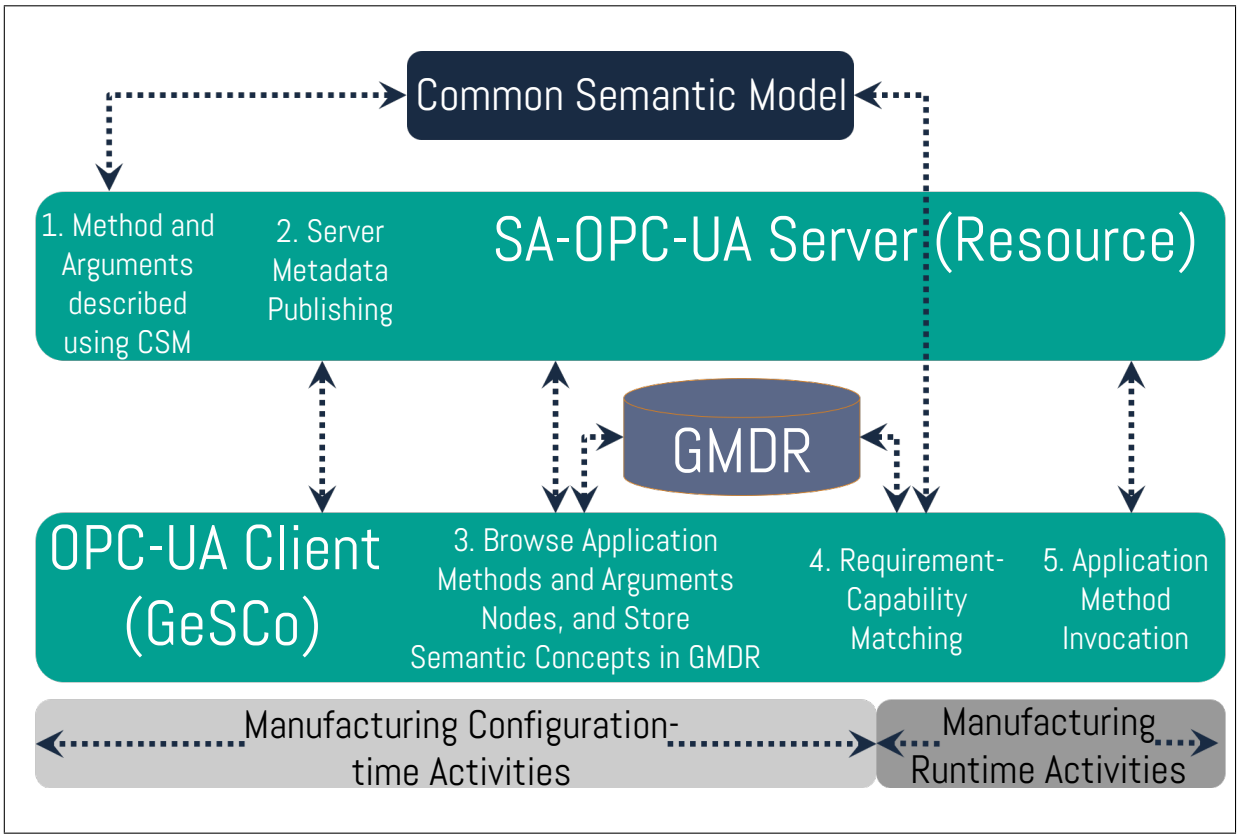

Figure 4.8: Sequence of Events in Dynamic SA-OPC UA Method Invocation [Katt18a].

the WSDL equivalent metadata by the SA-OPC UA compliant manufacturing resource to the GeSCo RPL component, a session is established with the SA-OPC UA server whose endpoint matches the published endpoint URL in the RPL component of GeSCo. Following this step, the OPC UA client browses both the manufacturing and status method nodes of the server, the corresponding nodes of input and output arguments of the methods, and dereferences (or lifts) the semantic concepts attached to the *Description* attributes of the nodes in order to discover their capabilities. It then stores the semantic concepts associated with the application methods in the MDR component of GeSCo. During production runtime, the abstract manufacturing service requirements of the PO are paired with the concrete manufacturing services provided by the resources on the shop-floor by dereferencing the concept expressions stored in MDR against the common semantic model. Naturally, this arrangement also leads to method composition due to provision for adding any number of additional constraints to the *method* concept expression (refer Listing 4.6).

4.3 Incorporation of a Hybrid of OWL-S Framework and SAWSDL Specification to OPC UA

Following the introduction of the prominent and the industry-neutral OWL-S and SAWSDL frameworks to the OPC UA specification, known as *SA-OPC UA* (refer Section 4.1) and *SemOPC UA* (refer Section 4.2) respectively, this section introduces a composition of these schemes to reap the advantages of both approaches to enable true factory automation. It also provides the rationale behind the conception of such a hybrid approach. The simplicity and easy implementation of the SAWSDL concepts, coupled with the general-purpose representation framework of OWL-S make this technique irreproachable and attractive to industry adoption.

Section 4.3.1 puts forward the arguments for and against both the OWL-S and the SAWSDL specifications with regards to software quality characteristics such as usability, efficiency and maintainability. Consequently, it draws the conclusion that a hybrid approach is more feasible. It derives the benefits and at the same time, precludes the shortcomings of both approaches. Section 4.3.2 describes the hybrid methodology.

4.3.1 Motivation for the Hybrid Approach

Analysis of SA-OPC UA

Following the guidelines of the SAWSDL specification, the independently modeled ontological concepts are attached to the OPC UA method and related nodes. Every node in the OPC

UA information model has a *Description* field which is the container for a semantic reference. The pre- and post-conditions of the method execution which are important parameters of the method orchestration are also encapsulated in the semantic reference of the method node. The server publishes its endpoint URL along with the manufacturing and status method nodes to the Resource Perception Layer (RPL) of GeSCo, which is consequently used at runtime for the purpose of intelligent discovery, composition and consumption of OPC UA methods (refer Section 4.2.1).

The simplicity and easy implementation of SAWSDL concepts are the factors that lower the entrance barrier, and make SA-OPC UA agreeable for industrial recognition. However, SAWSDL does not prescribe the usage of a particular framework to describe the ontologies, taxonomies or mappings to semantically annotate the WSDL interfaces, operations, and their input and output messages. This gives the web service programmer the freedom to choose the ontological language that best represents the capabilities of web services and also allows to integrate well with the other project modules. However, at the same time the arbitrary usage of the semantic annotations does not bode well in the production automation in the context of enabling standardized communication interfaces and protocols between every entity from the shop-floor to the IT solutions on the cloud. Furthermore, the semantic references attached to various OPC UA nodes can also be from diverse ontologies. In such a situation, the OPC UA base node, methods and its arguments nodes cannot be semantically correlated. In addition, even when the OPC UA base node, methods and argument nodes are semantically referenced homogeneously, SAWSDL does not impose restrictions on the logical relationship between them. In order to provide a perspective to the incorporation of generic SAWSDL concepts to the OPC UA information model, a general-purpose representation framework must be employed to semantically describe the application specific methods provided by the OPC UA server.

Analysis of SemOPC UA

With regard to the above argumentation of SAWSDL insufficiency, the widely accepted SWS framework : OWL-S fits the bill. In case of OWL-S, the semantics of services provided by these manufacturing resources are modeled within the boundaries of a defined ontological framework, and published to the generic method discovery repository component of GeSCo in order to automate the process of intelligent OPC UA method discovery, orchestration and execution by client software agents. However, the OWL-S on its own is not a good choice of semantic description framework of OPC UA application methods for the reasons explained in the following:

- The OWL-S publishes structured semantic data which is a combination of the capability

of the OPC UA method and the corresponding grounding information that provides the information for invocation of the method. The comprehensive data that is sent from the OPC UA server to GeSCo warrants an elaborated discovery repository.

- Modern manufacturing resources have rich service provision capabilities in that they are capable of delivering more than one functionality, and this further increases the complexities of the manufacturing planning and execution. On the other hand, a process model ontology in the OWL-S framework only describes one method offered by an OPC UA server. Therefore, the OPC UA server should publish an OWL-S ontology for each of the application methods corresponding to the services provided by the manufacturing resources. This amounts to storing huge amounts of metadata in GeSCo.
- The labyrinthine browse paths of the OPC UA server nodes, which form part of the OWL-S grounding ontology also add additional burdens on the production network. This also goes against the envisioned idea of nurturing a light-weight edge component. The OPC UA server browse path disclosure also necessitates a sophisticated grounding process.
- The publication of long browse paths is also not feasible in the event the OPC UA server is not capable of preserving the same address space on reboot.
- In a manufacturing setting where the entire machinery is connected to the network, the publishing of the method node browse path also poses a significant security risk.

An additional illustration is provided by Table 4.2 which lists the attributes with regard to both functional and non-functional properties and compares how the OWL-S framework and SAWSDL specification fare against these attributes.

Table 4.2: Empirical comparison of OWL-S framework and SAWSDL specification

Attributes	**OWL-S**		**SAWSDL**	
	Remarks	**Rating**	**Remarks**	**Rating**
Simplicity and Implementation	No, more effort	−	Yes, less effort	+
Industry Adoption	Relatively difficult to pursue	−	Smooth	+
Framework Prescription	Yes	+	No	−

Restriction of Ontology Language	Yes	±	No	±
Strict Guidelines for Programming	Yes	±	No	±
Standardized Communication Interfaces and Protocols	Supports standardization	+	Possibility of creation of ad-hoc semantics structures	−
Homogeneous Ontology	Yes	+	Multiple ontology languages can be used for the same server annotations	−
Arbitrary Usage of Ontology	No	+	Subjected to arbitrary usage	−
Volume of published Server Metadata	More data	−	Less data	+
	Entire OWL-S ontology is published for each of the server methods	−	Constant size metadata is sent on a per server basis	+
	Storing of huge amounts of data in clients	−	Storing of comparatively little amounts of data in clients	+
	More burden on production network	−	Less burden on production network	+
	If the server cannot preserve address space, the updated method grounding ontologies has to be published after every reboot	−	Client only stores ID of the application methods base node, and hence it only needs to get updated node ID from the rebooted server	±
	Security risk	−	Relatively less security risk	+

Built in Support for Orchestration	Yes	+	No	−
Sophisticated Discovery Mechanism	Yes	±	No	±
Matchmaking	Efficient	+	Convoluted mechanism due to absence of a framework	−
Support for OWL	Yes	+	Yes	+

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

4.3.2 Methodology

To find a right balance, this thesis proposes a procedure which is a heterogeneous composition of SAWSDL concepts and the OWL-S framework. In the OWL-S approach, the complete ontology is developed and the corresponding OWL-S sub-ontology is built parallel to the OPC UA server. Hence, the OPC UA server and OWL-S ontology are completely decoupled entities. On start of the OPC UA server, the accompanying OWL-S ontology is published to the generic method discovery repository. The client in pursuit of method discovery only interacts with the published OWL-S file and comes into contact with the OPC UA server only during method invocation time. In the SAWSDL approach, the developed custom ontology and OPC UA are tightly coupled to each other. The OPC UA server provides compact grounding information that is utilized by the client to locate the application specific methods of the server. In contrast to the OWL-S scheme, the client directly interacts with the semantically annotated OPC UA server over its complete *discovery - orchestration - invocation* cycle. In case of the hybrid approach, this thesis employs the OWL-S framework to develop the sub-ontologies in order to model the methods. These sub-ontologies are annotated to the corresponding nodes in the OPC UA server and compact grounding information is published to the RPL component of the GeSCo layer (see Figure 4.9).

The profile and the method model ontologies of OWL-S semantically describe the capabil-

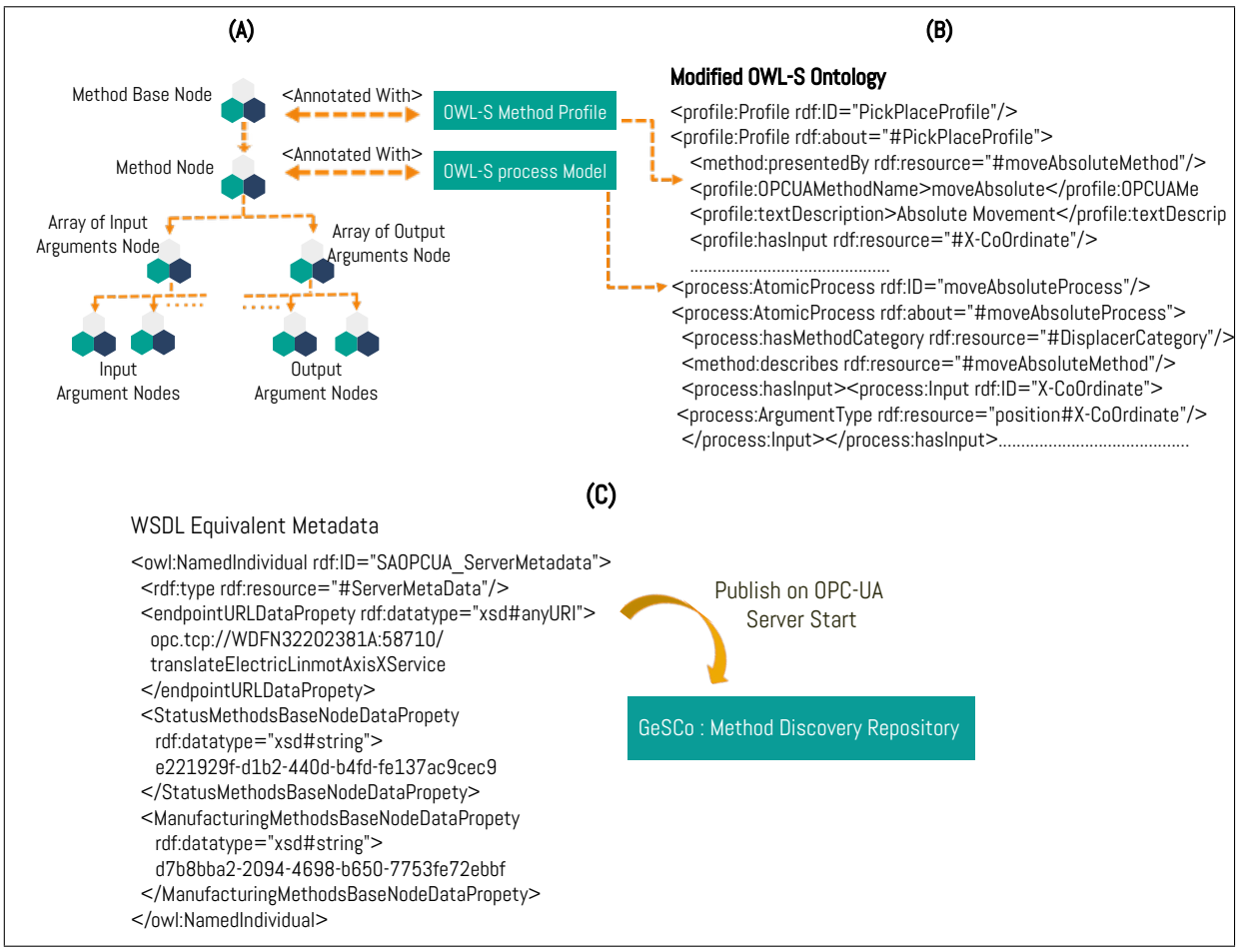

Figure 4.9: Hybrid of OWL-S and SAWSDL specifications applied to a method of a *Pick and Place Robot* for the purpose of illustration: (A) Annotated OPC UA Node Structure. (B) Rough approximation of corresponding OWL-S ontology for node annotation. (C) Grounding information to be published to OPC UA Clients.

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

The profile ontology of the OWL-S framework is attached to the method (both status and manufacturing) base nodes. These base nodes provide a high-level description of the capabilities provided by the OPC UA server as illustrated in Figure 4.9 (A) and (B). The method base node is a parent node for a collection of OPC UA methods. Therefore, it has to be ensured that the profile ontology that is attached to the manufacturing (or status) base method node is generically designed to encompass the functionality provided by all the manufacturing (or status) methods. In accordance with the above argument, the thesis recommends to keep the generic data property *methodClassification* attached to the method

profile ontology, and move the more specific object property *hasMethodCategory* which has more concrete concepts attached down the hierarchy such as *method category*, *execution cost*, and *execution time* to the method model ontology.

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

The organization of OPC UA application methods as described in Section 4.2.1 provides a standard mechanism to browse the manufacturing and status methods. The browse path from the server root node to a specific method node becomes irrelevant in such a scenario. Instead, the node ID of the Application Methods Base Nodes that act as a starting point to browse the methods down the hierarchy can be supplied as a substitute. The only other requirement is the provision of the endpoint URL of the OPC UA. Hence, the endpoint URL along with the application methods base nodes is published to the GeSCo method discovery repository as illustrated in Figure 4.9 (C). Such an arrangement allows to do away with the standard grounding ontology of the OWL-S framework.

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

5 Implementation

This chapter describes in detail the relevance of ontologies in manufacturing and inefficiencies involved in the traditional method of the encoding of ontologies. In order to avoid burdening the manufacturer purchasing the MES with the task of ontology encoding, it then describes a semi-automatic bidirectional transformation of MES source code to a corresponding reference ontology taking into account the OPC UA collaborations (refer Section 5.3). With human involvement, various production and variant-based business rules (refer Sections 5.1 and 5.4) are formulated on top of the generated reference ontology with an intent of reuse. These rules make up for the information loss suffered in the transformation process.

Introduction and Motivation

Future production environments must be flexible and reconfigurable. To achieve this, the devices and services to fulfill the different steps of a production order (PO) should not be selected in the manufacturing execution system (MES), but in an edge component close to the shop floor. To enable this, abstract services in the PO and concrete services provided by the field devices on the shop floor need to refer to a production ontology. The creation of this ontology is a challenge of its own.

The traditional approach has been to design the information model in a formal modeling language such as OWL for the purpose of correctness, and subsequently, this information model is used to build the applications. To this end, the general assumption is that an ontology is predefined by a domain expert, and can then be taken as a starting point for further software engineering processes. However, the ontology-encoding is a time-consuming task. From the perspective of the manufacturer who purchases a MES, ontology encoding and maintenance is an overhead activity in addition to the maintenance of MES. Mere ontology encoding is also not the final exercise in the formal representation of the production information model. Additionally, semantic rule languages need to be employed to fill in the knowledge gap as OWL is not fully capable of expressing all the aspects of the information model due to a lack of constructs. The modeling of product variant based rules in semantic rule languages is an additional task. Additionally, ad-hoc encoding of ontology due to partial state of isolation of various departments involved in various steps of manufacturing processes

create knowledge silos. These production design time activities also border on duplication of work. These diverse ontologies do not facilitate interoperability, and hence, design and PO changes are not transferred to production in real-time. Though one of the use-cases of the creation of ontologies is to enable decentralization of manufacturing, the random ontology encoding defeats the purpose. In order to achieve the objective of decentralization in a dynamic system without introducing a chaotic behavior, it is important that the generated ontology is grounded on description logics, and must originate from a centralized system such as MES.

The thesis proposes a pragmatic automation of an encoding of a primary and light weight production ontology based on the source code of MES. This chapter describes the transformation procedure of source code to resource, product and generic concepts of the manufacturing plant ontology. The proposed approach substantially decreases the ontology engineering effort at design time. To this end, the knowledge of OPC UA collaborations is also exploited during the creation of resource ontologies. The generated ontology is used as a reference to create further rules in OWL-based rule languages.

Due to a fundamental difference between source code implementation (imperative paradigm) and ontology representation (declarative paradigm), the problem of information loss is inevitable. This problem is overcome by the formulation of production and business rules that encapsulate the logic of the MES. The foundation of ontology is exploited to formulate these rulesets using OWL based constructs and OWL based rule languages such as Semantic Web Rule Language(SWRL), Semantic Query-Enhanced Web Rule Language (SQWRL) and SPARQL Protocol and RDF Query Language (SPARQL) based on feasibility and requirements of specific rules. Further, these rulesets are either run on the automatically generated ontology at design time with an intention to enrich the knowledge base, or production runtime to validate the pre-defined business rules between the production steps. The generated ontology also acts as a basis for automatically generating the OWL-S/SAWSDL/hybrid ontologies for the OPC UA application methods for the purpose of dynamic manufacturing service discovery and orchestration. The generated ontology and an abstract production order (PO) hooked with formulated rules are cached to the shop-floor network for consequent production control to enable smart edge production.

Excluding exceptions, the generation of a conceptual ontology and the formulation of the corresponding semantic rules is one-time process. Thus, these ontology concepts and rules can be reused to design the PO, and validate the conditions of the assembly, resources and shop-floor during intermediate steps of the production to determine the next courses of action in production execution and control. In the rare events of changes in the base ontologies owing to the corresponding changes in the source code of MES, only the production rulesets have to be readjusted to reflect the ontology changes.

5.1 Classification of Rulesets

This study classifies the explicit rules that govern the execution of each of the production steps into two categories, namely production process ruleset and business ruleset. These rulesets make further advancement in the direction of formalizing manufacturing domain specific knowledge in order to explicitly characterize the production execution process. These rulesets need to hold true as a precursor to the start of each of the production steps.

A human production designer designs the set of operations, Bill of Materials (BOM) and routing algorithms and sequences through various work stations in the Manufacturing Execution Systems (MES). This step is followed by the creation of a PO which is basically a hierarchical specification of production steps that details the entire production process in the cloud based MES. It also includes predecessor and successor relations, and details regarding parallel processing of production steps. The PO only contains abstract definitions of resources. A concrete mapping to actual resources takes place in GeSCo.

The PO created in the cloud is dispatched to the GeSCo for production which has to perform the task of delegation of manufacturing operations to suitable resources that match in principle with the abstract resources defined in the PO. This task can be accomplished provided that the manufacturing resources in addition to the provision of manufacturing services, also incorporate the functionality of self-description of the offered manufacturing service [Bede17]. The published description along with the formal ruleset facilitate the automatic search, discovery, orchestration and invocation of manufacturing services that perform the transformation steps on the assembly components to produce the end product.

As mentioned before, this research work makes the distinction between the production process and business rulesets. The production process ruleset constitutes general guidelines in the form of OWL class axioms to deduce the inference under the following circumstances:

- Discovery of an OPC UA application method that provides a suitable manufacturing service to execute a manufacturing operation
- Assessment of the compatibility between the production step set points defined in MES with the inputs of the OPC UA application method
- Evaluation of the OPC UA result with regard to the production step objective
- Location of the positions of the manufacturing resources in the shop-floor layout for aiding autonomous guided vehicles or conveyor belts to transport raw materials and products to the corresponding work-stations
- Pre- and post-conditions of the production step execution

The above-mentioned production rules are stipulated in the modified OWL-S ontology which

```
<owl:class ....>
....
<methodProfile:methodCategory rdf:resource="http://emea.global.corp.
    sap/rescom#MaterialProvider"/>
<methodProfile:methodclassification rdf:resource="http://emea.global.
    corp.sap/rescom#Transporter"/>
<methodModel:ManufacService rdf:resource="http://emea.global.corp.sap
    /rescom#DisplacerService"/>
<methodModel:PreCondition_RuleName rdf:DataType="xsd#string">
    check_IsInMotion_Rule</methodModel:PreCondition_RuleName>
<methodModel:PostCondition_RuleName rdf:DataType="xsd#string">
    set_InMotion_Rule</methodModel:PostCondition_RuleName>
....
</owl:class>
```

Listing 5.1: Example of Production Ruleset to Choose the Pick and Place Robot.

is customized to manage the OPC UA method description data. As the production ruleset is related to the OPC UA method offered by the resource, it remains largely unchanged over the life cycle of a resource as the concepts of all the configurations and the topology of the resource are covered in the ontology. The OPC UA servers running in the manufacturing resources publish the OWL-S ontology that contains the production ruleset to the GeSCo method discovery repository, and hence, the production ruleset is stored and validated on the factory shop-floor.

On the other hand, there are certain rules that express the constraints from a business perspective. The constraints that define the business ruleset may be the following:

- Type of sub-components to be used
- Quality assurance activities on sub-assemblies before and after a specific production step
- Definition of alternative routing in case of non-conformance arising out of quality assurance activities
- Selection of an optimal manufacturing resource when more than one resource offer the exact/similar manufacturing services

The business ruleset is formalized, expressed, classified and managed in the cloud based MES. The business ruleset is employed by the production planner during PO design in the cloud MES. In essence, the business ruleset comprises a set of OWL concepts and the associated ruleset, and it varies for each of the product variants. The production planner assigns the relevant ontology concepts to the individual elements of the PO where the ontology concepts represent the manufacturing requirement/service that needs to be fulfilled. This

step is followed by the injection of the business ruleset as a bridge between the ordered set of manufacturing operations in the PO routing plan. These rules are executed during PO execution to assert the validation of the manufacturing process. However, the common objective of both the rulesets is to largely reduce the engineering effort to realize simultaneous and event-based production processes.

Representation of Production and Business Rulesets

The parts of a DL knowledge base are TBox, ABox and RBox. TBox statements in a knowledge base are terminological conceptualizations that describe the intentional knowledge in terms of ontology concepts and their general properties. ABox are TBox compliant statements and contain facts or assertion knowledge, which is associated with the individuals of the domain of discourse [Zhao17] specified using the domain ontology. The RBox of a DL knowledge base allows role-centric modeling constructs that support role-inclusion and role equivalence axioms. An example of an RBox axiom from the research use case is, picks $\circ$ moves $\circ$ places $\sqsubseteq$ displaces, where [$\forall$p$\forall$q (picks(p,q)) $\wedge$ (moves(p,q)) $\wedge$ (places (p,q)) $\rightarrow$ (displaces(p,q))]. In other words, the relation *displaces* is an implication of the conjunction of the *picks*, *moves* and *places* relations. In our research work, a TBox ontology contains the concepts of resources, manufacturing operations, production states, and subcomponents and end-products, while the ABox ontology contains the corresponsing individuals.

OWL-DL is the universally adopted sub-language of OWL in industrial applications [Li18]. OWL enables automated deduction capabilities, notably the automated reasoning that comprises ontology consistency checking, classification of ontology concepts and information retrieval based on principles of first order logic such as subsumption, instantiation and general satisfiability. A combination of a mature standard and a good tool support to OWL also makes it a preferable choice to represent the structured terminological knowledge of classes, their properties, inter-relations and assertions. In addition, an OWL ontology can also be processed by software applications so that it can be easily integrated into the GeSCo architecture. However, the OWL language can only impose simple constraints such as domain, range, existential and universal quantification on cardinality, equivalence and subclass concepts, and property chains. The ontology-based reasoning of OWL cannot express formalized and advanced constraints between inter-structural knowledge bases (TBox conceptualizations and their relations) as its modeling constructs are not adequate for complex scenario representation. Moreover, it also has limitations with regards to computing and related mathematical operations.

The expressive limitations of OWL formalisms in knowledge modeling can be overcome by the alternative paradigm of rule-based reasoning. Among several rule languages such as

RuleML [Rul18], Rule Interchange Format (RIF) [Rul18], SPARQL Inferencing Notation (SPIN) [SPA18a] and Semantic Web Rule Language (SWRL) [HORR04], SWRL was chosen in this thesis due to its proximity to OWL, relatively small learning curve, and user friendly file formats such as Turtle. Additionally, the tool support of the Protégé ontology editor and SWRL engine support of popular ontology reasoners such as Pellet, Drools and Jess make it a preferred choice for implementing rule-based industrial applications. As SWRL is a syntactic extension of OWL, it enhances the OWL-DL expressivity with its ability to formulate rules using a subset of RuleML. Though SWRL provides strong formal guarantees similar to OWL when performing inference, it must be used with due diligence as indiscriminate usage might lead to undecidable ontologies. In other words, no known algorithm entails all possible deductions for all knowledge bases even with unlimited resources and time. Therefore, fragments of SWRL called DL-safe rules which bind concrete values to only the known individuals of an ontology make these rules free of contradictions and hence, decidable in finite time.

The basic form of SWRL is XML, but it can also be expressed in other human readable forms. It supports unary predicates for describing the OWL classes and data types, binary predicates for data and object properties and *n-ary* predicates for some *built-in* functions. Simple rules of SWRL are syntactic sugar of OWL, meaning they can be expressed in either OWL or SWRL. Built-ins, as they are called that support mathematical and logical operations on data properties, dramatically increase the expressivity of SWRL, and help construct complex rules which are not OWL syntactic sugar.

SWRL rules are expressed on the basis of formal deduction sequents which is a kind of conditional assertion to express line-by-line logical arguments. A sequent can be represented as follows:

$$A1, A2, ..., Am \vdash B1, B2, ..., Bn$$

where Ai and Bi are formulae. A sequent is an assertion that whenever all of Ai are true, then at least one of the Bi is true. In other words, the commas on the left of the turnstile are considered conjunctions, and commas on the right of it are considered an inclusive disjunctions. Each formula Ai on the left of the turnstile is called antecedent (also called body) and each formula Bi on the right of the turnstile is called precedent (also called head), and represented as $antecedent \rightarrow precedent$. However, in case of SWRL rule, both the antecedents and precedents of SWRL rule are conjunctions of asserted atoms in the form of horn like rules. SWRL rules also do not support negation and disjunction of asserted atoms. The assertions can take the following forms: $C(x)$, $DP(x, y)$, $OP(x, y)$, $sameAs(x, y)$, $differentFrom(x, y)$ and built-ins, where x and y are variables, and $C(x)$, $DP(x, y)$, $OP(x, y)$ are OWL class description, data and object properties respectively [SWR18a].

SQWRL is a SWRL based query language that provides SQL-like operators in order to retrieve information from OWL ontologies, and inferred knowledge of OWL and SWRL rules. The built-in libraries of SWRL can also be employed in SQWRL statements. As the name suggests, SQWRL cannot write back the aggregated query results to the ontology. Therefore, this thesis evaluation formulates certain rulesets in SQWRL in case of data retrieval operations which do not perform modifications to the ontology database. Continuing the previous argumentation, the thesis evaluation also formulates rulesets in SPARQL which require recursively nested conditions and certain complex scenarios which require greater flexibility in terms of mathematical constructs.

During the course of this research work, the production ruleset is modeled as a combination of class-based axioms of OWL and explicit rules are modeled in formal rule languages. On the other hand, the business ruleset is modeled employing only explicit rules and queries.

5.2 Integration of Ontology to Cloud based MES

The modified automation pyramid proposed in this research work is as illustrated in Figure 3.2. When the manufacturer opts for the cloud based MES, the GeSCo is also shipped as part of the manufacturing solution suite. In order to provide ontological support, the MES vendor must put considerable effort into the ontology development process also known as ontology engineering. It includes the following steps: Determination of scope, enumeration of terms of taxonomy, encoding of the ontology in terms of definitions of classes, definitions of properties and constraints, instantiation of individuals, and design of ABox rules, ontology update and ontology enrichment. In cases of heterogeneous ontologies, ontology alignment (also referred as ontology learning) is another supplementary task. This effort with regards to ontology development and maintenance is additional to the maintenance of the main codelines of the MES and the GeSCo.

In addition to the ontological engineering task, the ontology should also be subjected to adaption, extension and/or reconstruction owing to the corresponding changes in the development codelines of the MES and the GeSCo. Moreover, ontology enrichment which does not change the concepts and relations, but only refines the existing constraints also needs to be handled as a result of fine tuning the source code in MES. On the other hand, delegating the ontology engineering task to the manufacturer is not a feasible solution either. In practice, the domain experts on the manufacturer's side who are involved in PO creation might not have sufficient first hand ontological development experience. Furthermore, the construction of ontologies using techniques of ontology engineering is also a time-consuming task. The fact that MES software development and maintenance, and ontology development require a very different skillset also make complementing the MES with an ontology an incompatible

task.

In such a scenario, the solution to the problem of ontology creation and maintenance to keep it *current* is to automate the process of encoding of the ontology. The ontology corresponding to manufacturing resources is one of the pre-requisites to design the OWL-S/SAWSDL/hybrid OPC UA frameworks described in Chapter 4. Such a manufacturing resource ontology can be extracted from the so-called state of art OPC UA collaborations. The OPC Foundation coordinates with various organizations to create OPC UA common information models of different domains of manufacturing. Its objective is seamless information flow between the diverse manufacturing resources in a platform independent manner. The OPC UA collaboration defines the complete information model of an OPC UA server of the particular *type* of manufacturing resource to integrate the machinery with the manufacturing domain with regards to installation, version management and machine operation. This open and standardized machine model provides well defined profiles for different machine variants and the corresponding functionality offered by these machine variants. Any vendor who supplies the manufacturing resource in this industry vertical has to adhere to this information model.

These open machine models are offered in two flavors: text based portable document format (PDF) and embedded links inside these PDF files to XML documents known as *UANodeSet* files. There are several methods to achieve the process of (semi-)automatic encoding of ontologies: plain text processing and machine learning are notable approaches. Based on these approaches, the following section enumerates different possibilities of encoding the information model of OPC UA collaborations to the corresponding resource ontology.

Different Approaches to Automatic Encoding of Ontologies

This subsection lists different approaches that can be applied for automatic encoding of production ontologies employing the OPC UA collaborations of manufacturing resources serves.

- Ontology generation employing natural language documentation of the standard does not yield an optimal result even after several iterations of review. Even with the best statistical algorithms, this unsupervised activity generates sub-optimal ontologies.
- Another idea conceived by the author is to create a list of terms along with the classification and relationship data, and feed it to the OPC UA collaboration PDF document to create custom tags in the document. The tagged document contains the marked words accompanied by categorization annotation. The tagging of the key words is indispensable for a good translation of the OPC UA collaboration into the ontology. The annotations are created by defining the rules that assign the concepts of the corresponding OWL entities such as *class, objectProperty, dataProperty, entityDomain,*

entityRange, subClassOf and *equivalent*. This controlled ontology generation process is also subjected to repeated iterations with human involvement until all the words are correctly marked with the correct annotation. This step is followed by the supervised encoding step of the ontology.

- The third approach is the exploitation of the UANodeSet XML file that describes the complete information model of the corresponding OPC UA collaboration compliant manufacturing resource. An XML file to ontology transformation is explained in [Bohr05, Van 08].

However, in agreement with the UANodeSet metadata, the state of the art MES software already implements *machine models* using this XML metadata file. These machine models are exploited to represent the *digital twin* model required for condition monitoring from the cloud. Another area of application of machine models in MES is the design of a static routing plan. In such a scenario, the production designer who sets out to create an ordered list of manufacturing services to transform the raw materials to end products has to choose the corresponding manufacturing resources that provide these services. At this point, s/he maps the methods provided by these machine models against the required manufacturing services of the PO.

The author believes that the source code of the cloud based MES that implements these *digital twins* is the right information model for ontology modeling for the reasons explained in the following:

- It is not realistic to assume that every manufacturing resource in the shop-floor possesses a corresponding OPC UA companion specification. However, irrespective of the availability of a UANodeSet file, the MES implements machine models for the reasons stated above.
- It is also possible that a companion specification compliant manufacturing resource might have additional functionality and hence, the resource can have an enhanced information model which is not identical to the information model of the companion specification.
- Furthermore, there are also additional utility and calculated fields in the resource information model of MES that play a vital role in formulating decision-making rules.
- For an effective production control, it is required that the generated ontology covers all the concepts that play a role in the shop-floor such as manufacturing resources, manufacturing operations, product, its variants and sub-assemblies. However, the UANodeSet file only contains the select node information about the manufacturing resource type. The constructed ontology employing the UANodeSet file which only describes the manufacturing resources and their offered services is of little utility in the context

of holistic production control. However, there exists corresponding source code in the cloud based MES that incorporates all the above-mentioned concepts of materials, resources, work-stations, manufacturing operations, BOM, routing plan and PO. Hence, the source code of the MES must be employed to generate a common reference ontology model of the different entities.

- Any change to the information model and/or production control logic in MES is immediately reflected in the ontology when the ontology generation algorithm is triggered on updated source code. This in turn enables the creation and the formal reasoning of production logic based on the latest version of the ontology and thereby substantial reduction of the latency of propagation of source code changes to the real time production.

The author proposes to exploit the machine model data in the cloud based MES to encode the ontology to serve the following purposes:

- Assignment of ontology concepts or/and production process ruleset to PO constituents so that formal reasoning is possible during the selection of suitable services in the manufacturing shop-floor
- Formulation of the business ruleset using the generated ontology that needs to hold true before or/and after a production step during the runtime of PO execution

5.3 Automatic Encoding of the Ontology

Based on the discussion of the previous section, this thesis considers the machine model classes and interfaces that are implemented in the cloud based MES for the automatic ontology translation process. On the flip side, high-level programming languages such as Java and the OWL modeling language belong to the different spheres of software engineering and ontology engineering, respectively. Hence, an appropriate translation mechanism has to be devised to find analogous concepts.

The classes and interfaces of the Java source code of the cloud MES are translated to the corresponding OWL entities. The generalization relationship of the object-oriented concepts is translated to the *IS-A* type of relationship in OWL language. An important point of consideration in the translation process is the handling of operations of a class in java. There is no equivalent concept for an operation in OWL, and moreover, an operation of a class on its own does not convey any special information and at best, it merely participates in the transformation of the value set of the class member fields. Hence, the operations of a class are ignored during the ontology modeling process. However, the logic of transformation of member fields is applied during the construction of SWRL business rules. The associations

```
@ClassEntityAnnotation
public class AutonomousTransporter implements rescomResources, Transporter {

    @ObjectPropertyAnnotation(objectPropertyName = "hasPicker")
    public Picker picker;
    @ObjectPropertyAnnotation(objectPropertyName = "hasPlacer")
    public Placer placer;
    @ObjectPropertyAnnotation(objectPropertyName = "hasMover")
    public Mover mover;
    @ObjectPropertyAnnotation(objectPropertyName = "currentPositionX")
    public XCoOrdinate currentRobot_X_CoOrdinate;
    @ObjectPropertyAnnotation(objectPropertyName = "currentPositionY")
    public YCoOrdinate currentRobot_Y_CoOrdinate;
    @ObjectPropertyAnnotation(objectPropertyName = "currentPositionZ")
    public ZCoOrdinate currentRobot_Z_CoOrdinate;
    @DataPropertyAnnotation
    public long x_Direction_Acceleration;
    @DataPropertyAnnotation
```

Figure 5.1: Annotations on MES Source Code.

and aggregations are translated by means of the *subject-verb-predicate* model of the semantic web. Associations and aggregations are allowed in object-oriented concepts, but it may introduce contradictions in strict modeling language like OWL. Therefore, these association fields are annotated with different values to differentiate during the encoding of the ontology. The authors define annotations on these fields that are available at execution time via Java reflections. These annotations provide additional information to the encoding algorithm to construct the *subject-verb-predicate* model of the OWL object properties (refer Figure 5.1). The annotations on the MES source code present an additional, but small overhead activity to the developers in the form of documentation that states the purpose and context for smooth code modifications and extensions in the future.

The transformation rules of source code to OWL ontology

The source code of the machine model falls into the category of software engineering / implementation, while the encoding of the ontology is categorized under ontology engineering / knowledge modeling process. The authors propose the transformation rules from object-oriented source code in cloud based MES to formal ontology generation as depicted in Table 5.1. These rules are integrated into the application that performs the above-mentioned transformation. The OWLAPI [Horr11] which is a Java API for the construction of ontologies in OWL in an application is employed to realize this step of the process. The OWLAPI is capable of generating, querying, deleting and reasoning on even the most evolved ontology. In other words, it provides all the features provided by the Protégé-OWL API [Pro19].

Table 5.1: Rules to transform Source Code to Formal Ontology.

Source Code Attribute	**OWL Entity**	**Comments**
Project source code files	Ontology File	The complete ontology corresponding to the machine model source code is stored into a single ontology file
Class	Class	Both the source code and OWL ontology have identical class concepts. The fully qualified name of the class is used in naming the ontology class to prevent ambiguity.
Primitive class member field	Data Property	The primitive types in source code are translated to the corresponding *Datatype* in OWL
Primitive member field name	Name of the Data Property	——
	Domain and Range of Data Property	The encircling class in the source code is made the domain of the encoded data property. The encircled data type of the primitive field is made the range of the encoded object property
Complex class member fields/Composition/ Composite Aggregation	Object Property	The class and complex fields are related by a *whole-part* relationship. The *part* fields are related to the *whole* by an object property
Binary Association / Aggregation	Two Object Properties	The associations are annotated with different values. Therefore, two object properties are encoded, and they are related by means of *inverse* relations in OWL.
Association / Aggregation / Composition	Domain of Object Property	The encircling class in the source code is made the domain of the encoded object property.
	Range of Object Property	The encircled complex data type is made the range of the encoded object property

	Cardinality of the Object Property on Domain and Range entities	There is no general pattern in the object-oriented paradigm to represent this aspect of ontology. Hence, it does not figure in the ontology transformation process.
Annotations on Association/Aggregation/ Composition fields	Name of the Object Property	The complex member fields are annotated with the object property name. This value is extracted at runtime using Java reflection.
Inheritance / Generalization (Implementation and Extension)	subClassOf restriction	The interface and classes that a class inherits in source code are translated to the OWL subClassOf restriction.
	DisjointWith restriction	In the absence of inheritance in source code, all the OWL classes are marked disjoint with other classes. However, for the sake of simplicity, this fact is ignored in the ontology.
	EquivalentTo restriction	The classes in the source code are unique, and hence, the case of equivalent classes in OWL does not arise.
Class Instance	OWL Individual	The instantiated class object in source code is equivalent to the OWL type individual.
Class Methods	Equivalent SWRL rules	There is no behavioral concept of entity that performs data transformation in OWL. Instead, this shortcoming is overcome by the formulation of SWRL rules.
Suitable Annotation on source code classes and fields	Annotations on TBox and ABox entities	The OWL annotation properties such as comments, label, deprecated, versionInfo et cetera are represented in suitable in annotations in source code.
——	Class necessary restrictions	The object-oriented concepts only support IS-A, whole-part, has-a relationships between classes. There exist no further constructs in OOPS to express further formal restrictions that can be placed on OWL classes.

——	ABox restrictions	If this step is subjected to automation, it reduces the reusability and also increases the development effort in the long run. Hence, the necessary production process and business ruleset has to be encoded in SWRL/SPARQL by a human expert.

The output of this process is the generation of a formalized ontology file that corresponds to the concepts of manufacturing operation, product, routing and resources. The development of a formal description of inter-relationships between software artifacts is not the end goal. They merely provide a medium to the rule-based applications for the formal analysis of various concepts. In other words, the generated ontology is used as a reference to construct the ruleset for the defined concepts and also to deduce the inferences. Arguing on the same lines, even though the above-mentioned transformation rules provide guidance to transform a class object to an OWL individual, the author only automates the TBox ontology generation process in practice. Later, during the description of concrete manufacturing services in shop-floor, in OWL-S framework for example, ABox ontologies are also created as instantiations of earlier generated TBox ontologies.

It is also to be noted that the automatically generated preliminary ontology does not describe the intricate details of all the entities of the production process. It only describes the necessary concepts that are required for manufacturing automation through automatic resource discovery based on the offered methods and their characteristics, and support the formal design of the business ruleset. In contrast, manually built ontologies are much larger and more complex which might not necessarily be an advantage. In practice, only a small portion of an ontology is usually reasoned on business ruleset, and for the purpose of automatic manufacturing service and subsequent resource discovery. A larger ontology also necessitates larger processing times, which goes against the principles of high speed manufacturing. To that end, a relatively small ontology is generated during the research evaluation and feasibility analysis phase containing around 750 ontology concepts that are relevant to only the key finder unit PO at hand.

This generated ontology is distributed among all the actors of the manufacturing system namely cloud MES based applications in general, and in particular, the common semantic model component of GeSCo. This ontology is the underpinning for the construction of manufacturing resource ontologies. These ontologies, in turn, formally describe the various methods and the corresponding manufacturing services offered by these methods which are based on the SAWSDL/OWL-S framework.

The complete process of ontology generation to formulation of production and business rules in terms of SWRL, SQWRL and SPARQL specifications [SPA18b] is shown in Figure 5.2.

Additionally, the terminological knowledge expressed by the generated ontology is exploited to construct SWRL/SPARQL rules so that the design and manufacturing knowledge can be combined to formulate complex business constraint rules and inference rules. The business rules that are slotted in between the various production steps provide clear and event-based guidelines to the production orchestrator and hence, facilitates smooth process automation. Based on the classes and properties modeled with formal restriction that have been defined in the generated ontology, the production designers model the PO and write business rules. Details follow in the experimental implementation Section 6. The business ruleset offers

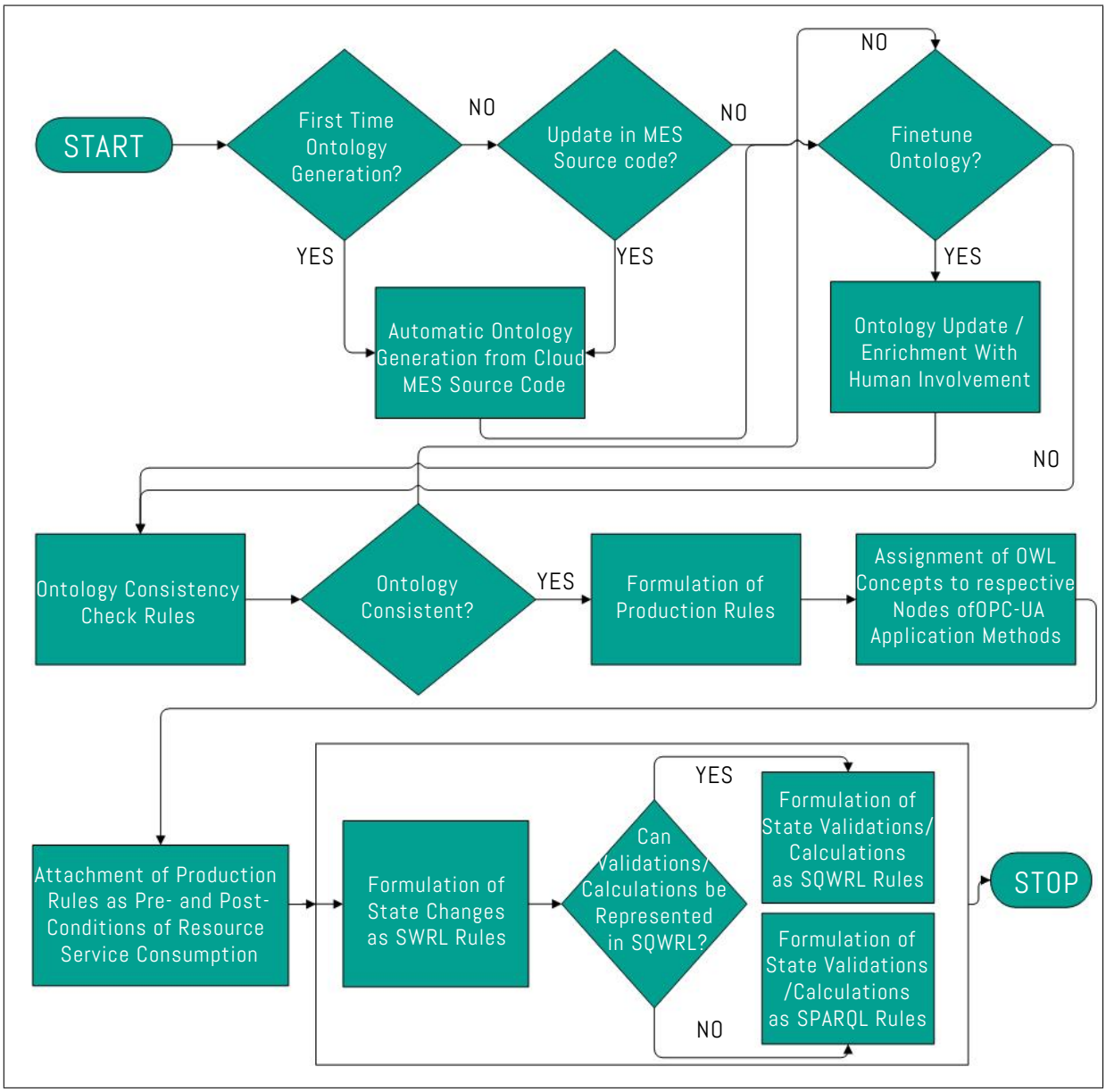

Figure 5.2: Complete Cycle of Automatic Ontology Generation and Formulation of Rules with Human Involvement at Production Design Time.

enhanced expressiveness to the PO. In contrast to centralized automation, where processes are explicitly specified, this approach specifies the processes and control of production logic through interrelated production and business rules.

Another use case of SWRL rules can also be the automated validation of a designed ontology. The SWRL rules that are formulated with such *assert* statements corroborate the consistency of the generated ontology.

5.4 Creation of Ontology Resolution and Business Rules in SWRL

When the common semantic model has been generated, the production designer can import the ontology file into Protégé. At this stage, the production designer, if familiar with ontology engineering, can enhance the ontology by adding or fine-tuning the automatically generated ontology. Such ontology resolution rules are shown in Listing 5.2. The traditional application of SWRL is the creation of data and object property assertions, and inference about the presence of individuals belonging to an OWL ontology class.

The next course of action involves the addition of new data properties which are the prerequisite for writing SWRL rules. The values of these data properties are indicative of the result of the execution of SWRL rules. In other words, usage and execution of SWRL rules sets the values of these data properties, and GeSCo takes cues from this information to decide on the next production step. This step is followed by the imposition of *SWRL IF-THEN* constraints on the assertion knowledge of resources, method characteristics and constraints on production steps of the PO execution by the production designer. Listing 5.3 provides examples of a few business rules involved in production of the key finder.

Unlike OWL that models declarative knowledge, these SWRL business rules (refer Listing 5.3) provide imperative knowledge that compensates for the shortcomings of the generated OWL ontology. The argument for imperative knowledge is even more applicable to define the behavior of the OWL individuals and their relationships. From the perspective of the use-case of this research work, these SWRL rules do not model static descriptive knowledge, but dynamic production knowledge. The SWRL rules assert the conditions of the shop-floor, and accordingly set the data properties added earlier. It has to be considered that the complex and concrete logic applied on production data at the MES level cannot be subjected to one-to-one translation into OWL language and the corresponding rule languages. However, the same rules can be formulated in a more formal manner in ontology languages that define the processes and their requirements in an abstract way and thus delegate the concrete decision-making responsibility to the GeSCo. To this end, the OWL ontology in conjunction with the

```
Rule_1
rescom_resources_AutonomousTransporter(?at) ^
   rescom_capability_abstracts_Picker(?m) ^
   rescom_capability_abstracts_Placer(?m) ^ hasPicker(?m, ?p) ^
   hasPlacer(?m, ?pl) -> rescom_concepts_Displacer(?m)

Rule_2
rescom_capability_concrete_MaterialSupplier(?ms) ^ supplies(?ms, ?mat
   ) ^ rescom_product_LowerShell(?mat) ->
   rescom_resources_LowerShellProvider(?ms)

Rule_3
rescom_capability_concrete_MaterialSupplier(?ms) ^ supplies(?ms, ?mat
   ) ^ rescom_product_UpperShell(?mat) ->
   rescom_resources_UpperShellProvider(?ms)

Rule_4
rescom_capability_concrete_MaterialSupplier(?ms) ^ supplies(?ms, ?mat
   ) ^ rescom_product_CircuitBoard(?mat) ->
   rescom_resources_CircuitBoardProvider(?ms)
```

Listing 5.2: Fragment of Ontology Resolution Rules for ResCom Demonstrator Ontology.

```
Rule_1 : <Is_LowerShell_Identified>
        rescom_product_LowerShell(?lowerShell) ^ hasColor(?lowerShell
           , ?clr) ^ rescom_concepts_Color(White) -> sqwrl:select(
           true)
        OR
        rescom_product_LowerShell(?lowerShell) ^ hasColor(?lowerShell
           , White) -> isIdentified(?lowerShell, true)

Rule_2 : <Can_Assemble>
        rescom_product_LowerShell(?lowerShell) ^ isPlacedInAssembly(?
           lowerShell, true) ^ rescom_product_CircuitBoard(?
           circuitBoard) ^ isPlacedInAssembly(?circuitBoard, true) ^
           rescom_product_UpperShell(?upperShell) ^
           isPlacedInAssembly(?upperShell, true) -> sqwrl:select(true
           )
```

Listing 5.3: Examples of Business Rules for the Key Finder PO.

SWRL rules provide situational awareness and the corresponding in-memory computation. Therefore, GeSCo need not play a subsidiary role to MES anymore as it does not rely on the guidance of the cloud MES after the production data, OWL ontology and associated SWRL rules are cached to the production network. It can arrive at decisions locally by interaction between the modules. These decisions have traditionally been taken by centralized MES. This line of reasoning also diverges from the concept of the classical automation pyramid (refer Figure 1.1) which puts emphasis on the strong division of responsibilities.

The relative ease with which these SWRL rules can be written using a tool such as Protégé makes it easier to formulate contextual rules for a production designer with only a little experience in the knowledge modeling realm. The creation of SWRL rules is product variant based, and it formally describes the dynamic contexts of the shop-floor environment at different points of production runtime. Hence, automation of this step is not possible. However, the ontology and SWRL rules that are constructed for a specific product variant can be reused for similar POs in the future.

Additionally, SWRL rules can be attached to the reference ontology file in one of the many supported syntaxes such as RDF/XML format, or they can be constructed and consumed programmatically on the fly in plain text form. It is recommended to consume the SWRL rules programmatically [Amer12] and the argument becomes even more applicable if the parameters to be fed to the rules are generated at program run time. Hence, the business rules are attached in plain text form to the PO sub steps (refer Figure 6.5).

Nevertheless, the point of contemplation is that certain business rules cannot be evaluated in the shop-floor even at the expense of increased rule complexity. This could be due to a variety of reasons such as reliance on other applications for the production step validation, requirement of large amounts of computing power to arrive at the validation decisions and in some cases, inability to represent a rule that processes a large amount of data due to the lack of efficient constructs in OWL/SWRL combination. In such situations, a trade-off should be made where such rule validations are retained in the cloud based MES.

Furthermore, the SWRL rules need to be manually inspected and changed if necessary in the event of change in the ontology necessitated by the corresponding change in the MES source code. The construction and inference of SWRL rules does not take into account the original OWL ontology restrictions, and hence, newly added SWRL rules may contradict earlier constraints. Hence, in order to assure the ontology consistency, the production designer should incrementally add the rules and run the reasoner to ensure the updated ontology does not contain contradictions.

5.5 Centralized and Semi-Automatic Modeling of Capabilities of Manufacturing Resources

In order to provide complete control of the production processes to the centralized decision systems and also eliminate the prospect of low degree of correlation with the automatically generated ontology, the author of this work does not recommend a decentralized modeling of manufacturing resources and their manufacturing services. The bottom up approach of resource capability may also introduce *knowledge gaps* from the standpoint of centralized production planning and hence, the system may experience a chaotic behavior. This knowledge gap may arise due to the usage of different ontological concepts stemming from different taxonomies. It may also be attributed to thought processes of designers/engineers at the factory shop-floor differing from the centralized production planner. Similar to the discipline of software engineering, ontology engineering also faces the discrepancies that may arise between conceptualization and realization phases in case the responsibilities are delegated to different stakeholders. Some concept(s) might be missed and an equivalent concept(s) might be added in the event of unsupervised modeling by the production resource ontology designer. In such a scenario, the additional task of determining the correspondences between the various concepts needs to be carried out to iron out the asymmetricities.

From another standpoint, the natural alternative is the creation of separate ontologies for the business and production systems in the absence of a central ontology. Subsequently, the business rules that are constructed and stored in the cloud MES are no longer compatible with ontologies constructed at the shop-floor for the execution of process validations in the production runtime. Hence, they cannot be directly used in the production without subjecting them to advanced ontology aligning treatment. The alignment process based on SWRL rules is more complex than basic ontology alignment, and involvement in such a process defeats the purpose of keeping the manufacturing planning and execution simple and transparent.

In order to avoid this situation, entities that participate in the production planning and production control must operate on a single centralized ontology. This in turn allows the cloud MES to have a global view of all the concurrent processes and hence, exercise more control over the shop-floor. This scheme also allows a unified integration of all the production participants, and thereby plays a vital role in realizing the shared visions of the organization. [Mena00] also stresses that services based on agents and one ontology offer flexibility and adaptability with a low overhead.

The generated common semantic model is pushed to the production network in general, and to the GeSCo in our research use case. The OPC UA servers of the manufacturing resources are then formally modeled using the SAWSDL/OWL-S/Hybrid framework on top of the

automatically encoded ontology. The various aspects, configurations and capabilities of the manufacturing resources are instantiated into OWL individuals using OWL constructs such as *equivalent, subClassOf, Instance, sameIndividual* and *DifferentIndividual*. The generation of this ABox ontology cannot be generated in an unsupervised environment and hence requires the involvement of a human expert. The human expert also formulates the SWRL rules that depict the pre- and post-conditions on the product, resource, transportation facilities, PO and the environment on account of provision of manufacturing service by the manufacturing resource. With this information at hand, a paradigm shift from machine driven production definition to information and context driven control processes is possible.

Figure 5.3 shows the UI designed to capture the OWL-S ontology of each of the methods of the manufacturing resources based on OPC UA servers. This web application tool is hosted on the cloud based MES and was created as part of this research prototype. It allows to create an OPC UA client session to connect the HTTP WS-* SOAP-based resource servers of the key finder demonstrator that are wrapped by a single OPC UA server for the sake of simplicity. The tool captures the information of various aspects of a manufacturing service provided by an OPC UA method in terms of the semantic concepts of the reference ontology. This information is exploited to create a corresponding OWL-S ontology which contains the profile, method model and grounding ontologies with regards to a specific method of the OPC UA server of a resource using OWLAPI. These semi-automatically generated OWL-S ontologies of various manufacturing resources of shop-floor are cached at the production network for local manufacturing service discovery.

Figure 5.3: Partial Screenshot of Cloud based MES UI to generate the OWL-S ontology for the *moveAbsolute* method of the Pick and Place Robot.

The generated OWL-S ontology along with the resource ABox ontology is relevant during the PO orchestration by the GeSCo. The required manufacturing service for each of the production steps of the PO is compared to the offered services in these OWL-S ontologies, and a best fit service is chosen to execute a manufacturing operation corresponding to the production step. A case of subsumption reasoning is shown in Figure 5.4 where one of the production steps of the PO puts forward the abstract demand for a *PressService*. This PO abstract demand is associated with the concrete manufacturing service supply, named *PneumaticPressServiceInstance*, of the factory shop-floor. The manufacturing service supply is an instance of the concept *PneumaticPressService* which is subsumed by the required demand concept of *PressService*.

As an alternative to automatically generating an OWL-S ontology, the SAWSDL specification extended to OPC UA application methods can also be used as a substitute for the purpose of dynamic production orchestration [Katt18a]. Though the SAWSDL specification reduces the degree of match of a particular manufacturing service, it simplifies the process of finding the right manufacturing services. Independent ontology or OWL-S based ontology concepts can be annotated to resource OPC UA servers as illustrated in Figure 4.6 in order to imple-

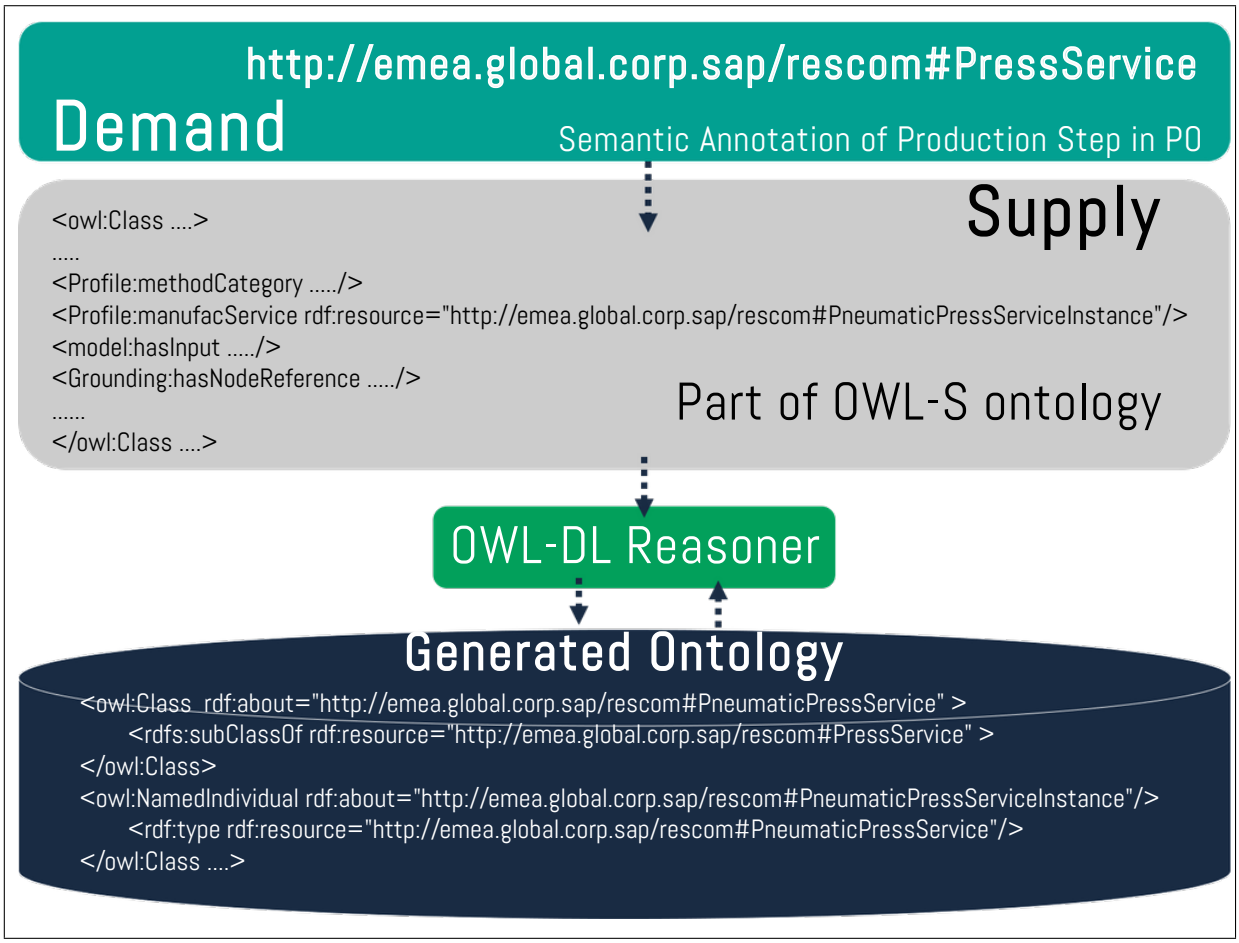

Figure 5.4: Semantic Manufacturing Service Demand-Supply Matching: Simple Case of DL-Based Subsumption Reasoning.

```
Resource (?rsrc) ^ hasPickerTool (?rsrc, ?piTool) ->
  PickerResource (?rsrc)
```

Listing 5.4: Explanation of DL Safe Assertions in SWRL.

ment SAWSDL or hybrid OPC UA servers, respectively. For the sake of completeness, this research evaluates the efficacy of the automatically generated ontology, and the subsequently constructed ruleset in orchestrating the key finder production workflow using all the three proposed semantic OPC UA frameworks.

The generation of an ABox ontology involves not only the creation of a skeletal system, but also includes instantiation of the inner classes and properties of the main concepts. Therefore, when a manufacturing resource is instantiated in the resource maintenance UI in the cloud based MES, the corresponding class along with its sub-component classes and their properties are also instantiated in the ontology. For example, suppose a *Resource* concept is instantiated and is assigned as *domain* of the *hasPickerTool* object property. However, in the absence of a named individual of the range of the *hasPickerTool* object property (for example, the '?piTool' individual), there is no *hasPickerTool* object property assertion. In such a case, when the SWRL rule shown in Listing 5.4 is executed, the resource individual does not transform into an instance of *PickerResource*. This is due to the fact that SWRL rules are DL safe [Klus], and hence do not bind values to individuals that are not explicitly known at the time of the invocation of the SWRL rules. Therefore, a successful firing of a rule only takes place when all the instances of the ontology concepts that are part of the rule are instantiated. However, the DL safety of an ontology is only safeguarded by imposing restrictions on the inference engine of the SWRL reasoner, but SWRL rules on their own may still yield an undecidable ontology.

5.6 Ontology as Placeholder for Manufacturing Data Collection

During production runtime, the generated ontology also serves the purpose of performing data collection, and stores the PO specific data in the PO individual entity of the ontology. GeSCo does not report the result of each of the production steps to the central cloud MES during production control. Instead it accumulates the data in the ontology over the course of PO execution, and at the end of it, uploads the entire ontology to the cloud based MES. The MES performs the reverse engineering of transforming the ontology to the corresponding instance of the java classes, and invokes the appropriate transactions to store the collected data into cloud MES information systems. Instead of transmitting the raw data results after the completion of every production step, this technique provides a platform for GeSCo to

filter and analyze the data at the factory network level, and hence, substantially reduces traffic to the cloud.

It is clear that the source code to ontology transformation procedure in general is not reversible. The nature of data representation and processing in both the formats is completely different. However, the MES is aware of both the structure of the reference ontology as it is generated consuming its own source code and the source code to ontology transformation algorithm. Therefore, the extraction of information from the reference ontology which is updated over the course of PO execution to persistent memory in the MES is a relatively uncomplicated process.

An alternative to sending the whole ontology at the end of PO is to send the partially updated ontology at the end of every production step of the PO in a one directional asynchronous pattern. This reduces the effect of the network latency from the cloud communication perspective, and also keeps the MES updated with the real time data and activities of the manufacturing plant. This data in the cloud MES can be transformed to intelligent information, and further utilized for the purposes of visualization and condition monitoring.

The described approach involves certain repetitions in tasks such as the definition of various entities involved in the production and their relationships, and the capability of resources and their application specific methods of the OPC UA server at design time so that these capability descriptions serve as pointers to discover the required manufacturing services at production runtime. However, these pre-production tasks are necessary to establish a harmonious balance between the centralized systems that plan and schedule on a high level, and decentralized systems that are involved in detailed level scheduling of the PO and control of operations. Subsequently, the above-mentioned approaches also effectively manage the complexity of the production workflow processing.

6 Experimental Evaluation

6.1 Infrastructure Set-Up

The demonstrator system which was originally developed in the RES-COM project [RES18] (see Figure 6.1) that produces smart key finders was enhanced in order to realize the Proof-of-Concept and evaluate the applicability of semantic frameworks presented in Chapter 4 based on the generated ontology using the methodology described in Chapter 5 in the context of proposed GeSCo architecture described in Chapter 3. The resources of the key finder work station are augmented with embedded systems to transform them into cyber physical production systems. The demonstrator setup contains industrial equipment from various vendors which constitute a lot size 1 production cell where three individual parts of the key-

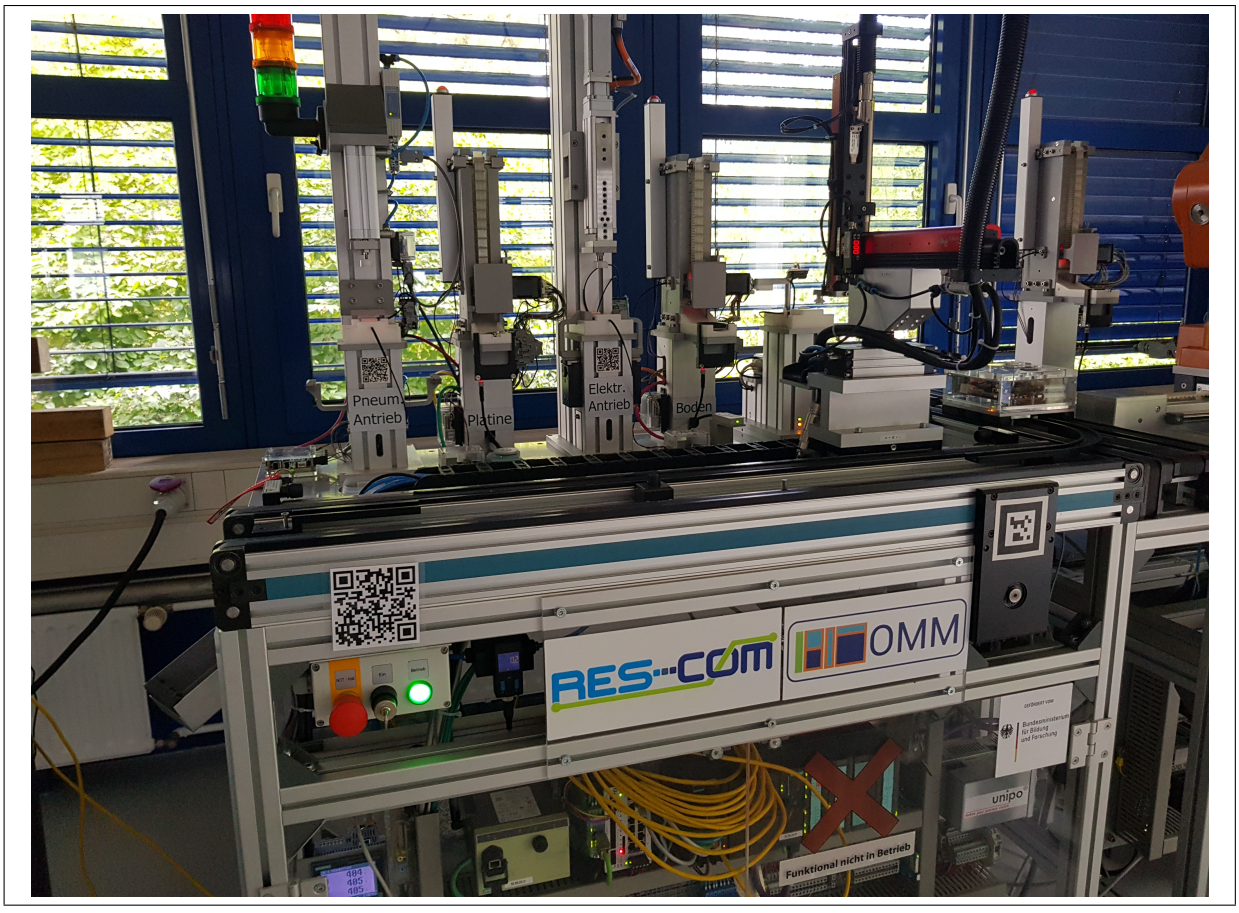

Figure 6.1: Automated Key Finder Assembly Demonstrator.

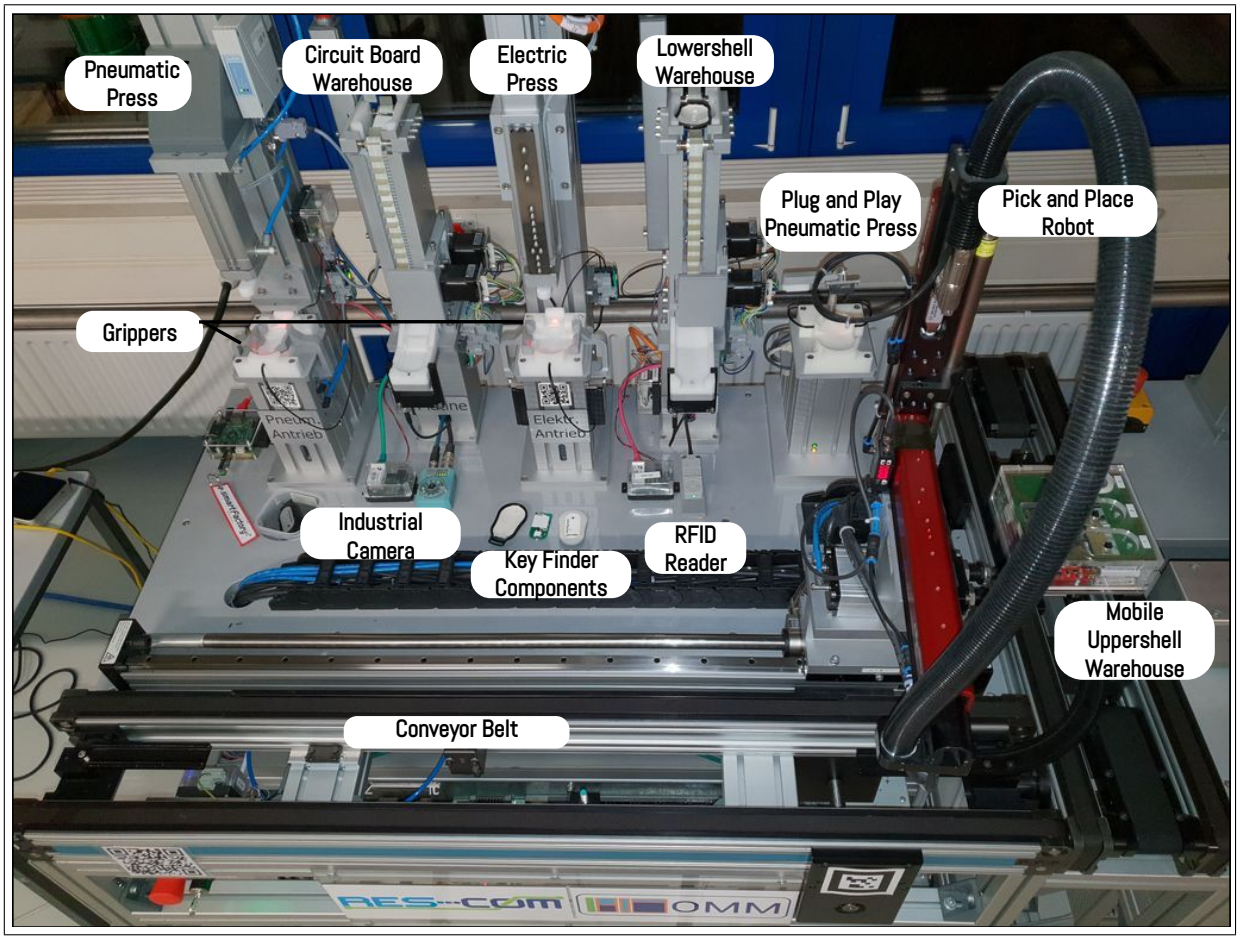

Figure 6.2: Layout of the Key Finder Demonstrator.

finder, namely, housing cover, housing base and the circuit board, are assembled. The work station has multiple key finder assembly units and a general purpose pick-and-place robot which makes it convenient for experimenting with adaptability and reusability features of the manufacturing resources (see demonstrator layout in Figure 6.2). Besides the provision of rich process variants, the demonstrator also consists of infrastructure for the material flow, raw material warehouses and quality control identification systems totaling nearly 50 field devices.

The demonstrator landscape is designed as SOA based control architecture to enable collaborative industrial automation. Several embedded systems expose the capabilities of varying complexity from simple services of field devices to composed services of complex mechatronic production modules over local area network. Such a landscape where the resources are monitored and controlled during production over the network instead of hard-coded standard Programmable Logic Controllers (PLC) programs is conducive for achieving decentralized control of the production. From the outset, the key finder demonstrator modules are controlled via SOAP based web services. OPC UA server wrappers were created around each of these SOAP based services to make them viable for experimentation of research topics of this thesis.

The digital representations of real manufacturing resources are stored as models in the cloud based MES. These models serve the purpose of product and process planning, validation, and execution [Tao18]. These virtual models are associated with real manufacturing resources on the shop floor to pull/push real time data about status of resources and processes. The cloud based MES transforms this raw data to information, and enables to take automated or human centered decisions based on the statistical process control activities. Until the introduction of OPC UA Collaborations, there was no standardized modeling procedure for these digital resource assets. With the direction from the OPC UA collaboration UANodeSet files, the MES implements the digital representation of assets.

Existing MES solutions proved to be inflexible to experiment since they are passive in behavior and hence, do not voluntarily react to the conditions of the shop-floor, and also percolate the changes in the PO to GeSCo. In general, the MES solutions provide directions to the events of the shop-floor only when the information is sought. In order to engineer a seamless change in PO and have more control over the simulation, a cloud based MES was developed that emulates the real MES in the context of production design, planning and execution. This simulation cloud based MES which is implemented over the course of this research also contains the information models of manufacturing resources of the key finder demonstrator in the form of Java source code. The OPC UA servers of the demonstrator modules which are designed to communicate with the programmable logic controllers to control the kinematics of the mechanical manufacturing resources do not conform to any of the OPC UA *collaboration types*. In addition, the evaluation of thesis concepts is executed under the pretense that the information models of manufacturing resources recovered from the source code of the cloud MES are designed based on the resource *UANodeSet* files.

The SAP Plant Connectivity (SAP-PCo) [SAP19a], which is a framework of set of services and management tools was chosen as a basis for GeSCo. SOAP, REST and an ODATA [Auer07] based web servers, and OPC UA client and servers were implemented inside the PCo. During the research evaluation, barring the *Production Process Logger* module which is related to supporting the analytics in the cloud MES modules, PCo was architecturally enhanced to accommodate all the functional modules of GeSCo (refer Figure 3.3). The GeSCo modules were developed inside a Dynamically Linked Library (DLL) along with a set of wrapper operations that were exposed as both web service operations and OPC UA application specific methods that contain the production execution logic. The methods/operations encapsulated in the DLL enable to cache the production control and routing data, and also embed the orchestration plan algorithms via the design principle of *dependency injection*. This concept is also called the Enhanced Method Processing (EMP) [SAP19b] in SAP-PCo product terminology. Furthermore, the EMP implementation assists in behavior specification of the edge component by allowing flexible definition of the actions that need to be executed when invoked by web service or OPC UA client. By inheriting the API class of

the PCo, the EMP DLLs are implemented independently to configure the actions that need to be executed during the production runtime. The resulting compiled DLL is imported into the PCo agent instance at design time and the resulting loaded operations/methods are hooked onto the PCo SOAP/REST/OData Webserver(s) and/or OPC UA server(s) (refer Figure 6.9).

Simulation to measure Cloud Network Latency without Formal Semantics For the purpose of this simulation, the cloud MES was geographically separated by approximately 1000km from the GeSCo and demonstrator system to reproduce the typical network latency involved with cloud solutions, whereas the GeSCo and demonstrator system were deployed on the same Local Area Network (LAN). A production process without exceptional scenarios which corresponds to the lot size 1 production use case illustrated in Subsection 6.1 where production routing contained manufacturing operations that were distributed to resources in a fixed manner was conceptualized, and the corresponding PO was created in cloud MES. GeSCo received only the individual production steps of PO, and communicated the corresponding results from and to the cloud MES, respectively, in order to measure the network latency encountered during the production execution. The network response time, which is a combination of network latency plus computation time, was measured in the SOAP UI tool [SOA19]. However, the simulation cloud MES did not have the data complexity observed in the real MES, and hence, the computation time was very small in comparison. Therefore, for practical purposes, this network response time can be considered as minimum baseline. The simulation results showed an average network latency of approximately 400 milliseconds between GeSCo and cloud based server. In the event production execution and control modules of MES continue to reside in cloud infrastructure, this network latency scales linearly with the number of production steps involved in a PO, and such large waiting times are not viable in the context of high speed manufacturing. Reduction of these high waiting times is the secondary goal of this thesis.

6.2 Production Design Time Activities

The automatic generation of the manufacturing ontology of the demonstrator based on the simulated cloud MES source code with the aid of OWLAPI is the first step among the design time activities. The rules laid out in Table 5.1 are followed for this ontology transformation. The jar files corresponding to the source code of the information model of the demonstrator modules are fed to the automatic ontology generation application which also imports OWLAPI. Using Java reflection concepts, the source code is analyzed and the corresponding ontology is generated in the form of OWL classes, object and data properties, and similar class expressions. A screenshot of a fragment of the generated ontology is shown in Figure

6.3. Typically, the generated ontology originating from the source code of MES modules will not contain fine grained ontologies pertaining to field devices. However, the abstraction of lower level services offered by field devices as illustrated in Figure 4.1 makes field device ontologies obsolete.

The next step is the formulation of the production and business rulesets. In the intelligent key finder manufacturing use case, most of the rules can be stated in the rule languages. However, some operations in the PO are executed if certain conditions hold true. Such

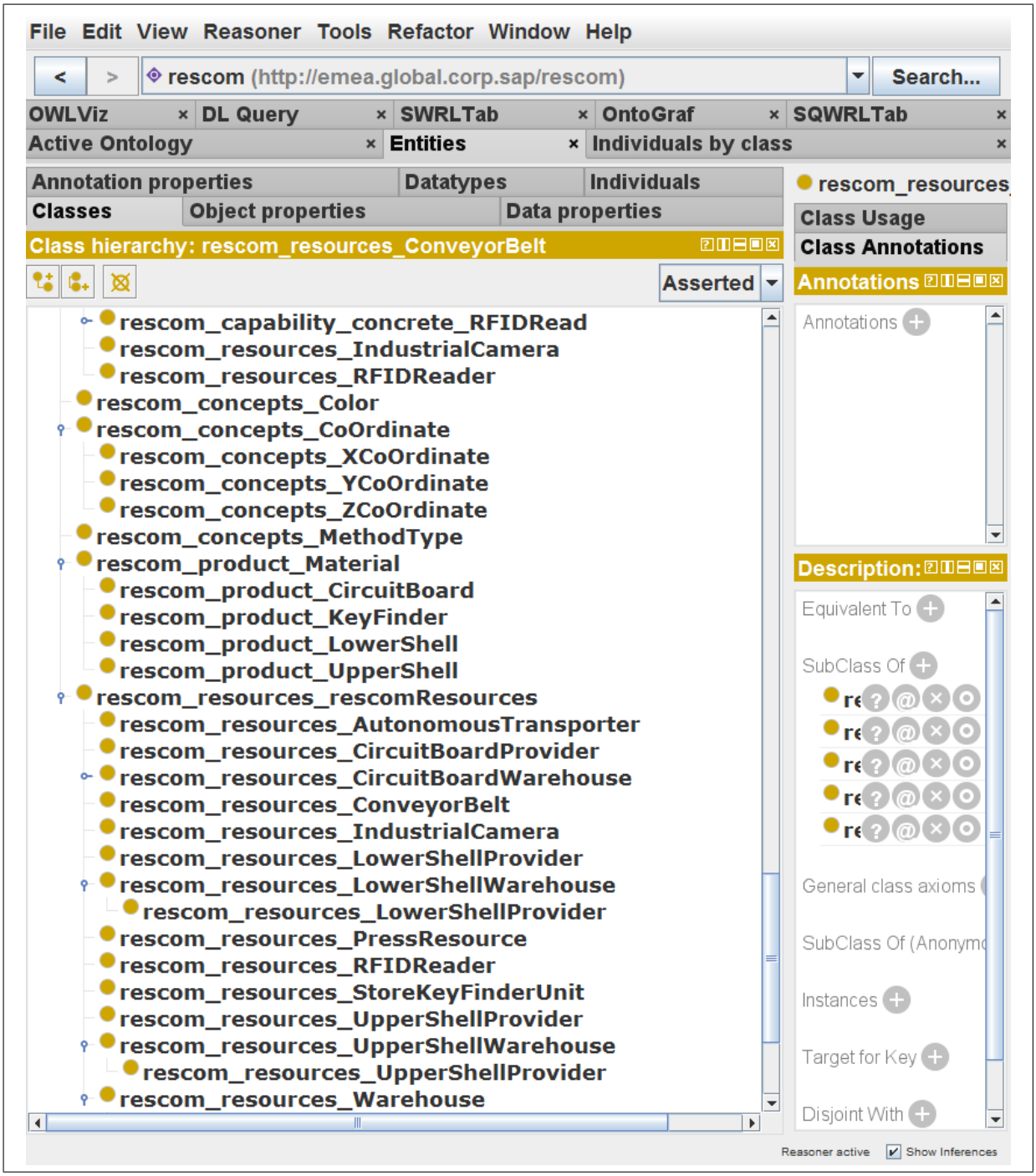

Figure 6.3: Fragment of the Automatically Generated OWL Ontology in Protégé.

assert statements require data property characteristics that represent the status of each of the production steps of the PO. The use case PO contains 35 production sub-steps and if each of the steps requires formulation of pre-conditions before delivery of service by the manufacturing resource, it necessitates defining 35 different data properties to store the status of the pre-conditions. A single data property that reflects the validation result of pre-conditions of the current production step cannot be updated multiple times for all the production steps of the PO. This is due to the property of monotonicity in description logics that does not allow modification or retraction of existing information in an ontology. If an assertion is updated due to a successful firing of a SWRL rule it results in that property having multiple values. When validations are performed against such a property, the validation result is not correct. For example, an update of the *isInMotion* data property of the pick and place robot from *false* to *true* when it starts moving results in two data property assertions with both *off* and *on* instances. The solution to overcome this property update problem is to programmatically remove the existing property prior to saving the new value.

The alternative is to use an elegant querying language known as SQWRL [Sem18] based on SWRL that provides operators to extract information from an OWL ontology. SQWRL queries only gather information and compute aggregations. However, they do not allow to write the query results back to the ontology. Such SQWRL queries are suitable to compute the rule validation result of the current production step, and apply it directly in the application without the need to store it in a result variable. Both the SWRL rules and SQWRL queries were executed using the Java based SWRLAPI [O'Co08], version 2.0.5.

SWRLAPI comes with two major advantages over the other popular open source reasoners such as Hermit [Her18]. SWRLAPI contains implementations to a number of built-in libraries such as temporal built-ins, mathematical built-ins, extensions built-ins, and string, boolean and date built-ins, and thereby largely increases the expressivity of the rules. It also provides a powerful extension mechanism to define user-defined built-in libraries. Another advantage of SWRLAPI which is absent in popular ontology editors such as Protégé is the provision of built in libraries for TBox and RBox ontologies using SQWRL to query all the OWL axiom types of an ontology [SWR18b].

Continuing the monotonicity argument in artificial intelligence, the GeSCo executes a PO only on a copy of the ontology, and the updated ontology is discarded by GeSCo after it is transmitted to the MES at the end of PO. A reuse of the ontology files that are employed for previous PO execution introduces possible duplicate assertions in which case the ontology is rendered inconsistent by an OWL reasoner. Even within a same PO, there is a need to change the values, for example, updating the current co-ordinates of the pick and place robot. These values are programmatically removed, and new positions of the robot destination are added invoking the *readActualPosition* method of its OPC UA server (refer Figure 6.9 marked in

blue).

However, both SWRL and SQWRL do not support deep and nested querying features that are required to compare varying ontology entities and their corresponding property values. For instance, there are three different assembly resources that provide similar *press* manufacturing services in the demonstrator. A rule needs to be constructed that determines the most suitable press resource based on the least effort in terms of distance to fetch different raw materials to the press assembly taking into account various intermediate quality assurance station visits. This rule involves the creation of nested conditions to compare the efforts for the three assembly resources which cannot be achieved with a single SQWRL query. In such cases, the rules are written as SPARQL queries. SPARQL is an RDF query language

```
PREFIX : <http://emea.global.corp.sap/rescom#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?PressInstance
    ((abs(?Press_XCoOrd_Value - ?LowerShellWarehouse_XCoOrd_Value))
    +(abs(?Press_XCoOrd_Value - ?UpperShellWarehouse_XCoOrd_Value))
    +(abs(?Press_XCoOrd_Value - ?CircuitBoardWarehouse_XCoOrd_Value))
    +(abs(?Press_XCoOrd_Value - ?Camera_XCoOrd_Value))
    +(abs(?Press_XCoOrd_Value - ?RFIDReader_XCoOrd_Value)) as ?CumulativeDistance_X_Direction)
    WHERE{
      ?LowerShellWarehouseInstance :hasPositionX ?XCoOrd_LowerShell_Inst;
                                   rdf:type :rescom_resources_LowerShellProvider.
      ?XCoOrd_LowerShell_Inst :Value ?LowerShellWarehouse_XCoOrd_Value.

      ?UpperShellWarehouseInstance :hasPositionX ?XCoOrd_UpperShell_Inst;
                                   rdf:type :rescom_resources_UpperShellProvider.
      ?XCoOrd_UpperShell_Inst :Value ?UpperShellWarehouse_XCoOrd_Value.

      ?CircuitBoardWarehouseInstance :hasPositionX ?XCoOrd_CircuitBoard_Inst;
                                     rdf:type :rescom_resources_CircuitBoardProvider.
      ?XCoOrd_CircuitBoard_Inst :Value ?CircuitBoardWarehouse_XCoOrd_Value.

      ?IndustrialCameraInstance :hasPositionX ?XCoOrd_Camera_Inst;
                                rdf:type :rescom_resources_IndustrialCamera.
      ?XCoOrd_Camera_Inst :Value ?Camera_XCoOrd_Value.

      ?RFIDReaderInstance :hasPositionX ?XCoOrd_RFIDReader_Inst;
                          rdf:type :rescom_resources_RFIDReader.
      ?XCoOrd_RFIDReader_Inst :Value ?RFIDReader_XCoOrd_Value.
      {
        SELECT ?PressInstance ?Press_XCoOrd_Value
        WHERE{  ?PressInstance :hasPositionX ?XCoOrd_Press_Inst;
                               rdf:type :rescom_capability_abstracts_Assembler.
                ?XCoOrd_Press_Inst :Value ?Press_XCoOrd_Value.}
  }} ORDER BY ASC(?CumulativeDistance_X_Direction) LIMIT 1
```

Figure 6.4: SPARQL Query to find suitable Assembly Resource with Least Distance Coverage in X-Axis.

for diverse data sources stored natively as RDF, or viewed as RDF via middleware. If automatically generated OWL ontologies that describe the demonstrator resources and their corresponding manufacturing services is serialized into its primary exchange syntax of RDF/XML syntax, then SPARQL queries can be applied to obtain the required results. In such a case, the generated common semantic model ontology stored in RDF/XML format acts as RDF store for the SPARQL query. The SPARQL query that computes the least effort in terms of distance involved among the three assembly resources is shown in Figure 6.4. This query only computes the distances along X-axis. Similar queries are run to find distances along Y- and Z-axes. In implementation, these three rules corresponding to the X, Y and Z axes are merged into a single query, and it is not shown here owing to space constraints. At the end, the suitable assembly unit is chosen based on least distance coverage in all the three directions of motion. The advantage of writing such simple and small to moderate sized rules is avoidance of writing complex source code in a high level language such as Java to realize the same objective. Furthermore, such rules neither presume any knowledge of presence of specific resources on the shop-floor nor require any hard-coded values for computations/validations. Similar queries can be formulated on a variety of other constraints such as least energy consumption, feasibility analysis in terms of resource capacity and its queue length, and least time duration to manufacturing service provision. An aspect that must be taken into consideration is that the previously defined SWRL rules must be fired and the subsequently reasoned and inferred ontology must be saved preceding to running the queries in SPARQL language as it lacks the full stack reasoning engine. The SQWRL rules are not relevant here as they do not change the knowledge base. The SPARQL queries were executed programmatically using Apache Jena ARQ SPARQL 1.1 query engine, version 3.9.0 [ARQ18].

This step is followed by the formulation of pre- and post-conditions of services provided by the manufacturing resources, and business rulesets. The pre- and post-conditions of services provided by the manufacturing resources are hooked to the OWL-S ontology described in Section 5.5. The business rules are also hooked as pre- and/or post-conditions to the production sub-steps of the PO (refer Figure 6.5). During the production orchestration phase, these variables in the business rules of the PO (that are prefixed with '?') are replaced by concrete instance of resources and associated components that have been chosen during the manufacturing service discovery phase. As shown in Figure 6.3, different ontology concepts of the key module production process come from various java source code packages. A short name for a semantic annotation such as 'rescom_resources_IndustrialCamera' (here dots in package name are replaced by underscores as dots are not allowed in URI fragments) is sufficient to identify the concepts and it also fosters readability (see 'Semantic_Annotation' coloumn in Figure 6.5). For the sake of UI simplicity, the conditions are prefixed with either `PRE_COND` and `POST_COND`, and assigned to a single field.

Another aspect of SWRL business rules which are hooked to the production steps of a PO is that they characterize a dataflow program. They ensure that production steps do not enter *incorrect states* of production. If a production flow enters an incorrect state, the production must *self-heal* itself to the correct state before continuing to the next state of production. For example, when the quality assurance production step at the camera station detects a black upper shell instead of a white upper shell, the robot places the black upper shell back to its original co-ordinates. The GeSCo invokes the *getCoOrdinates* method on the lower shell warehouse OPC UA server, and chooses the second item from the co-ordinates array of the response and repeats the process until it finds the white upper shell.

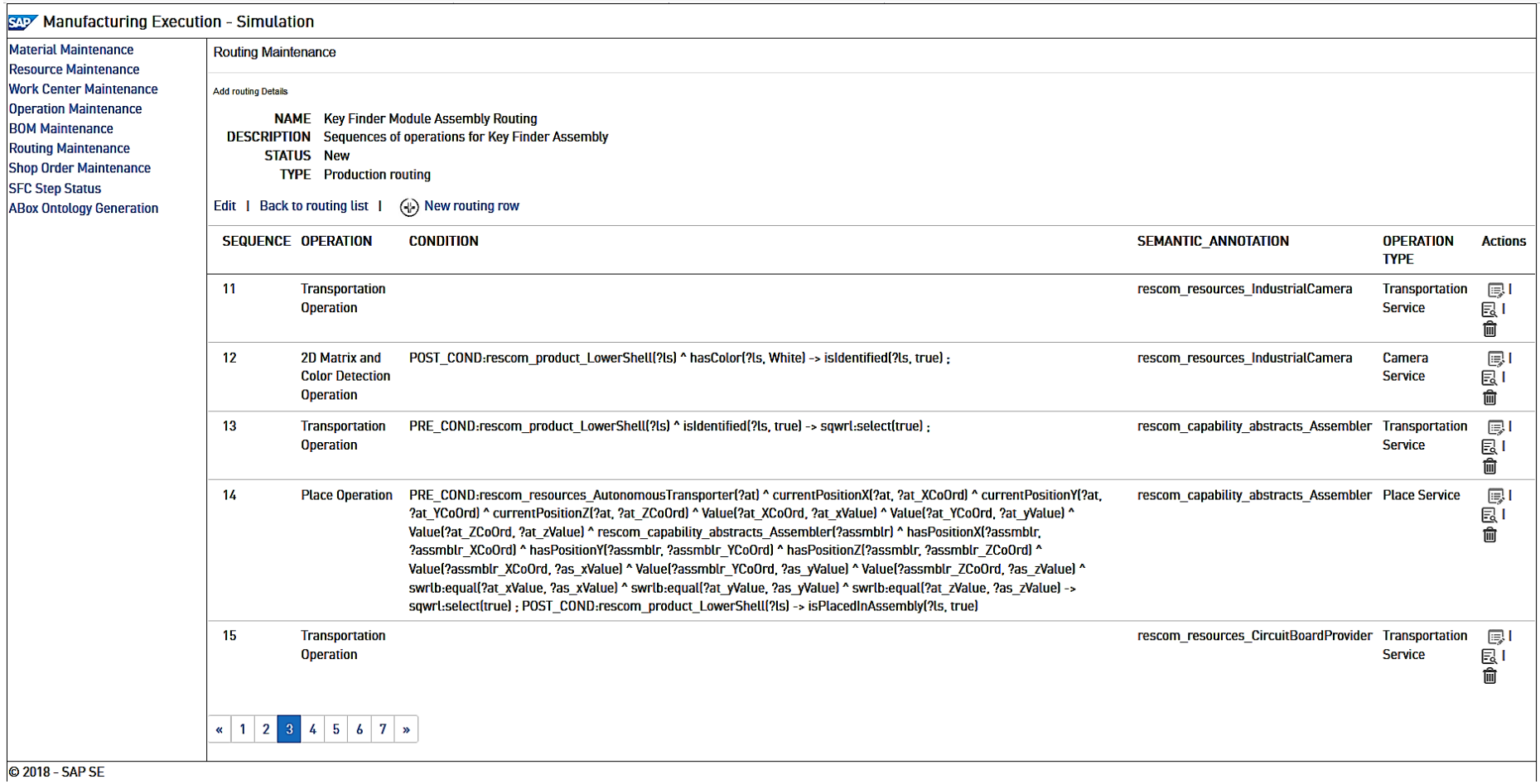

Figure 6.5: Partial Screenshot of Key Finder PO Generation UI.

6.3 Method Discovery and Orchestration

This thesis recommends implementing the hybrid semantic framework (refer Section 4.3) on the OPC UA server of manufacturing resources based on its advantages over its individual constituent approaches. However, it is up to the manufacturer to decide the best-fit approach based on the requirements and preference. This section only describes the implementation with regards to the hybrid semantic framework of OPC UA. From a general perspective of method discovery and orchestration, the procedure described here can also be roughly extrapolated to the other presented semantic frameworks.

The formal definitions of web service discovery concepts, and subsequent method composition are stated as following. The set of production ontology O generated from various MES software modules is defined as:

$O = O_1 \cup O_1 \cup ..... \cup O_n$

Manufacturing services offered by OPC UA methods are described employing O using any of the three semantic matchmaking methods described in Chapter 4. These methods are published to MDR M.

$M = \{m_1, m_2, ..., m_n\}$

In general, each of the methods are described as a tuple of assertions A referring to the concepts C of ontologies O.

$m_i = \{(a_1, c_{i1}), (a_2, c_{i2}), ..., (a_n, c_{in}) \ni c_{i1} \cup c_{i2} \cup .... \cup c_{in} \subseteq O\}$

More concretely, each of the m_i in the context of three semantic matchmaking methods are described as a 5-tuple consisting of its method description, input, output, and initial and final status flags as Pre- and Post-conditions as follows:

$m_i =< Met\_Desc, In_m, Out_m, Pre_m, Post_m >$

where Met_Desc represents a semantic annotation corresponding to the manufacturing service provided by the method. This semantic annotation can also be enhanced to represent additional qualities such as purpose, functional category, quality properties and any other non-functional attributes of the method.

In the same manner, the abstract manufacturing service requirements of PO are also translated to the corresponding method request template r referring to the same ontologies O for runtime method discovery.

$r_i =< Req\_Desc, In_r, Out_r, Pre_r, Post_r >$

The discovery mechanism then employs the reasoning techniques of description logics to

select the methods from M that share the concepts of r_i. For a specific r_i, a suitable m_i is found from the MDR if and only if following holds:

$(Req\_Desc \doteq Met\_Desc \vee Met\_Desc \subseteq Req\_Desc) \wedge \forall input_m.In_m \exists input_r.In_r \ni input_m \doteq input_r \cup input_r \subseteq input_m \wedge \forall output_r.Out_r \exists output_m.Out_m \ni output_r \doteq output_m \cup output_m \subseteq output_r$

In a simple scenario, the method discovery mechanism for a given abstract manufacturing demand is the subsumption check against the concrete manufacturing service supplies of the production shop-floor (refer Figure 5.4). However, if the discovery mechanism by the OWL reasoner returns more than one OPC UA method for a specific manufacturing service demand, the filtered methods should be subjected to additional ranking process to choose the most optimal method of manufacturing resources. In such cases, the resource whose manufacturing service category is nearest in class hierarchy to the PO demand is chosen. In other words, if req is the abstract manufacturing service requirement, and $(M^{'}, \subseteq)$ is a partially ordered set of discovered methods based on class hierarchy including req, then such a set is described as follows:

$M^{'} = \{m_1^{'}, m_2^{'}, ..., m_n^{'}, req\}$

In such case, the manufacturing service $m_n^{'}$ which is nearest subclass of req is chosen, i.e., $m_n^{'} \subseteq req \ni \nexists m_{n-1}^{'} \in M^{'} : m_n^{'} \subseteq m_{n-1}^{'} \subseteq req$

Practically, this can be realized by writing a simple SPARQL SELECT statement with the help of *rdfs:subClassOf* construct. In case of a tie, either a manufacturing resource is chosen randomly, or a further filtering takes place.

The third use case is the selection of suitable manufacturing service based on the custom business logic residing in cloud MES. In this scenario, a preknowledge of multiple resources that provide similar manufacturing services is acknowledged, and a corresponding filtering logic is replicated into the SPARQL SELECT queries as described in Figure 6.4.

Suppose $Pre(m)$ and $Post(m)$ are pre- and post-conditions of a method respectively represented as formulae in disjunctive normal form, $In(m)$ and $Out(m)$ are inputs and outputs of a method respectively, and $m_i^{'}$ and $m_j^{'}$ are two methods selected in the method discovery process. The thesis implementation makes following convention for the process of method composition of $m_i^{'}$ and $m_j^{'}$:

$m_i^{'} \prec m_j^{'} \ni \forall out(m_i^{'}).Out(m_i^{'}) \exists in(m_j^{'}).In(m_j^{'}) \ni out(m_i^{'}) \doteq in(m_j^{'}) \cup out(m_i^{'}) \subseteq in(m_j^{'}) \wedge \forall pre(m_j^{'}) \in Pre(m_j^{'}) \exists post(m_i^{'}) \in Post(m_i^{'}) \ni post(m_i^{'}) \implies pre(m_j^{'}) = true$

The above condition which formalizes causal link using the respective functional parameters allows the chaining of methods $m_i^{'}$ and $m_j^{'}$. The theory mentioned in this section has

been incorporated in intelligent key-finder demonstrator in the context of OPC UA method discovery and subsequent flexible orchestration processes of these discovered methods.

The manufacturing and status methods of the OPC UA servers of manufacturing resources were implemented under their respective base nodes as illustrated in Section 4.2.1. The GeSCo acts as method discovery repository, OPC UA client and OPC UA server for the shop-floor communication. The UI of the OPC UA servers of manufacturing resources and GeSCo was designed such that the OWL-S profile and process model sub-ontologies can be annotated to the *Description* fields of the manufacturing and status method base node, and the corresponding method nodes respectively (refer Figure 4.9). In order to automate this annotation process, a configuration file was created that mapped the method base nodes and the application methods to their corresponding OWL-S ontologies. During the start of the OPC UA servers, this configuration file was referred in order to hook the semantic annotations to the respective nodes. When the method model concept related to a method is further dereferenced against a Common Semantic Model (CSM), additional semantic attributes of the method can be discovered such as method category, and pre- and post-conditions. The OPC UA client (GeSCo Resource Perception Layer) queries these semantic annotations of application specific methods against the CSM to arrive at decisions of assigning suitable resources for the manufacturing operation tasks.

The OPC UA servers of manufacturing resources including GeSCo register themselves by publishing server metadata containing application methods base nodes to the GeSCo resource perception layer when they go online. With the help of the published server metadata, the GeSCo, which now acts as OPC UA client, semantically queries all the application specific methods of the registered manufacturing resources in order to discover the necessary methods and stores this information regarding the method functionalities in the method

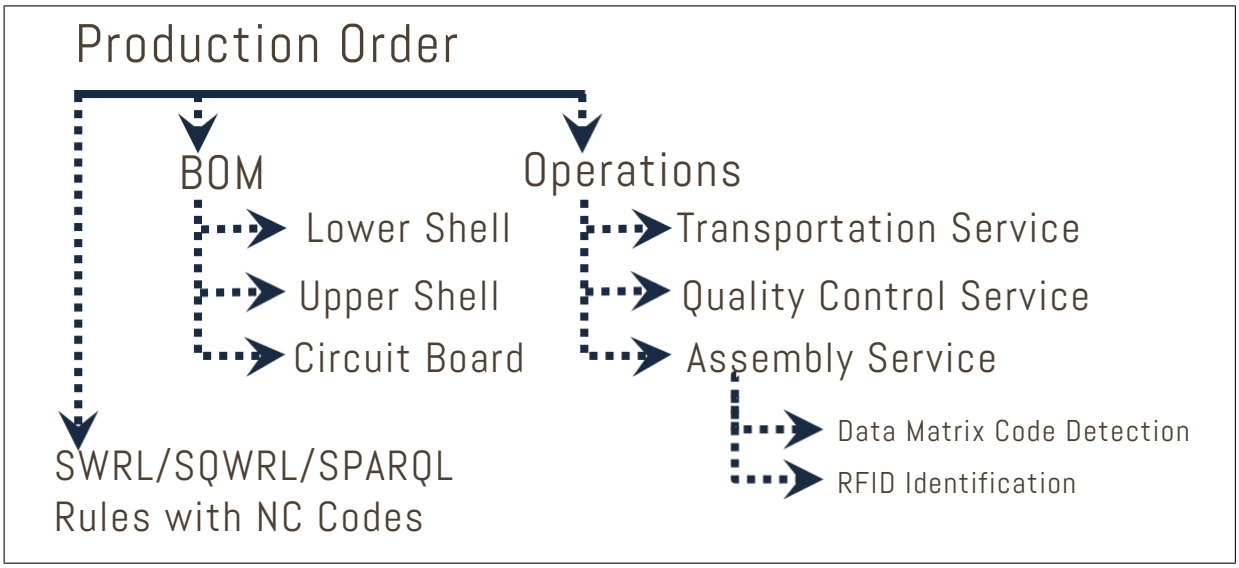

Figure 6.6: Contents of PO.

Table 6.1: Illustration of methods offered by resources

Resource	**Manufacturing Methods**	**Status Methods**
All three Press units	prepare	isInUse
	assembly	isInUseEvent
Pick and Place Robot for all three degrees of freedom	home	readActualPosition
	jogMove	
	jogMoveStop	
	moveAbsolute	
	stop	

discovery repository component of GeSCo. The method discovery repository stores these functionalities in a key-value pair collection for the purpose of fast retrieval. The manufacturing resources also refer to the method discovery repository in order to find suitable methods to communicate with GeSCo.

The PO for the production of an intelligent key finder is generated in the cloud MES. It is basically an ordered list of abstract requirements to produce the product. For the evaluation use case at hand, it lists the Bill of Materials (BoM), the abstract resource which has at least three degrees of freedom that organizes the movement of sub-components from warehouse to the final assembly, the set of quality control operations, and the final assembly operation of component materials. The quality control operations are data matrix code detection on the circuit board and the RFID identification of the upper shell of the key finder (refer Figure 6.6 for a brief description of the PO).

The resources of the key finder demonstrator offer various manufacturing and status methods. The result of the status methods along with the other contextual conditions is taken into account during formulation of the preconditions for manufacturing methods. For the purpose of illustration, Table 6.1 lists the methods offered by the *press* and *pick and place robot*. The PO also contains possible non-conformance codes that might arise in production and a corresponding list of abstract services to resolve the non-conformance issue. When the dispatcher in the cloud MES dispatches the PO to the GeSCo cache, the production engine in GeSCo requests GeSCo decentralization facilitator to find the most suitable manufacturing resource to each of its abstract PO requirements. The decentralization facilitator parses the semantic concepts related to resource methods from its method discovery repository, and

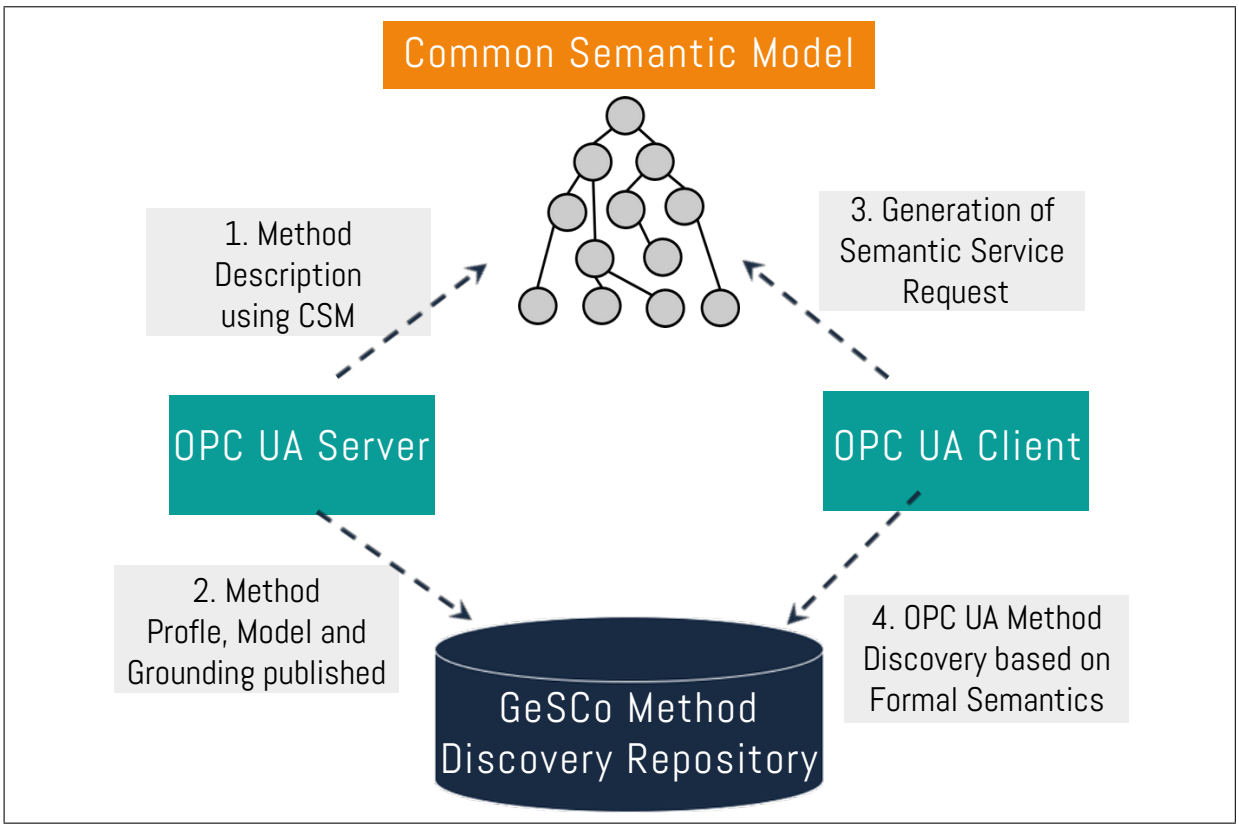

Figure 6.7: Ontology Driven OPC UA Method Creation, Discovery and Invocation employing OWL-S Framework.

matches the required services of the PO to the capabilities offered by resources on the shop-floor with the aid of the centrally accessible ontology service. The semantic method creation, discovery and invocation is illustrated in Figure 6.7.

In implementation, the decentralization facilitator analyzes the method discovery repository for the manufacturing services that offer the sub-components of the BoM, transportation service for picking, moving and placing the raw materials, and quality control and assembly services. The aid of formal representation in the form of OWL-S ontologies hooked to the OPC UA nodes and their subsequent reasoning capabilities enables the decentralization facilitator component of GeSCo to automatically select the available manufacturing services at production runtime. When the production engine receives the chosen manufacturing services it combines them to create an adaptive orchestration plan. GeSCo also takes into the account the necessary pre- and post-conditions of a method to hold true for continuation of the planned orchestration. For example, in the key-finder use case, the pre-condition for placing the circuit board onto the assembly unit is the detection of its data matrix code. Similarly, the upper shell has to be recognized by the RFID reader before it is transported to the assembly unit. As the OWLAPI is available only for the Java stack, a custom Java application is implemented that loads, queries, creates, updates, and saves the ontology. It is also capable of reasoning and adding restriction on the entities of the ontology. At the end of

each step, the production engine inspects for possible non-conformance logs against the PO, and also invokes the Java application to reason about the context of method execution by asserting the class axioms and custom rulesets. The decoupling of manufacturing resources from manufacturing operations during planning in combination with the proposed semantically enriched OPC UA information model effectively addresses the problems of quality non-conformance (refer use case 1 in Section 6.3.1) and resource breakdowns (refer use case 2 in Section 6.3.2).

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

6.3.1 Use case 1: Quality Control

A study was carried out where a quality issue was logged with regard to the *assembly* operation with the corresponding non-conformance code. During the orchestration plan creation step, GeSCo chooses the electric press as first choice for the final assembly operation of the PO because the pick-place-robot can transport all the sub-components of the key finder to the electric press mounting area by covering the smallest distance in relation to other presses (refer to the Figure 6.2). GeSCo arrives at this decision based on the evaluation of a SWRL rule for selecting the mounting assembly (refer Figure 6.4). The algorithm also takes into consideration the distance covered by the pick-and-place robot to perform the quality control operations of the sub-components.

In this use case, the electric assembly unit is issued a command to *press* after the verification of pre-conditions. The electric assembly does not move down to press the sub-assembly although it does not generate an HTTP 5XX server error. Instead, it returns status code *HTTP 200 OK* which corresponds to successful execution of the *press* operation. These HTTP response status codes are originated in the encapsulated web servers of the demonstrator modules, and propagated to the outer OPC UA server. A manual worker who oversees this production step observes the defective electric assembly unit, and logs a corresponding Non-Conformance (NC) code against the PO. However, the production work-flow of the evaluation landscape does not have the option for manual logging of defects. Hence, the production flow was intercepted through an external application, and the defect is injected into the production execution. At this point, the production engine realizes the defect in the *assemble* step, and it retrieves the abstract services related to the non-conformance code required to resolve the quality issue from PO cache and sends it to the decentralization facilitator to find the relevant manufacturing resources. The GeSCo production engine creates

a new orchestration plan containing only those operations relevant to the logged defect. It searches in the method discovery repository for the equivalent method providers. Based on the second choice of the SPARQL query that returns the optimal *press* resource, GeSCo concludes that the plug and play pneumatic press also presents a set of methods that provide the equivalent manufacturing services. Finally, the production step is executed via this pneumatic press situated on the right part of the demonstrator.

6.3.2 Use case 2: Resource Breakdown

The resource breakdown scenario was also simulated in another PO where the *Pneumatic Press* resource which is part of the PO orchestration plan was rendered *unresponsive* with the HTTP 500 internal server error status code. The specific PO was put on hold until a new resource that provided the same manufacturing service was plugged in to the production landscape and its capability metadata is published to the GeSCo. A new pneumatic press resource is plugged in (rightmost press resource in Figure 6.2). GeSCo checks in its production engine queue for the pending status/new PO that relies on the newly installed resource, and accordingly changes the orchestration plan.

The decision making process is entirely localized to the shop-floor as GeSCo draws its conclusions based on the evaluation results of the SWRL/SQWRL/SPARQL rulesets. Thus, it proves that with the aid of a formal ontology and subsequent semantic OPC UA framework, the edge component of a cloud based MES can make adaptive orchestration plans at runtime in the factory shop-floor even in the event of exceptional scenarios.

6.4 Open Integrated Factory - Generation 2017: Supplementary Use Case

The research concept of extending the functional modules which are related to the runtime production activities to the production edge layer, and subsequent caching of the PO to the Production Control Data Cache component of GeSCo was also implemented in the Open Integrated Factory - Generation 2017 (see Figure 6.8) that SAP along with other technology partners showcased in Hannover Industrial Fair 2017. This approach significantly reduced the network latency during the PO execution as GeSCo only communicated with manufacturing resources within the local area network. The average network latency between the GeSCo and the manufacturing resource was in the range of 10-20 milliseconds. However, the idea of semantic frameworks extended to application specific methods of OPC UA was not implemented on this demonstrator. The complete PO was cached to the GeSCo with a concrete mapping of manufacturing resources. The orchestration plan of the PO was created

Figure 6.8: Open Integrated Factory - Generation 2017.

in the cloud MES as described in Section 1.1. A guidance from the cloud MES is sought by GeSCo only in the event of a deviation from the production plan.

6.5 Implementation Considerations, Challenges, Results and Lessons Learned

- The extraction of knowledge and its representation is already a difficult proposition. In the manufacturing plant, this challenge is amplified owing to the complexity and the varied nature of the resources, work stations, raw materials, part assemblies and products. To add to this complexity, the manufacturing plant evolves continuously during its lifecycle due to the installation and operation of new resources, removal of a dilapidated resource and introduction of new product variant. This calls for continuous updates of the knowledge in the shop-floor. Any knowledge extraction solutions should also be resilient to unscheduled changes of the plant.
- SOAP based web services and subsequently, SWS are deemed to be heavy-weight service descriptions after the advent of REST based web services in the last decade. Hence, SWS could not be widely adopted and consequently, no large scale experimental results are available to ascertain on their scalability on the internet level. However, for bounded usage, the semantic description of communication protocols for the purpose of intelligent discovery and invocation is scalable for new requirements, for example inter-enterprise operations. This argument also justifies enriching the OPC UA with

Table 6.2: Comparison of UTF-8 Encoded Data Transfer between OPC UA Server and GMDR in SemOPC UA and Hybrid Methodologies

	SemOPC UA Approach	**Hybrid Approach**
Average Size of the Published Data per Method	5250 Bytes	510 Bytes
Assumed Number of Methods per Resource	10	
Average Size of the Published Data per Resource	52500 Bytes	510 Bytes

semantics for communication within the manufacturing plant.

- Taking into account that this work is a first attempt at enriching OPC UA with semantics, there is no tool to automatically create a modified OWL-S sub-ontologies template that corresponds to a method in OPC UA. Nor is there a tool to attach the modeled ontology to the OPC UA nodes in case of the SA-OPC UA and hybrid schemes. The author only developed a utility software that serves the purpose. For industrial adoption, a sophisticated solution is required that is infalliable, mature, and hides the complexity involved in developing semantic descriptions.
- All three semantic frameworks of OPC UA presented in Chapter 4 were evaluated on the key finder demonstrator. Each of the approaches generates data traffic in order to publish the metadata of the application methods offered by the manufacturing resources. The size of the metadata in case of SA-OPC UA and hybrid approach is the same as both approaches follow the same concept of server grounding. However, the hybrid approach shows a significant decrease in the data traffic in comparison to the SemOPC UA server, and hence, trumps the SemOPC UA approach in this regard. The SemOPC UA server publishes the entire OWL-S ontology of each of the offered methods. In addition, the semantic OPC UA server based on the OWL-S framework sends method grounding data that varies directly with deeper node browse paths and the number of supported methods. Such method grounding metadata publishing of all the OPC UA servers of the shop-floor adds the additional burden on the network bandwidth. On the contrary, the hybrid approach publishes constant size and one-time metadata of its application method nodes of the OPC UA server to the GeSCo method discovery repository irrespective of the number of supported methods. Table 6.2 provides an estimate of the amount of UTF-8 encoded data transferred from an OPC UA server to the method discovery repository for both the approaches.

- During experimental evaluation, it was observed that ontology processing does not scale well with large ontologies. Furthermore, the average response time of the rule validation services was one second. Specifically, it was also observed that the creation of a new instances of an OWL ontology manager and an OWL reasoner, creation of SWRL, SQWRL and SPARQL query engine instances, and loading the ontology documents from the file system take an inordinate amount of time. In production runtime, this impacted the resource discovery and rule validations negatively in terms of processing times. Therefore, REST based web services that provide OWL reasoning and inference services, and execute SWRL/SQWRL/SPARQL rules/queries were implemented using a singleton software design pattern in a way it does not affect the concurrency of the whole system. This permitted in-memory computation of ontology processing, querying and rule validations, and consequently brought down the average response time of these web services to under 100 ms. The research use case on the key finder demonstrator does not contain a large number of parallel processes. Even for a production scenario involving parallel processes, the waiting times due to a small queue of requests to these singleton OWL reasoners and rule engines outweigh the drawbacks of large response times owing to creation of these instances on *per service call* basis. Furthermore, this substantial reduction of network latency is an additional benefit over the ability of GeSCo to react to the local conditions on the shop-floor using formal reasoning which allows it to function without the explicit guidance from cloud based MES.
- Another aspect that needs to be kept in mind is that the transformation from source code to OWL ontology is industry and product specific. Furthermore, the rules and queries that are formulated in the later steps rely heavily on the nitty-gritties of the generated OWL ontologies. This could be a challenge to general applicability. However, a stable OWL ontology is guaranteed when the transformation algorithm is run on a baseline source code that undergoes limited changes. Moreover, the generated OWL ontology, and associated business and production rulesets can be reused for a particular product variant.
- Although the presented rule-based approach covers most business scenarios, it was also discovered during evaluation that the semantic web languages fail in the processing of large amounts of data and complex calculations. However, in retrospect, the semantic web languages were essentially conceptualized with only formal knowledge representation in mind. The research use-case has moderate data processing in the form of identification of colors of upper and lower shells of the key finder module, and identifying a 2D barcode on the circuit board which is sandwiched between these upper and lower shells. The corresponding image processing source code was compiled and the resulting 6kb dynamic linked library (DLL) was cached along with the PO to GeSCo. The GeSCo is designed such that custom process logic that inherits its API classes can

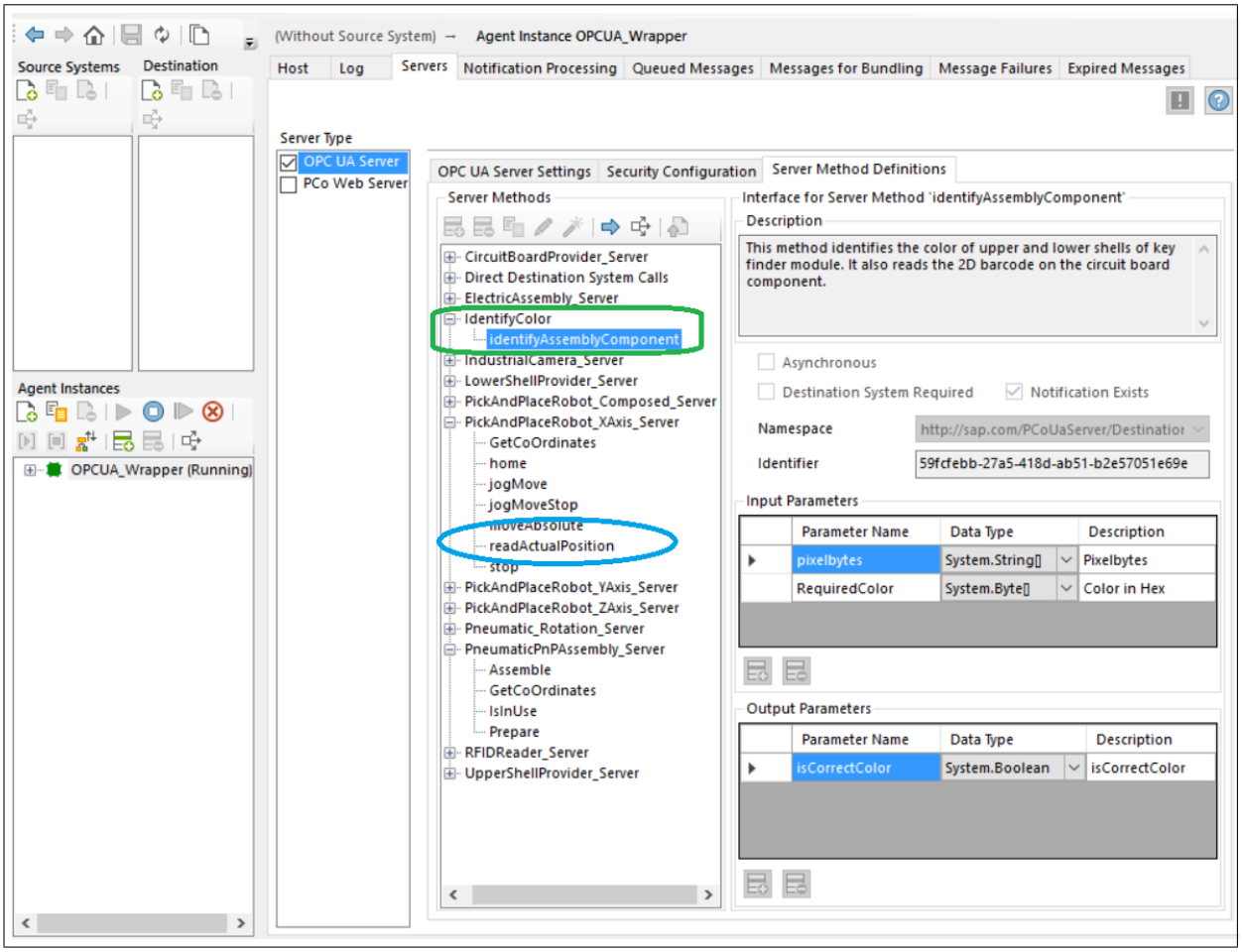

Figure 6.9: OPC UA Wrapper for HTTP based ResCom servers, and Caching of source code for validations involving large data processing (Highlighted in Green Color).

be attached to it as either OPC UA, SOAP, REST or ODATA based WS-* servers (refer Figure 6.9). Alternatively, when the source code size for the production step validation is too large to cache in the production network, it is inevitable to request the cloud-based business applications such as MES for rule validations and determination of the next step of production.

7 Discussion: Summary and Outlook

The final chapter of the thesis summarizes the research work concisely, and lays down some plans for future work.

Decentralization in production control is an important aspect of manufacturing in today's world. In line with this objective, even as master control information resides in a cloud based MES, reaction to local conditions on the factory shop-floor with a negligible amount of network latency is another requirement of present and future digital factories. The first approach of this thesis was the distribution of manufacturing responsibilities, which involved delegation of production runtime tasks to the edge component in order to readily interact with the physical world. The edge component brings memory and computing physically closer to the location where it is needed in order to improve the quality of service in a cloud computing paradigm.

The touchpoints of the thesis include manufacturing activities ranging from production design to production planning, execution and control. Although these three stages of manufacturing are distinct in theory, the boundaries of each of these activities come into contact with each other in reality. In order to achieve higher efficiency, integration of these stages is essential. Ontologies make it possible by making the data interoperable across different stages and modules.

With an intention to make data machine processable, and achieve precise and unambiguous communication, RAMI4.0, which helps in identifying relevant standards to realize the concept of digital factory stresses the importance of semantics. Accordingly, it defines informal semantics for application-agnostic services, but admits a lacuna in the definition of the application methods of AAS. Though it acknowledges and recommends the importance of formal ontologies [Econ17], it has not explicitly mandated its usage. To this end, with the objective to address the difficulties that stem from differences in the meaning and usage of manufacturing vocabulary, and subsequent inability to express semantic information about manufacturing services offered by the OPC UA resources, and automatic discovery and orchestration by software agents, a scientific methodology is formulated that serves the following purposes:

- Pragmatic automation of formal domain knowledge creation exploiting the source code

of MES modules

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

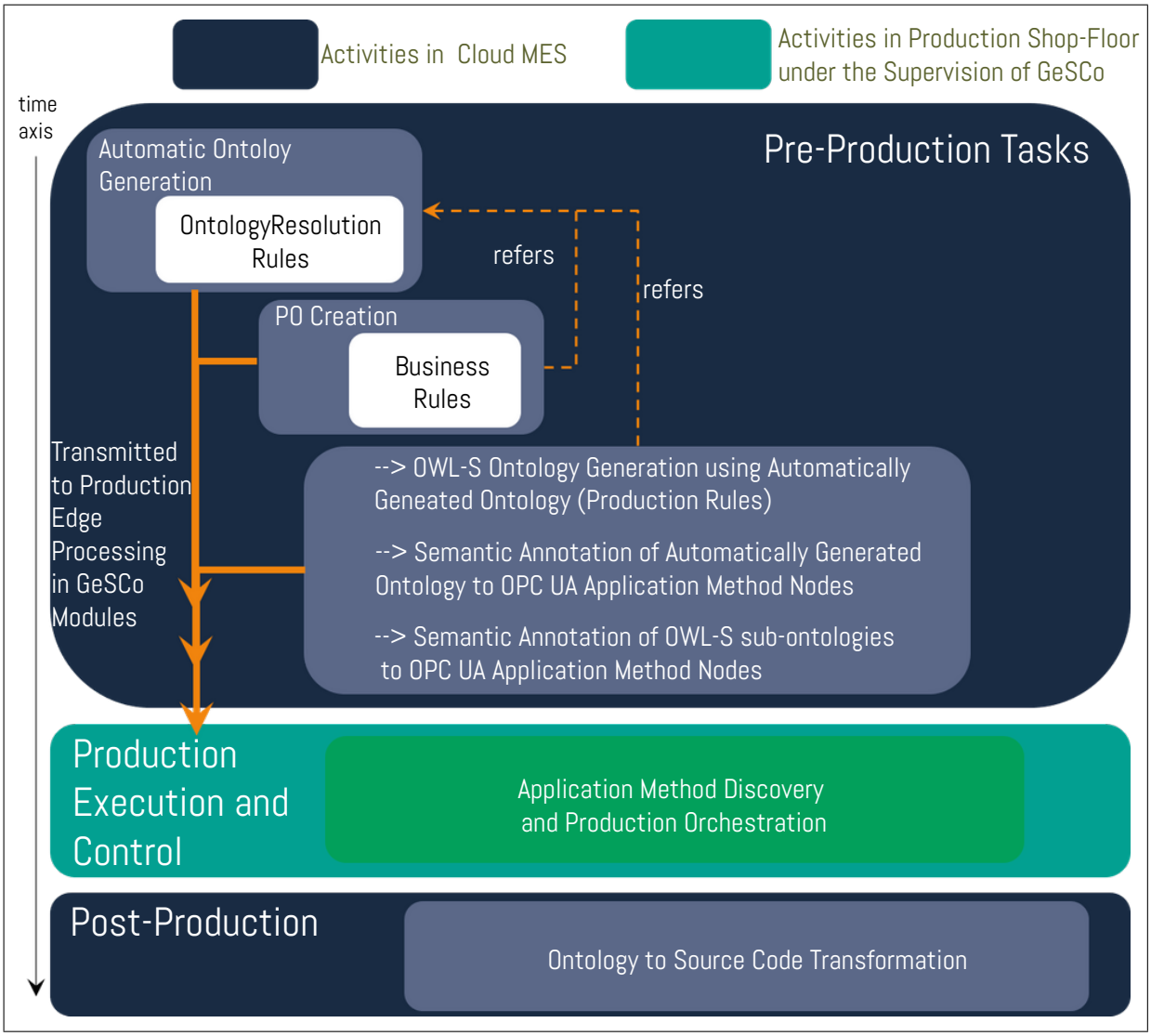

Figure 7.1: Contributions of Thesis in Chronological Order.

Contributions of the Thesis In order to address the requirements at hand, the thesis presents an aligned three step solution proposal, which are listed in the following (refer Figure 7.1):

1. Partial extension of functional modules of the cloud MES to the edge of factory shop-floor, collectively called Generic Shop-Floor Connector (GeSCo), so that control data temporarily resided, processed, and updated within the local area network
2. Due to inherent inadequacy of OPC UA to formally describe the data in its information model, introduction of formal semantic OPC UA framework to facilitate the production execution and control for autonomous discovery, adaptive orchestration and subsequent invocation of its application-specific methods taking into account the contextual information in manufacturing
3. In order to avoid the ad-hoc and time-consuming process of creation of formal ontologies silos by different departments of manufacturing, and the corresponding ontology alignment for potential mismatches, the thesis also presents a methodology of semi-automatic ontology-encoding from the central cloud based MES. As cloud MES has overarching visibility, the resulting generated homogeneous ontology can be consumed directly in order to model the data by GeSCo and manufacturing resources. To this end, the thesis evaluation also allows the semi-automatic creation of an OWL-S ontology for OPC UA application methods employing the OWLAPI.

With regards to item 2 in the above-mentioned enumeration, the thesis adopts the popular semantic web services frameworks and specifications of OWL-S and SAWSDL, respectively, to the application methods of OPC UA. Later, a feasible hybrid approach which is a composition of the adopted OWL-S and SAWSDL frameworks that derives the benefits, and at the same time, precludes the shortcomings of both approaches is introduced. This thesis recommends implementing this hybrid semantic framework on the OPC UA server of manufacturing resources due to its advantages over its individual constituent approaches. However, the common rationale behind the conception and consequent adoption of these various semantic web services frameworks to the OPC UA information model is the automatic discovery, orchestration and invocation of web services. Therefore, the decision to adopt a specific semantic OPC UA framework is up to the manufacturer, who determines the best-fit approach based on the requirements and preference. With the view on industry acceptance, an intentional lightweight *SA-OPC UA* approach has been proposed to smoothen the learning curve for early adopters of formal semantics on OPC UA application methods. To that end, one of the important goals of the thesis is to prove the extension of the semantic web services concept, general applicability and implementation feasibility to OPC UA application methods, and draw further research interest in this direction. Therefore, it does not experiment with each of the features of these semantic web service frameworks.

In relation to item 3, the automatic ontology-encoding process involves the translation of concepts from the software engineering domain to ontology engineering domain which leads to inevitable loss of data. The thesis proposes to overcome this inefficacy of the translation process due to the nature of ontology construction languages such as OWL with the addition of imperative knowledge provided by formal rule based languages such as SWRL, SQWRL and SPARQL.

With the above-mentioned solution, the entire PO can be delegated to the GeSCo, and consequently, GeSCo can handle both normal and exceptional scenarios in manufacturing. Even though the design was conceptualized keeping in mind multiple PO executions, it was not experimentally validated due to the logistic limitations of the key-finder demonstrator modules to conduct parallel implementations.

As an acknowledgement to this thesis, the OPC Foundation has begun the process of adding support for OWL semantic descriptions as amendments to the existing specification through an OPC UA working group, tentatively named "Automatic Validation of OPC UA Information Model". Global names, dictionary references, method metadata, interfaces and other addins are being introduced as initial development. The author is also a member of that working group.

Future Work

In the discussion of semantic web services, the important frameworks are OWL-S, SAWSDL and WSMO. The work in this thesis focuses only on the first two frameworks due to their strong orientation towards OWL, which is a W3C recommendation and part of the semantic web technology stack. WSMO introduces a formal language, called Web Service Modeling Language (WSML), for the specification of different aspects of semantic web services. Due to the poor OWL integration of WSMO, it did not fit into the thesis landscape along with the other concepts. Nonetheless, it would be an interesting research prospect to apply the WSMO framework for discovery, composition and invocation of the OPC UA application methods.

The thesis considers semantic annotation of the nodes of the OPC UA information model only from the perspective of the automatic application method discovery and composition. However, with the growing number of OPC UA collaborations, it becomes increasingly difficult to check for the *correctness* of every such collaboration document which spans several hundreds to thousands of pages. In such a situation, the formal semantic annotation of the complete information model of OPC UA enables automatic detection of anomalies and logical inconsistencies in such a *collaboration* design of OPC UA servers. Another interesting research in this direction would be the development of custom rulesets in the form of SWRL

or SQWRL rules which when executed provide guidance regarding the custom behaviors in design in addition to *correctness*. Such rules and rule based languages dramatically increase the possibilities.

The thesis only concentrates on the *current* data at hand in the factory shop-floor. However from the cloud infrastructure perspective, it did not experiment with the building of sophisticated statistical models in the centralized MES and training based on past data to gain insights into the nature of production. For example, computational algorithms which rely on repeated random sampling understand all the possible outcomes of decisions, assess the associated impacts and make optimal decisions under uncertain conditions. Such prediction models take into account uncertainty, and hence, prevent/reduce the adaptive measures taken in the production edge components in case of exceptional situations.

Finally, there is some scope for future work from the standpoint of implementations and tools around ontologies for the OPC UA information model. The research assigns a functional sub-module to store the metadata published by the manufacturing resources in a method discovery repository. However, this is only an abstract concept in the thesis, and its implementation is also ad-hoc in the context of the realization of the evaluation landscape. The detailed and scientific structure of such a published metadata storage is interesting in the absence of standards such as WSDL documents. Since it is a first attempt at annotating OWL reference concepts to OPC UA methods and associated nodes, there exist no tools to automate this process. Also from a OWL-S point of view, there is a need for a tool that automatically generates OWL-S ontologies for a specific application method of a remotely situated OPC UA server, for example from cloud infrastructure.

Bibliography

[ARQ18] *ARQ - A SPARQL Processor for Jena*. 2018. – URL: https://jena.apache.org/documentation/query/ [Date: 2018-11-28].

[Art09] *Artemis Industry Association*. 2009. – URL: https://artemis-ia.eu/ [Date: 2019-03-24].

[Emb14] *Embedded Multi-Core systems for Mixed Criticality applications in dynamic and changeable real-time environments*. 2014. – URL: https://www.artemis-emc2.eu/ [Date: 2019-03-24].

[Her18] *HermiT OWL Reasoner*. 2018. – URL: http://www.hermit-reasoner.com/ [Date: 2018-11-28].

[IMC14] *IMC-AESOP European Project*. 2014. – URL: http://imc-aesop.org/ [Date: 2019-03-24].

[OWL18a] *OWL 2 Web Ontology Language*. 2018. – URL: https://www.w3.org/TR/owl2-overview/ [Date: 2018-11-28].

[OWL18b] *OWL 2 Web Ontology Language Profiles, Second Edition*. 2018. – URL: https://www.w3.org/TR/owl2-profiles/ [Date: 2019-03-24].

[Pro19] *Protege-OWL API*. 2019. – URL: https://protegewiki.stanford.edu/wiki/ProtegeOWL_API_Programmers_Guide/ [Date: 2019-03-24].

[RES18] *RES-COM Project*. 2018. – URL: http://www.res-com-projekt.de [Accessed Date: 2018-08-15].

[Rul18] *RuleML Wiki*. 2018. – URL: http://wiki.ruleml.org/index.php/RuleML_Home/ [Date: 2018-11-28].

[SAP19a] *SAP Plant Connectivity*. 2019. – URL: https://help.sap.com/viewer/p/SAP_PLANT_CONNECTIVITY [Accessed Date: 2019-03-24].

[SAP19b] *SAP Plant Connectivity, Enhanced Method Processing*. 2019. – URL: https://help.sap.com/doc/saphelp_mii151sp03/15.1.3/en-US/75/819a57c209ab6be10000000a4450e5/frameset.htm [Accessed Date: 2019-03-24].

[Sem18] *Semantic Query-Enhanced Web Rule Language*. 2018. – URL: https://github.com/protegeproject/swrlapi/wiki/SQWRL/ [Date: 2018-11-28].

[SOA19] *SOAP UI*. 2019. – URL: https://www.soapui.org/ [Accessed Date: 2018-03-24].

[SOC09] *SOCRADES European Project*. 2009. – URL: http://www.socrades.net/ [Date: 2019-03-24].

[SPA18a] *SPARQL Inferencing Notation*. 2018. – URL: https://www.w3.org/Submission/spin-overview/ [Date: 2018-11-28].

[SPA18b] *SPARQL Protocol and RDF Query Language*. 2018. – URL: https://www.w3.org/TR/rdf-sparql-query/ [Date: 2018-11-28].

[SWR18a] *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. 2018. – URL: https://www.w3.org/Submission/SWRL/ [Date: 2018-11-28].

[SWR18b] *SWRL TBox BuiltIn Library*. 2018. – URL: https://github.com/protegeproject/swrlapi/wiki/SWRLTBoxBuiltInLibrary/ [Date: 2018-11-28].

[Abde17] Abdelaal, Alaa E.; Hegazy, Tamir ; Hefeeda, Mohamed: Event-based control as a cloud service. In: *American Control Conference (ACC), 2017* IEEE, 2017, S. 1017–1023.

[Adol15] Adolphs, Peter; Bedenbender, H; Dirzus, D; Ehlich, M; Epple, U; Hankel, M; Heidel, R; Hoffmeister, M; Huhle, H; Kärcher, B [u. a.]: Status Report-Reference Architecture Model Industrie 4.0 (RAMI4. 0). In: *VDI-Verein Deutscher Ingenieure eV and ZVEI-German Electrical and Electronic Manufacturers Association, Tech. Rep* (2015).

[Aker18] Akerman, Magnus: *Implementing Shop Floor IT for Industry 4.0*, Diss., 06 2018. – 2–2 S.

[Akht08] Akhtar, Waseem; Kopecký, Jacek; Krennwallner, Thomas ; Polleres, Axel: XSPARQL: Traveling between the XML and RDF worlds–and avoiding the XSLT pilgrimage. In: *European Semantic Web Conference* Springer, 2008, S. 432–447.

[Amer05] Ameri, Farhad; Dutta, Deba: Product lifecycle management: closing the knowledge loops. In: *Computer-Aided Design and Applications* 2 (2005), Nr. 5, S. 577–590.

[Amer12] Ameri, Farhad; Urbanovsky, Colin ; McArthur, Christian: A systematic

approach to developing ontologies for manufacturing service modeling. In: *Proc. 7th International Conference on Formal Ontology in Information Systems (FOIS 2012), Graz, Austria* Citeseer, 2012.

[Ande00] Anderson, Carl; Bartholdi, John J.: Centralized versus decentralized control in manufacturing: lessons from social insects. In: *Complexity and complex systems in industry* (2000), S. 92–105.

[Apos13] Apostu, Anca; Puican, Florina; Ularu, Geanina; Suciu, George; Todoran, Gyorgy [u. a.]: Study on advantages and disadvantages of Cloud Computing–the advantages of Telemetry Applications in the Cloud. In: *Recent Advances in Applied Computer Science and Digital Services* 2103 (2013).

[Arab17] Arab-Mansour, Ikbal; Millet, Pierre-Alain ; Botta-Genoulaz, Valérie: A business repository enrichment process: A case study for manufacturing execution systems. In: *Computers in Industry* 89 (2017), S. 13–22.

[Arbe11] Arbeitsgruppe MES, ZVEI; (Verlag), ZVEI (Hrsg.). *Manufacturing Execution Systems (MES), Market specific requirements and supplier independent solutions*. July 2011.

[Arch18] Architecture, OPC U.: Interoperability for Industrie 4.0 and the Internet of Things Industrie 4.0. In: *OPC Unified Architecture White Paper* (2018), June.

[Auer07] Auer, Sören; Bizer, Christian; Kobilarov, Georgi; Lehmann, Jens; Cyganiak, Richard ; Ives, Zachary: Dbpedia: A nucleus for a web of open data. In: *The semantic web*. Springer, 2007, S. 722–735.

[Auto17] working group in the Automation Division, ZVEI Manufacturing Execution S.: Position paper - Industrie 4.0: MES ? Prerequisite for Digital Operation and Production Management, Tasks and Future Requirements. In: *ZVEI-German Electrical and Electronic Manufacturers Association, Tech. Rep* 1 (2017), August, S. 25.

[Baad08] Baader, Franz; Horrocks, Ian ; Sattler, Ulrike: Description logics. In: *Foundations of Artificial Intelligence* 3 (2008), S. 135–179.

[Bail03] Bailey, David; Wright, Edwin: *Practical SCADA for industry*. Elsevier, 2003.

[Bake12] Baker, Thomas; Noy, Natasha ; Swick, Ralph; Herman, Ivan (Hrsg.): *Semantic Web Case Studies and Use Cases*. June 2012. – URL: `https://www.w3.org/2001/sw/sweo/public/UseCases/` [Date Accessed: 2019-03-24].

[Barb15] Barbosa, José; Leitão, Paulo; Adam, Emmanuel ; Trentesaux, Damien:

Dynamic self-organization in holonic multi-agent manufacturing systems: The ADACOR evolution. In: *Computers in Industry* 66 (2015), S. 99–111.

[Bary10] Baryannis, George; Plexousakis, Dimitris: Automated Web Service Composition: State of the Art and Research Challenges. In: *ICS-FORTH, Tech. Rep* 409 (2010).

[Batc12] Batchelor, Joy; Andersen, HENRIK R.: Bridging the product configuration gap between PLM and ERP?an automotive case study. In: *19th international product development management conference, Manchester*, 2012.

[Batt05] Battle, Steve; Bernstein, Abraham; Boley, Harold; Grosof, Benjamin; Gruninger, Michael; Hull, Richard; Kifer, Michael; Martin, David; McIlraith, Sheila; McGuinness, Deborah [u. a.]: Semantic web services framework (SWSF) overview. In: *W3C Member Submission* 9 (2005).

[Beac00] Beach, R; Muhlemann, AP; Price, DHR; Paterson, A ; Sharp, JA: The selection of information systems for production management: An evolving problem. In: *International Journal of Production Economics* 64 (2000), Nr. 1-3, S. 319–329.

[Beck15] Becker, Klaus; Frtunikj, Jelena; Felser, Meik; Fiege, Ludger; Buckl, Christian; Rothbauer, Stefan; Zhang, Licong ; Klein, Cornel: RACE RTE: a runtime environment for robust fault-tolerant vehicle functions. In: *CARS 2015-Critical Automotive applications: Robustness & Safety*, 2015.

[Bede17] Bedenbender, H; Bentkus, A; Epple, U; Hadlich, T; Heidel, R; Hillermeier, O; Hoffmeister, M; Huhle, H; Kiele-Dunsche, M; Koziolek, H [u. a.]: Industrie 4.0 plug-and-produce for adaptable factories: Example use case definition models and implementation. In: *Federal Ministry for Economic Affairs and Energy (BMWi), Tech. Rep.* (2017).

[Bern01] Berners-Lee, Tim; Hendler, James ; Lassila, Ora: The semantic web. In: *Scientific american* 284 (2001), Nr. 5, S. 34–43.

[Besc12] Beschi, Manuel; Dormido, Sebastián; Sanchez, José ; Visioli, Antonio: Characterization of symmetric send-on-delta PI controllers. In: *Journal of Process Control* 22 (2012), Nr. 10, S. 1930–1945.

[Bett13] Bettenhausen, Kurt D.; Kowalewski, Stefan: Cyber-physical systems: Chancen und Nutzen aus Sicht der Automation. In: *VDI/VDE-Gesellschaft Mess-und Automatisierungstechnik* (2013), S. 9–10.

[Bohn06] Bohn, Hendrik; Bobek, Andreas ; Golatowski, Frank: SIRENA-Service Infrastructure for Real-time Embedded Networked Devices: A service oriented

framework for different domains. In: *Networking, international conference on systems and international conference on mobile communications and learning technologies, 2006. ICN/ICONS/MCL 2006. International conference on* IEEE, 2006, S. 43–43.

[Bohr05] Bohring, Hannes; Auer, Sören [u. a.]: Mapping XML to OWL Ontologies. In: *Leipziger Informatik-Tage* 72 (2005), S. 147–156.

[Borg04] Borgo, Stefano; Leitão, Paulo: The role of foundational ontologies in manufacturing domain applications. In: *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* Springer, 2004, S. 670–688.

[Box00] Box, Don; Ehnebuske, David; Kakivaya, Gopal; Layman, Andrew; Mendelsohn, Noah; Nielsen, Henrik F.; Thatte, Satish ; Winer, Dave. *Simple object access protocol (SOAP) 1.1.* 2000.

[Brat07] Bratt, Steve: Semantic web and other W3C technologies to watch. In: *Talks at W3C, January* (2007).

[Bray97] Bray, Tim; Paoli, Jean; Sperberg-McQueen, C M.; Maler, Eve ; Yergeau, François: Extensible markup language (XML). In: *World Wide Web Journal* 2 (1997), Nr. 4, S. 27–66.

[Bric04] Brickley, Dan: RDF vocabulary description language 1.0: RDF schema. In: *http://www. w3. org/TR/rdf-schema/* (2004).

[Brow84] Browne, Jim; Dubois, Didier; Rathmill, Keith; Sethi, Suresh P.; Stecke, Kathryn E. [u. a.]: Classification of flexible manufacturing systems. In: *The FMS magazine* 2 (1984), Nr. 2, S. 114–117.

[Buss13] Bussmann, Stefan; Jennings, Nicolas R. ; Wooldridge, Michael: *Multiagent systems for manufacturing control: a design methodology.* Springer Science & Business Media, 2013.

[Camb12] Cambridge Semantics: *Examples of Semantic Web Applications.* August 2012. – [Online; Accesses 24-March-2019].

[Când10] Cândido, Gonçalo; Jammes, François; de Oliveira, José B. ; Colombo, Armando W.: SOA at device level in the industrial domain: Assessment of OPC UA and DPWS specifications. In: *Industrial Informatics (INDIN), 2010 8th IEEE International Conference on* IEEE, 2010, S. 598–603.

[Carv18] Carvalho, Núbia; Chaim, Omar; Cazarini, Edson ; Gerolamo, Mateus: Manufacturing in the fourth industrial revolution: A positive prospect in Sustainable Manufacturing. In: *Procedia Manufacturing* 21 (2018), S. 671–678.

[Chen17] Cheng, Haibo; Xue, Lingling; Wang, Peng; Zeng, Peng ; Yu, Haibin: Ontology-based web service integration for flexible manufacturing systems. In: *Industrial Informatics (INDIN), 2017 IEEE 15th International Conference on* IEEE, 2017, S. 351–356.

[Chir00] Chirn, Jin-Lung; McFarlane, Duncan C.: A holonic component-based approach to reconfigurable manufacturing control architecture. In: *Database and Expert Systems Applications, 2000. Proceedings. 11th International Workshop on* IEEE, 2000, S. 219–223.

[Chiu10] Chiu, Yuan-Shyi P.; Chen, Kuang-Ku; Cheng, Feng-Tsung ; Wu, Mei-Fang: Optimization of the finite production rate model with scrap, rework and stochastic machine breakdown. In: *Computers & mathematics with applications* 59 (2010), Nr. 2, S. 919–932.

[Chri01] Christensen, Erik; Curbera, Francisco; Meredith, Greg; Weerawarana, Sanjiva [u. a.]. *Web services description language (WSDL) 1.1*. 2001.

[Cisc15] Cisco: the Internet of Things: Extend the Cloud to Where the Things are. In: *Cisco White Paper* (2015).

[Clea96] Clearwater, Scott H.: *Market-based control: A paradigm for distributed resource allocation*. World Scientific, 1996.

[Colo12] Colombo, Armando W.; Mendes, J M.; Leitão, Paulo ; Karnouskos, Stamatis: Service-oriented SCADA and MES supporting petri nets based orchestrated automation systems. In: *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society* IEEE, 2012, S. 6144–6150.

[Colo14] Colombo, Armando W.; Bangemann, Thomas ; Karnouskos, Stamatis: IMC-AESOP outcomes: Paving the way to collaborative manufacturing systems. In: *Industrial Informatics (INDIN), 2014 12th IEEE International Conference on* IEEE, 2014, S. 255–260.

[Comm96] Committee, Technical; IMS (Hrsg.): *Intelligent Manufacturing Systems Research Program Archive*. June 1996. – URL: `http://ksi.cpsc.ucalgary.ca/IMS/IMS.html/` [Date Accessed: 2019-03-24].

[DANE99] DANEELS, A: WHAT IS SCADA? In: *International Conference on Accelerator and Large Experimental Physics Control Systems, 1999*, 1999.

[Dast16] Dastjerdi, Amir V.; Buyya, Rajkumar: Fog computing: Helping the Internet of Things realize its potential. In: *Computer* 49 (2016), Nr. 8, S. 112–116.

[De S08] De Souza, Luciana Moreira S.; Spiess, Patrik; Guinard, Dominique; Köhler, Moritz; Karnouskos, Stamatis ; Savio, Domnic: Socrades: A web

service based shop floor integration infrastructure. In: *The internet of things*. Springer, 2008, S. 50–67.

[Deug06] de Deugd, Scott; Carroll, Randy; Kelly, Kevin; Millett, Bill ; Ricker, Jeffrey: SODA: Service oriented device architecture. In: *IEEE Pervasive Computing* 5 (2006), Nr. 3, S. 94–96.

[Dilt91] Dilts, David M.; Boyd, Neil P. ; Whorms, HH: The evolution of control architectures for automated manufacturing systems. In: *Journal of manufacturing systems* 10 (1991), Nr. 1, S. 79–93.

[Econ17] for Economic Affairs, Federal M.; Paper, Energy (BMWi) W.: Application Scenario in Practice: Order-controlled Production of a Customised Bicycle Handlebar – Working Paper. In: *MKL Druck GmbH & Co. KG, Ostbevern*, 2017, S. 1–20.

[ElMa05] ElMaraghy, Hoda A.: Flexible and reconfigurable manufacturing systems paradigms. In: *International journal of flexible manufacturing systems* 17 (2005), Nr. 4, S. 261–276.

[ElMa09] ElMaraghy, Hoda A.: Changing and evolving products and systems–models and enablers. In: *Changeable and reconfigurable manufacturing systems*. Springer, 2009, S. 25–45.

[Endr04] Endrei, Mark; Ang, Jenny; Arsanjani, Ali; Chua, Sook; Comte, Philippe; Krogdahl, Pål; Luo, Min ; Newling, Tony: *Patterns: service-oriented architecture and web services*. IBM Corporation, International Technical Support Organization, 2004.

[Fens11a] Fensel, Dieter; Facca, Federico M.; Simperl, Elena ; Toma, Ioan: Lightweight semantic web service descriptions. In: *Semantic Web Services*. Springer, 2011, S. 279–295.

[Fens11b] Fensel, Dieter; Facca, Federico M.; Simperl, Elena ; Toma, Ioan: *Semantic web services*. Springer Science & Business Media, 2011.

[Fens18] Fensel, Anna: *Lecture notes in Semantic Web Services, Summer Semester, 2018*. June 2018. – URL: `https://www.sti-innsbruck.at/node/24522/` [Date Accessed: 2019-03-24].

[Fiel00] Fielding, Roy T.; Taylor, Richard N.: *Architectural styles and the design of network-based software architectures*. Bd. 7. University of California, Irvine Irvine, USA, 2000.

[Fort14] Fortineau, Virginie; Fiorentini, Xenia; Paviot, Thomas; Louis-Sidney, Ludovic ; Lamouri, Samir: Expressing formal rules within ontology-based

models using SWRL: an application to the nuclear industry. In: *International Journal of Product Lifecycle Management* 7 (2014), Nr. 1, S. 75–93.

[Fral03] Fraleigh, Chuck; Tobagi, Fouad ; Diot, Christophe: Provisioning IP backbone networks to support latency sensitive traffic. In: *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies* Bd. 1 IEEE, 2003, S. 375–385.

[Fray01] Frayret, Jean-Marc; D?Amours, Sophie; Montreuil, Benoit ; Cloutier, Louis: A network approach to operate agile manufacturing systems. In: *International Journal of Production Economics* 74 (2001), Nr. 1-3, S. 239–259.

[Frei04] Freitag, Michael; Herzog, Otthein ; Scholz-Reiter, Bernd: Selbststeuerung logistischer Prozesse–ein Paradigmenwechsel und seine Grenzen. In: *Industrie Management* 20 (2004), Nr. 1, S. 23–27.

[Früh17] Frühwirth, Thomas; Einfalt, Alfred; Diwold, Konrad ; Kastner, Wolfgang: A distributed multi-agent system for switching optimization in low-voltage power grids. In: *Emerging Technologies and Factory Automation (ETFA), 2017 22nd IEEE International Conference on* IEEE, 2017, S. 1–8.

[Gagn14] Gagnaire, Maurice; Diaz, Felipe; Coti, Camille; Cerin, Christophe; Shiozaki, Kazuhiko; Xu, Yingjie; Delort, Pierre; Smets, Jean-Paul; Le Lous, Jonathan; Lubiarz, Stephen [u. a.]: Downtime statistics of current cloud solutions. In: *International Working Group on Cloud Computing Resiliency, Tech. Rep* (2014).

[Gazi15] Gazis, Vangelis; Leonardi, Alessandro; Mathioudakis, Kostas; Sasloglou, Konstantinos; Kikiras, Panayotis ; Sudhaakar, Raghuram: Components of fog computing in an industrial internet of things context. In: *Sensing, Communication, and Networking-Workshops (SECON Workshops), 2015 12th Annual IEEE International Conference on* IEEE, 2015, S. 1–6.

[Gire04] Giret, Adriana; Botti, Vicente: Holons and agents. In: *Journal of Intelligent Manufacturing* 15 (2004), Oct, Nr. 5, S. 645–659. – ISSN 1572–8145.

[Give13] Givehchi, Omid; Trsek, Henning ; Jasperneite, Jürgen: Cloud computing for industrial automation systems?A comprehensive overview. In: *Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on* IEEE, 2013, S. 1–4.

[Gold15] Goldschmidt, Thomas; Murugaiah, Mahesh K.; Sonntag, Christian; Schlich, Bastian; Biallas, Sebastian ; Weber, Peter: Cloud-based control:

A multi-tenant, horizontally scalable soft-PLC. In: *2015 IEEE 8th International Conference on Cloud Computing (CLOUD)* IEEE, 2015, S. 909–916.

[Grea17] Grealou, Lionel; Technologies, TATA (Hrsg.): *PLM+ERP+MES+CRM: THE FOUR CORNERSTONES OF MANUFACTURING*. October 2017. – [Online; posted 31-October-2017].

[Haer83] Haerder, Theo; Reuter, Andreas: Principles of transaction-oriented database recovery. In: *ACM Computing Surveys (CSUR)* 15 (1983), Nr. 4, S. 287–317.

[Haik18] Haik, Yousef; Sivaloganathan, Sangarappillai ; Shahin, Tamer M.: *Engineering design process*. Nelson Education, 2018.

[Haki] Hakimpour, Farshad; Cong, Suo ; Damm, Daniela E.: A Practical Tutorial on Semantic Web Services. In: *hakimpour. com* .

[Haup13] Haupert, Jens: *DOMeMan: Repräsentation, Verwaltung und Nutzung von digitalen Objektgedächtnissen*. Saarland University, 2013.

[Hawk12] Hawke, Sandro; Herman, Ivan; Archer, Phil ; Prud, Eric; Herman, Ivan (Hrsg.): *W3C SEMANTIC WEB ACTIVITY*. June 2012. – URL: `https://www.w3.org/2001/sw/` [Date Accessed: 2019-03-24].

[Hedm14] Hedman, Jonas; Andersson, Bo: Selection method for COTS systems. In: *Procedia Technology* 16 (2014), S. 301–309.

[Hega15] Hegazy, Tamir; Hefeeda, Mohamed: Industrial automation as a cloud service. In: *IEEE Transactions on Parallel and Distributed Systems* 26 (2015), Nr. 10, S. 2750–2763.

[Helo14] Helo, Petri; Suorsa, Mikko; Hao, Yuqiuge ; Anussornnitisarn, Pornthep: Toward a cloud-based manufacturing execution system for distributed manufacturing. In: *Computers in Industry* 65 (2014), Nr. 4, S. 646–656.

[Hill] Hillier, Bob; Rule, Design (Hrsg.): *PLM vs ERP do you really need both?*. – [Online; Accesses 24-March-2019].

[Hitz10] Hitzler, Pascal; Krotzsch, Markus ; Rudolph, Sebastian: Foundations of Semantic Web Technologies. (2010).

[Holl99] Holland, Christoper P.; Light, Ben ; Gibson, Nicola: A critical success factors model for enterprise resource planning implementation. In: *Proceedings of the 7th European conference on information systems* Bd. 1, 1999, S. 273–287.

[Hong08] Hong, I-Hsuan; Ammons, Jane C. ; Realff, Matthew J.: Decentralized decision-making and protocol design for recycled material flows. In: *Interna-*

tional Journal of Production Economics 116 (2008), Nr. 2, S. 325–337.

[Hopp17] Hoppe, Stefan; Foundation, OPC (Hrsg.): *There Is No Industrie 4.0 without OPC UA*. June 2017. – URL: `https://opcconnect.opcfoundation.org/2017/06/there-is-no-industrie-4-0-without-opc-ua/` [Date Accessed: 2019-03-24].

[Horr04] Horrocks, Ian; Patel-Schneider, Peter F.; Boley, Harold; Tabet, Said; Grosof, Benjamin; Dean, Mike [u. a.]: SWRL: A semantic web rule language combining OWL and RuleML. In: *W3C Member submission* 21 (2004), S. 79.

[Horr05] Horrocks, Ian; Parsia, Bijan; Patel-Schneider, Peter ; Hendler, James: Semantic web architecture: Stack or two towers? In: *International Workshop on Principles and Practice of Semantic Web Reasoning* Springer, 2005, S. 37–41.

[Horr11] Horridge, Matthew; Bechhofer, Sean: The owl api: A java api for owl ontologies. In: *Semantic Web* 2 (2011), Nr. 1, S. 11–21.

[Huan02] Huang, Chin-Yin: Distributed manufacturing execution systems: A workflow perspective. In: *Journal of Intelligent manufacturing* 13 (2002), Nr. 6, S. 485–497.

[Hwan11] Hwang, Jing-Jang; Chuang, Hung-Kai; Hsu, Yi-Chang ; Wu, Chien-Hsing: A business model for cloud computing based on a separate encryption and decryption service. In: *Information Science and Applications (ICISA), 2011 International Conference on* IEEE, 2011, S. 1–7.

[IEC16] IEC, 62264; for Standardization, International O. (Hrsg.): *IEC 62264-3:2016, Enterprise-control system integration – Part 3: Activity models of manufacturing operations management*. December 2016. – [Online; Accesses 24-March-2019].

[Iiva90] Iivari, Juhani: Implementation of in-house developed vs application package based information systems. In: *ACM SIGMIS Database: the DATABASE for Advances in Information Systems* 21 (1990), Nr. 1, S. 1–10.

[Izag11] Izaguirre, M Jorge A G.; Lobov, Andrei ; Lastra, Jose L M.: OPC-UA and DPWS interoperability for factory floor monitoring using complex event processing. In: *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on* IEEE, 2011, S. 205–211.

[Jenn03] Jennings, Nicholas R.; Bussmann, Stefan: Agent-based control systems: Why are they suited to engineering complex systems? In: *IEEE control systems* 23 (2003), Nr. 3, S. 61–73.

[Jeon17] Jeon, Byeong W.; Um, Jumyung; Yoon, Soo C. ; Suk-Hwan, Suh: An

architecture design for smart manufacturing execution system. In: *Computer-Aided Design and Applications* 14 (2017), Nr. 4, S. 472–485.

[Jevt16] Jevtic, Jovan: *Evaluating benefits realisation management (BRM) methodology as a tool for implementing manufacturing execution system (MES)*, Diss., 2016.

[Ka98] Ka, B; Monostori, László; Szelke, E [u. a.]: An object-oriented framework for developing distributed manufacturing architectures. In: *Journal of Intelligent Manufacturing* 9 (1998), Nr. 2, S. 173–179.

[Kane16] Kaneko, Yu; Ito, Toshio: A Reliable Cloud-Based Feedback Control System. In: *Cloud Computing (CLOUD), 2016 IEEE 9th International Conference on* IEEE, 2016, S. 880–883.

[Kang12] Kang, Woochul; Kapitanova, Krasimira ; Son, Sang H.: RDDS: A real-time data distribution service for cyber-physical systems. In: *IEEE Transactions on Industrial Informatics* 8 (2012), Nr. 2, S. 393–405.

[Kara14] Karadgi, Sachin: *A Reference Architecture for Real-Time Performance Measurement*. Springer, 2014.

[Katt18a] Katti, Badarinath; Plociennik, Christiane; Ruskowski, Martin ; Schweitzer, Michael: SA-OPC-UA: Introducing Semantics to OPC-UA Application Methods. In: *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, 2018. – ISSN 2161–8089, S. 1189–1196.

[Katt18b] Katti, Badarinath; Plociennik, Christiane ; Schweitzer, Michael: GeSCo: Exploring the Edge Beneath the Cloud in Decentralized Manufacturing. In: *International Journal On Advances in Systems and Measurements* v11,1&2 (2018), S. 183–195.

[Katt18c] Katti, Badarinath; Plociennik, Christiane ; Schweitzer, Michael: A Jumpstart Framework for Semantically Enhanced OPC-UA. In: *KI - Künstliche Intelligenz* (2018), Dec.

[Katt18d] Katti, Badarinath; Plociennik, Christiane ; Schweitzer, Michael: SemOPC-UA: Introducing Semantics to OPC-UA Application Specific Methods. In: *IFAC-PapersOnLine* 51 (2018), Nr. 11, S. 1230–1236.

[Kend17] Kendrick, Blake A.; Dhokia, Vimal ; Newman, Stephen T.: Strategies to realize decentralized manufacture through hybrid manufacturing platforms. In: *Robotics and Computer-Integrated Manufacturing* 43 (2017), S. 68–78.

[Khan17] Khan, Waqas A.; Wisniewski, Lukasz; Lang, Dorota ; Jasperneite, Jürgen: Analysis of the requirements for offering industrie 4.0 applications as a cloud service. In: *Industrial Electronics (ISIE), 2017 IEEE 26th International Sym-*

posium on IEEE, 2017, S. 1181–1188.

[Khil11] Khilwani, Nitesh: *Role of Semantic web in the changing context of Enterprise Collaboration*, © Nitesh Khilwani, Diss., 2011.

[Klus] Kap. 3 In: Klusch, Matthias: *CASCOM - Intelligent Service Coordination in the Semantic Web.*

[Klus08] Klusch, Matthias: Semantic web service description. In: *CASCOM: intelligent service coordination in the semantic web.* Springer, 2008, S. 41–103.

[Koen10] Koenig, Sven; Keskinocak, Pinar ; Tovey, Craig A.: Progress on Agent Coordination with Cooperative Auctions. In: *AAAI* Bd. 10, 2010, S. 1713–1717.

[Koes67] Koestler, Arthur [u. a.]: The ghost in the machine. (1967).

[Kolb18] Kolberg, Dennis: *Entwicklung einer Referenzarchitektur zur Realisierung von Methoden der Lean Production mittels digitaler Technologien*, Technische Universität Kaiserslautern, doctoralthesis, 2018. – VIII, 189 S.

[Kope07] Kopeckỳ, Jacek; Vitvar, Tomas; Bournez, Carine ; Farrell, Joel: Sawsdl: Semantic annotations for wsdl and xml schema. In: *IEEE Internet Computing* (2007), Nr. 6, S. 60–67.

[Kore10] Koren, Yoram; Shpitalni, Moshe: Design of reconfigurable manufacturing systems. In: *Journal of manufacturing systems* 29 (2010), Nr. 4, S. 130–141.

[Kuma02] Kumara, SRT; Lee, Y-H ; Chatterjee, K: Distributed multiproject resource control: A market-based approach. In: *CIRP Annals-Manufacturing Technology* 51 (2002), Nr. 1, S. 367–370.

[Lang12] Langmann, Reinhard; Makarov, Oleg; Meyer, Laurid ; Nesterenko, Sergey: The woas project: Web-oriented automation system. In: *Remote Engineering and Virtual Instrumentation (REV), 2012 9th International Conference on* IEEE, 2012, S. 1–3.

[Lang14] Langmann, R; Meyer, L: Automation services from the cloud. In: *Remote Engineering and Virtual Instrumentation (REV), 2014 11th International Conference on* IEEE, 2014, S. 256–261.

[Lass99] Lassila, Ora; Swick, Ralph R.: Resource description framework (RDF) model and syntax specification. (1999).

[Lége05] Léger, Alain; Nixon, Lyndon J.; Shvaiko, Pavel ; Charlet, Jean: Semantic Web applications: Fields and Business cases. The Industry challenges the research. In: *IFIP Working Conference on Industrial Applications of Semantic Web* Springer, 2005, S. 27–46.

[Leit06a] Leitão, Paulo; Colombo, Armando W. ; Restivo, Francisco: A formal specification approach for holonic control systems: the ADACOR case. In: *International journal of manufacturing technology and management* 8 (2006), Nr. 1-3, S. 37–57.

[Leit06b] Leitão, Paulo; Restivo, Francisco: ADACOR: A holonic architecture for agile and adaptive manufacturing control. In: *Computers in industry* 57 (2006), Nr. 2, S. 121–130.

[Leit09] Leitão, Paulo: Agent-based distributed manufacturing control: A state-of-the-art survey. In: *Engineering Applications of Artificial Intelligence* 22 (2009), Nr. 7, S. 979–991.

[Leit13] Leitão, Paulo; Mařík, Vladimír ; Vrba, Pavel: Past, present, and future of industrial agent applications. In: *IEEE Transactions on Industrial Informatics* 9 (2013), Nr. 4, S. 2360–2372.

[Lena11] Lenart, Anna: ERP in the Cloud–Benefits and Challenges. In: *EuroSymposium on Systems Analysis and Design* Springer, 2011, S. 39–50.

[Lenn17] Lennvall, Tomas; Gidlund, Mikael ; Åkerberg, Johan: Challenges when bringing IoT into industrial automation. In: *AFRICON, 2017 IEEE* IEEE, 2017, S. 905–910.

[Li18] Li, Zhi; Zhou, Xiaowu; Wang, WM; Huang, George; Tian, Zonggui ; Huang, Shaowei: An ontology-based product design framework for manufacturability verification and knowledge reuse. In: *The International Journal of Advanced Manufacturing Technology* (2018), S. 1–15.

[Lobo09] Lobov, Andrei; Lopez, Fernando U.; Herrera, Vladimir V.; Puttonen, Juha ; Lastra, Jose L M.: Semantic Web Services framework for manufacturing industries. In: *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on* IEEE, 2009, S. 2104–2108.

[Losk11] Loskyll, Matthias; Schlick, Jochen; Hodek, Stefan; Ollinger, Lisa; Gerber, Tobias ; Pîrvu, Bogdan: Semantic service discovery and orchestration for manufacturing processes. In: *Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on* IEEE, 2011, S. 1–8.

[Losk13] Loskyll, Matthias: *Entwicklung einer Methodik zur dynamischen kontextbasierten Orchestrierung semantischer Feldgerätefunktionalitäten.* Technical University of Kaiserslautern, 2013.

[Ltd12] Ltd, Clydebuilt Business S.: Developing In-House Vs. Off the Shelf. In: *Clydebuilt Business Solutions Ltd White Paper* (2012).

[Maci12] Maciá Pérez, Francisco; Berna-Martinez, Jose V.; Marcos-Jorquera, Diego; Lorenzo Fonseca, Iren; Ferrándiz Colmeiro, Antonio [u. a.]: Cloud agile manufacturing. (2012).

[Mang11] Mangiuc, Dragos M.: ENTERPRISE 2.0-IS THE MARKET READY? In: *Accounting and Management Information Systems* 10 (2011), Nr. 4, S. 516.

[Mari05] Marik, Vladimir; McFarlane, Duncan: Industrial adoption of agent-based technologies. In: *IEEE Intelligent Systems* 20 (2005), Nr. 1, S. 27–35.

[Marq16] Marques, Maria; Agostinho, Carlos; Poler, Raul; Zacharewicz, Gregory ; Jardim-Gonçalves, Ricardo: An architecture to support responsive production in manufacturing companies. In: *Intelligent Systems (IS), 2016 IEEE 8th International Conference on* IEEE, 2016, S. 40–46.

[Marq17] Marques, Maria; Agostinho, Carlos; Zacharewicz, Gregory ; Jardim-Gonçalves, Ricardo: Decentralized decision support for intelligent manufacturing in Industry 4.0. In: *Journal of Ambient Intelligence and Smart Environments* 9 (2017), Nr. 3, S. 299–313.

[Mars11] Marston, Sean; Li, Zhi; Bandyopadhyay, Subhajyoti; Zhang, Juheng ; Ghalsasi, Anand: Cloud computing?The business perspective. In: *Decision support systems* 51 (2011), Nr. 1, S. 176–189.

[Mart04] Martin, David; Burstein, Mark; Hobbs, Jerry; Lassila, Ora; McDermott, Drew; McIlraith, Sheila; Narayanan, Srini; Paolucci, Massimo; Parsia, Bijan; Payne, Terry [u. a.]: OWL-S: Semantic markup for web services. In: *W3C member submission* 22 (2004), Nr. 4.

[MATS10] MATSOKIS, Aristeidis: *An Ontology-Based Approach for Closed-Loop Product Lifecycle Management*, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, Diss., 2010.

[Matu99] Maturana, Francisco; Shen, Weiming ; Norrie, Douglas H.: MetaMorph: an adaptive agent-based architecture for intelligent manufacturing. In: *International Journal of Production Research* 37 (1999), Nr. 10, S. 2159–2173.

[McIl01] McIlraith, Sheila A.; Son, Tran C. ; Zeng, Honglei: Semantic web services. In: *IEEE intelligent systems* 16 (2001), Nr. 2, S. 46–53.

[Mena00] Mena, Eduardo; Illarramendi, Arantza ; Goni, Alfredo: Automatic ontology construction for a multiagent-based software gathering service. In: *International Workshop on Cooperative Information Agents* Springer, 2000, S. 232–243.

[Merd09] Merdan, Munir: *Knowledge-based multi-agent architecture applied in the as-*

sembly domain. na, 2009.

[MESA97] MESA, International: Mes explained: A high level vision. In: *MESA International White Paper6* 1 (1997), September, S. 25.

[Modr09] Modrák, Vladimír; Mandulák, Ján: Mapping Development of MES Functionalities. In: *ICINCO-SPSMC*, 2009, S. 244–247.

[Mono06] Monostori, László; Váncza, József ; Kumara, Soundar R.: Agent-based systems for manufacturing. In: *CIRP Annals-Manufacturing Technology* 55 (2006), Nr. 2, S. 697–720.

[Mour13] Mourtzis, Dimitris; Doukas, Michalis: Decentralized manufacturing systems review: challenges and outlook. In: *Robust Manufacturing Control*. Springer, 2013, S. 355–369.

[Mube17] Mubeen, Saad; Nikolaidis, Pavlos; Didic, Alma; Pei-Breivold, Hongyu; Sandström, Kristian ; Behnam, Moris: Delay mitigation in offloaded cloud controllers in industrial iot. In: *IEEE Access* 5 (2017), S. 4418–4430.

[O'Co08] O'Connor, Martin J.; Shankar, Ravi D.; Musen, Mark A.; Das, Amar K. ; Nyulas, Csongor: The SWRLAPI: A Development Environment for Working with SWRL Rules. In: *OWLED*, 2008.

[OPC 02] OPC Foundation: *OPC Common Specifications*. December 2002. – [Online; Accesses 24-March-2019].

[OPC 17a] OPC Foundation: *OPC UA Specifications*. November 2017. – [Online; Accesses 24-March-2019].

[OPC 17b] OPC Foundation: OPC Unified Architecture Part 3 :Address Space Model. In: *OPC UA Part 3 - Address Space Model Release 1.04 Specification* 3 (2017), November.

[OPC 18a] OPC Foundation: *OPC UA Collaborations*. December 2018. – URL: `https://opcfoundation.org/Collaborations/` [Date Accessed: 2019-03-24].

[OPC 18b] OPC Foundation: OPC Unified Architecture Part 12 :Discovery and Global Services. In: *OPC UA Part 12 - Discovery and Global Services Release 1.04 Specification* 1.04 (2018), February.

[OPC 19] OPC Foundation: *OPC Unified Architecture*. Jan 2019. – URL: `https://opcfoundation.org/about/opc-technologies/opc-ua/` [Date Accessed: 2019-03-24].

[Orga09] Organization for the Advancement of Structured Information Standards (OASIS): *Devices Profile for Web Services*. July

2009. – URL: http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.html/ [Date Accessed: 2019-03-24].

[Park12] Park, Hong-Seok; Tran, Ngoc-Hien: An autonomous manufacturing system based on swarm of cognitive agents. In: *Journal of Manufacturing Systems* 31 (2012), Nr. 3, S. 337–348.

[Paru87] Parunak, H Van D.: Manufacturing experience with the contract net. In: *Distributed Artificial Intelligence, Volume I.* Elsevier, 1987, S. 285–310.

[Paru97a] Parunak, H Van D.; Baker, Albert D. ; Clark, Steven J.: The AARIA agent architecture: An example of requirements-driven agent-based system design. In: *Agents*, 1997, S. 482–483.

[Paru97b] Parunak, H Van D.; VanderBok, Raymond S. [u. a.]: Managing emergent behavior in distributed control systems. In: *Ann Arbor* 1001 (1997), S. 48106.

[Pauk16] Pauker, Florian; Frühwirth, Thomas; Kittl, Burkhard ; Kastner, Wolfgang: A systematic approach to OPC UA information model design. In: *Procedia CIRP* 57 (2016), S. 321–326.

[Paul93] Paulk, Mark C.; Curtis, Bill; Chrissis, Mary B. ; Weber, Charles V.: The capability maturity model for software. In: *Software engineering project management* 10 (1993), S. 1–26.

[Plat16] Platform, Industrie 4.: DIN SPEC 91345: Reference Architecture Model Industrie 4.0. In: *Reference Architecture Model Industrie* 4 (2016), April.

[Ploc11] Plociennik, Christiane: *Device Cooperation in Ad-hoc Multimedia Ensembles*, University of Rostock, Diss., 2011.

[Prab15] Prabhakar, Badrinath: *Advantages, Disadvantages between in house developed MES system and industry standard MES systems.* November 2015. – [Online; posted 11-November-2015].

[Pěch08] Pěchouček, Michal; Mařík, Vladimír: Industrial deployment of multi-agent technologies: review and selected case studies. In: *Autonomous agents and multi-agent systems* 17 (2008), Nr. 3, S. 397–431.

[Ren12] Ren, Lei; Zhang, Lin; Tao, Fei; Zhang, Xiaolong; Luo, Yongliang ; Zhang, Yabin: A methodology towards virtualisation-based high performance simulation platform supporting multidisciplinary design of complex products. In: *Enterprise IS* 6 (2012), S. 267–290.

[Rohj11] Rohjans, Sebastian; Fensel, Dieter ; Fensel, Anna: OPC UA goes semantics: Integrated communications in smart grids. In: *Emerging Technologies &*

Factory Automation (ETFA), 2011 IEEE 16th Conference on IEEE, 2011, S. 1–4.

[Roma05] Roman, Dumitru; Keller, Uwe; Lausen, Holger; De Bruijn, Jos; Lara, Rubén; Stollberg, Michael; Polleres, Axel; Feier, Cristina; Bussler, Cristoph ; Fensel, Dieter: Web service modeling ontology. In: *Applied ontology* 1 (2005), Nr. 1, S. 77–106.

[Rose15] Rosen, Roland; Von Wichert, Georg; Lo, George ; Bettenhausen, Kurt D.: About the importance of autonomy and digital twins for the future of manufacturing. In: *IFAC-PapersOnLine* 48 (2015), Nr. 3, S. 567–572.

[Russ16] Russell, Stuart J.; Norvig, Peter: *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[Saha06] Saharidis, Georgios K.; Dallery, Yves ; Karaesmen, Fikri: Centralized versus decentralized production planning. In: *RAIRO-Operations Research* 40 (2006), Nr. 2, S. 113–128.

[Sang17] Sang, Zhiqian; Xu, Xun: The framework of a cloud-based CNC system. In: *Procedia CIRP* 63 (2017), S. 82–88.

[Schl13] Schlick, Jochen; Stephan, Peter ; Greiner, Thomas: Kontext, Dienste und Cloud Computing. In: *atp edition* 55 (2013), Nr. 04, S. 32–41.

[Schn10] Schniederjans, Marc J.; Hamaker, Jamie L. ; Schniederjans, Ashlyn M.: *Information Technology Investment: Decision-Making Methodology Second Edition*. World Scientific Publishing Company, 2010.

[Scho07] Scholz-Reiter, B; Freitag, M: Autonomous processes in assembly systems. In: *CIRP annals* 56 (2007), Nr. 2, S. 712–729.

[Scho17] Schoitsch, Erwin; Skavhaug, Amund: 12th International ERCIM/EWICS/ARTEMIS Workshop on Dependable Smart Embedded Cyber-Physical Systems and Systems-of-Systems (DECSoS 2017). In: *Computer Safety, Reliability, and Security: SAFECOMP 2017 Workshops, ASSURE, DECSoS, SASSUR, TELERISE, and TIPS, Trento, Italy, September 12, 2017, Proceedings* Bd. 10489 Springer, 2017, S. 96.

[Seet15] Seethamraju, Ravi: Adoption of software as a service (SaaS) enterprise resource planning (ERP) systems in small and medium sized enterprises (SMEs). In: *Information systems frontiers* 17 (2015), Nr. 3, S. 475–492.

[Shaw96] Shaw, Mary; Garlan, David: *Software architecture*. Bd. 101. Prentice Hall Englewood Cliffs, 1996.

[Shen03] Shen, Weiming; Norrie, Douglas H. ; Barthès, Jean-Paul: *Multi-agent systems for concurrent intelligent design and manufacturing*. CRC press, 2003.

[Silv01] Silveira, Giovani D.; Borenstein, Denis ; Fogliatto, Flavio S.: Mass customization: Literature review and research directions. In: *International journal of production economics* 72 (2001), Nr. 1, S. 1–13.

[Smar18] SmartProduction, KUKA; Center, KUKA Smart P. (Hrsg.): *Matrix production: an example for Industrie 4.0*. February 2018. – [Online; posted 21-February-2018].

[Stev07] Steve Bratt: *Semantic web, and other technologies to watch*. August 2007. – [Online; Accesses 24-March-2019].

[Stop09] Stopper, Markus; Katalinic, Branko: Service-oriented architecture design aspects of OPC UA for industrial applications. In: *Proceedings of the International Multi-Conference of Engineers and Computer Scientists* Bd. 2 Citeseer, 2009.

[Tao11] Tao, Fei; Zhang, Lin; Venkatesh, VC; Luo, Y ; Cheng, Ying: Cloud manufacturing: a computing and service-oriented manufacturing model. In: *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 225 (2011), Nr. 10, S. 1969–1976.

[Tao14] Tao, Fei; Cheng, Ying; Da Xu, Li; Zhang, Lin ; Li, Bo H.: CCIoT-CMfg: cloud computing and internet of things-based cloud manufacturing service system. In: *IEEE Transactions on Industrial Informatics* 10 (2014), Nr. 2, S. 1435–1442.

[Tao15] Tao, Fei; Zhang, Lin; Liu, Yongkui; Cheng, Ying; Wang, Lihui ; Xu, Xun: Manufacturing service management in cloud manufacturing: overview and future research directions. In: *Journal of Manufacturing Science and Engineering* 137 (2015), Nr. 4, S. 040912.

[Tao18] Tao, Fei; Cheng, Jiangfeng; Qi, Qinglin; Zhang, Meng; Zhang, He ; Sui, Fangyuan: Digital twin-driven product design, manufacturing and service with big data. In: *The International Journal of Advanced Manufacturing Technology* 94 (2018), Nr. 9-12, S. 3563–3576.

[Thar96] Tharumarajah, A: Comparison of the bionic, fractal and holonic manufacturing system concepts. In: *International Journal of Computer Integrated Manufacturing* 9 (1996), Nr. 3, S. 217–226.

[Tren09] Trentesaux, Damien: Distributed control of production systems. In: *Engineering Applications of Artificial Intelligence* 22 (2009), Nr. 7, S. 971–978.

[Tsen97] Tseng, Mitchell M.; Lei, Ming; Su, Chuanjun ; Merchant, M E.: A collaborative control system for mass customization manufacturing. In: *Cirp Annals* 46 (1997), Nr. 1, S. 373–376.

[Tsit84] Tsitsiklis, John N.: Problems in decentralized decision making and computation. / MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR INFORMATION AND DECISION SYSTEMS. 1984. – Forschungsbericht.

[Ueda97] Ueda, Kanji; Vaario, Jari ; Ohkura, Kazuhiro: Modelling of biological manufacturing systems for dynamic reconfiguration. In: *CIRP Annals* 46 (1997), Nr. 1, S. 343–346.

[Ueda04] Ueda, Kanji; Lengyel, Attila ; Hatono, Itsuo: Emergent synthesis approaches to control and planning in make to order manufacturing environments. In: *CIRP Annals-Manufacturing Technology* 53 (2004), Nr. 1, S. 385–388.

[Umbl03] Umble, Elisabeth J.; Haft, Ronald R. ; Umble, M M.: Enterprise resource planning: Implementation procedures and critical success factors. In: *European journal of operational research* 146 (2003), Nr. 2, S. 241–257.

[Van 98] Van Brussel, Hendrik; Wyns, Jo; Valckenaers, Paul; Bongaerts, Luc ; Peeters, Patrick: Reference architecture for holonic manufacturing systems: PROSA. In: *Computers in industry* 37 (1998), Nr. 3, S. 255–274.

[Van 08] Van Deursen, Davy; Poppe, Chris; Martens, Gäetan; Mannens, Erik ; Van de Walle, Rik: XML to RDF conversion: a generic approach. In: *Automated solutions for Cross Media Content and Multi-channel Distribution, 2008. AXMEDIS'08. International Conference on* IEEE, 2008, S. 138–144.

[Varg05] Varga, LZ; Sztaki, ÁH: Semantic Web Services Description Based on Web Services Description. In: *W3C Workshop on Frameworks for Semantics in Web Services*, 2005.

[Vega97] Vega-Redondo, Fernando: The evolution of Walrasian behavior. In: *Econometrica: Journal of the Econometric Society* (1997), S. 375–384.

[Vere07] Verein Deutscher Ingenieure, Richtlinie: 5600, Manufacturing Execution Systems. In: *Verein Deutscher Ingenieure* (2007).

[Verl14] Verl, Alexander; Lechler, Armin: Steuerung aus der Cloud. In: *Industrie 4.0 in Produktion, Automatisierung und Logistik*. Springer, 2014, S. 235–247.

[Vers06] Verstraete, Paul; Saint Germain, Bart; Hadeli, Karuna; Valckenaers, Paul ; Van Brussel, Hendrik: On applying the PROSA reference architecture in multi-agent manufacturing control applications. In: *Proceedings of the Multi-agent Systems and Software Architecture Special Track at Net. ObjectDays*, 2006,

S. 31–47.

[Vija08] Vijayaraghavan, Athulan; Sobel, Will; Fox, Armando; Dornfeld, David ; Warndorf, Paul: Improving machine tool interoperability using standardized interface protocols: MT connect. (2008).

[Voge13] Vogel-Heuser, Birgit; Kegel, Gunther ; Wucherer, Klaus: Global information architecture for industrial automation. In: *atp edition* 51 (2013), Nr. 01-02, S. 108–115.

[Voor11] Voorsluys, William; Broberg, James ; Buyya, Rajkumar: Introduction to cloud computing. In: *Cloud computing: Principles and paradigms* (2011), S. 1–41.

[Vrba11] Vrba, Pavel; Radakovič, Miloslav; Obitko, Marek ; Mařík, Vladimír: Semantic technologies: latest advances in agent-based manufacturing control systems. In: *International Journal of Production Research* 49 (2011), Nr. 5, S. 1483–1496.

[Wagn10] Wagner, Thomas; Hausner, Carolin; Elger, Jurgen; Lowen, Ulrich ; Luder, Arndt: Engineering processes for decentralized factory automation systems. In: *Factory Automation*. InTech, 2010.

[Wahl12] Wahlster, Wolfgang: From industry 1.0 to industry 4.0: Towards the 4th industrial revolution. In: *Forum Business meets Research*, 2012.

[Wang00] Wangler, B; Paheerathan, SJ: Horizontal and vertical integration of organizational IT systems. In: *Information Systems Engineering* (2000).

[Wang10] Wang, Guohui; Ng, TS E.: The impact of virtualization on network performance of amazon ec2 data center. In: *Infocom, 2010 proceedings ieee* IEEE, 2010, S. 1–9.

[Web 16] Web Object Oriented Protocol for Software and Automation: *Woopsa Protocol Specifications*. August 2016. – [Online; Accesses 24-March-2019].

[Webe16] Weber, Werner; Hoess, Alfred; van Deventer, Jan; Oppenheimer, Frank; Ernst, Rolf; Kostrzewa, Adam; Doré, Philippe; Goubier, Thierry; Isakovic, Haris; Druml, Norbert [u. a.]: The EMC2 Project on Embedded Microcontrollers: Technical Progress after Two Years. In: *Digital System Design (DSD), 2016 Euromicro Conference on* IEEE, 2016, S. 524–531.

[Wei07] Wei, Chun-Chin; Liang, Gin-Shuh ; Wang, Mao-Jiun J.: A comprehensive supply chain management project selection framework under fuzzy environment. In: *International Journal of Project Management* 25 (2007), Nr. 6, S. 627–636.

[Wien07] Wiendahl, H-P; ElMaraghy, Hoda A.; Nyhuis, Peter; Zäh, Michael F.; Wiendahl, H-H; Duffie, Neil ; Brieke, Michael: Changeable manufacturing-classification, design and operation. In: *CIRP annals* 56 (2007), Nr. 2, S. 783–809.

[Wood09] Wood, Timothy; Shenoy, Prashant J.; Gerber, Alexandre; van der Merwe, Jacobus E. ; Ramakrishnan, Kadangode K.: The Case for Enterprise-Ready Virtual Private Clouds. In: *HotCloud*, 2009.

[Wu14] Wu, Wen-Hsiung; Fang, Lung-Ching; Wang, Wei-Yang; Yu, Min-Chun ; Kao, Hao-Yun: An advanced CMII-based engineering change management framework: the integration of PLM and ERP perspectives. In: *International Journal of Production Research* 52 (2014), Nr. 20, S. 6092–6109.

[Xu12] Xu, Xun: From cloud computing to cloud manufacturing. In: *Robotics and computer-integrated manufacturing* 28 (2012), Nr. 1, S. 75–86.

[Yang16] Yang, Zhixin; Zhang, Pengbo ; Chen, Lei: RFID-enabled indoor positioning method for a real-time manufacturing execution system using OS-ELM. In: *Neurocomputing* 174 (2016), S. 121–133.

[Zabo02] Zabojnik, Jan: Centralized and decentralized decision making in organizations. In: *Journal of Labor Economics* 20 (2002), Nr. 1, S. 1–22.

[Zeeb07] Zeeb, Elmar; Bobek, Andreas; Bohn, Hendrik ; Golatowski, Frank: Lessons learned from implementing the Devices Profile for Web Services. In: *Digital EcoSystems and Technologies Conference, 2007. DEST'07. Inaugural IEEE-IES* IEEE, 2007, S. 229–232.

[Zhan09] Zhang, Liang-Jie; Zhou, Qun: CCOA: Cloud computing open architecture. In: *Web Services, 2009. ICWS 2009. IEEE International Conference on* Ieee, 2009, S. 607–616.

[Zhan11] Zhang, Yingfeng; Huang, George Q.; Qu, Ting; Ho, Oscar ; Sun, Shudong: Agent-based smart objects management system for real-time ubiquitous manufacturing. In: *Robotics and Computer-Integrated Manufacturing* 27 (2011), Nr. 3, S. 538–549.

[Zhan17] Zhang, Yingfeng; Zhang, Geng; Liu, Yang ; Hu, Di: Research on services encapsulation and virtualization access model of machine for cloud manufacturing. In: *Journal of Intelligent Manufacturing* 28 (2017), Nr. 5, S. 1109–1123.

[Zhao17] Zhao, Yuanyuan; Liu, Quan; Xu, Wenjun; Xu, Xun W.; Yu, Shiqiang ; Zhou, Zude: An interoperable knowledge base for manufacturing resource and service capability. In: *International Journal of Manufacturing Research* 12 (2017), Nr.

1, S. 20–43.

[Zhon13] Zhong, Ray Y.; Dai, QY; Qu, T; Hu, GJ ; Huang, George Q.: RFID-enabled real-time manufacturing execution system for mass-customization production. In: *Robotics and Computer-Integrated Manufacturing* 29 (2013), Nr. 2, S. 283–292.

[Zimm80] Zimmermann, Hubert: OSI reference model–The ISO model of architecture for open systems interconnection. In: *IEEE Transactions on communications* 28 (1980), Nr. 4, S. 425–432.

[Zueh10] Zuehlke, Detlef: SmartFactory?Towards a factory-of-things. In: *Annual Reviews in Control* 34 (2010), Nr. 1, S. 129–138.

Curriculum vitae

Personal Details

Name:	Badarinath Katti
Nationality:	Indian
Residence:	Kaiserslautern, Germany

Education

09.2015-05.2020: Scientific Researcher / Doctoral Candidate
Lehrstuhl für Werkzeugmaschinen und Steuerungen (WSKL), Technische Universität Kaiserslautern.
SAP SE, Walldorf.
Thesis: Ontology-Based Approach to Decentralized Production Control in the Context of Cloud Manufacturing Execution Systems
Result: **Summa Cum Laude**

04.2013-06.2015: Masters in Computer Science
Specialization in Embedded Systems,
Technische Universität Kaiserslautern
Thesis: Conception and Experimental Evaluation of Event Driven Architecture in MES Context
Result: **Very Good**

08.2005-06.2009: Bachelors in Electronics and Communication Engineering, KLE Technological University, Hubballi, India
Thesis: Wireless Hotel Management using Microcontrollers
Result: **First class with Distinction**

Work Experience

10.2019-Present:	Developer at SAP, Walldorf
09.2015-05.2020:	PhD Student & Part time developer, SAP SE, Walldorf
12.2014-05.2015:	Masters Thesis Student, BASF, Ludwigshafen
06.2014-12.2014:	Research Assistant(HiWi), Fraunhofer IESE, Kaiserslautern
09.2013-03.2015:	Research Assistant (HiWi), TU Kaiserslautern
10.2009-02.2013:	Software Engineer, Infosys Limited, Bengaluru, India

Patent

Title:	Transfer of Production Control in Proximity to Production Site for enabling Decentralized Manufacturing
First Author:	Badarinath Katti
Second Author:	Dr. Michael Schweitzer