Czech Technical University in Prague
Faculty of Electrical Engineering

DOCTORAL THESIS

October 2017 **Ing. Václav Jirkovský**

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

SEMANTIC INTEGRATION IN THE CONTEXT OF CYBER-PHYSICAL SYSTEMS

Doctoral Thesis

Ing. Václav Jirkovský

Prague, October 2017

Ph.D. Programme: Electrical Engineering and Information Technology
Branch of Study: Artificial Intelligence and Biocybernetics

Supervisor: prof. Ing. Vladimír Mařík, DrSc., dr.h.c.

Acknowledgements

First and foremost, I would like to thank my wife Alena for her wonderful support. Further, I would like to thank my family for enabling and supporting my studies.

I would like to express my sincere gratitude to my supervisor Prof. Vladimír Mařík for the continuous support of my Ph.D. study and research, for his patience, motivation, enthusiasm, and immense knowledge. He provided me with the background theories and tools that I needed to choose the right direction and successfully complete my dissertation.

I want to thank Marek Obitko my colleague and friend for introducing me the world of the knowledge engineering.

I want to thank Petr Kadera and Petr Novák, my colleagues and friends, for their collaboration and support. You supported me greatly and were always willing to help me.

I would like to thank my colleagues from my internship at NII in Tokyo mainly my advisor Ryutaro Ichise for their wonderful collaboration.

The research done behind this thesis has been supported by Rockwell Automation Laboratory for Distributed Intelligent Control (RA-DIC), by HydroCon Company, and by the Grant Agency of the Czech Technical University in Prague, grant No. SGS12/188/OHK3/3T/13.

Václav Jirkovský

Czech Technical University in Prague
Prague, 2017

Abstract

Industrial systems have been developing into more and more complex systems during last decades. They have changed from centralized solutions to distributed, more robust, and more flexible eco-systems comprising a high number of embedded systems. In recent years, we are witnessing the research trend in the area of embedded systems which concerns the very close integration of physical and computing systems.

This dissertation thesis deals with the problem of the semantic integration of components (sensors and actuators) of cyber-physical systems within industrial automation domain and presents resulting benefits. Cyber-physical systems were created based on the aforementioned trend of the close integration of computing systems and physical systems. This tight integration involves infrastructures responsible for control, computation, communication, and sensing. These systems are composed of many subsystems produced by various manufacturers, and the subsystems produce an enormous volume of data. Furthermore, data generated from all of the system parts has different dimensions, sampling rates, levels of details, etc. Next, cyber-physical systems form systems which represent building blocks of the fourth industrial revolution (Industry 4.0) for example (Industrial) Internet of Things, Smart Cities, Smart Factories. Thus, the right understanding of data (data meanings, given context, subsystems purposes, and possible ways of subsystems integration) belong to essential requirements for enabling Industry 4.0 visions. In this thesis, the utilization of ontologies was proposed to deal with the semantic heterogeneity for enabling easier cyber-physical system components integration.

Moreover, the current widespread effort to create flexible highly customized manufacturing requires novel methods for data handling together with subsequent data utilization. Storing knowledge and data in an ontology offers a needed solution. For example, an ontology employment brings easy system data model management, increase an efficiency of cyber-physical system components interoperability, advanced data processing, reusability of sensors and actuators, and utilization of ontology matching methods for an integration of other data models. This work concerns the problem, how to describe cyber-physical system components using ontologies to enable effective integration. Next, the ontology matching system suitable for integration of heterogeneous data models in industrial automation domain is described. The proposed solution of the semantic interoperability is demonstrated on the Plug&Play cyber-physical system components.

On the other hand, storing data in an ontology and mainly processing of RDF statements brings one significant bottleneck — performance issue. Thus, Big Data technologies are employed for overcoming this issue together with a proposal of suitable storage data models. The overall approach is demonstrated on the proposed and developed prototype named Semantic Big Data Historian.

In particular, the main contributions of the dissertation thesis are as follows:

1. The proposal of the solution for CPS low-level semantic integration based on Semantic web Technologies together with a verification of a feasibility of proposed approach using Semantic Big Data Historian.

2. The overcoming performance issues of processing shop floor data represented as RDF-triples with the help of Big Data technologies and suitable storage data models — vertical partitioning and hybrid SBDH model.

3. The proposal and implementation of a suitable way how to integrate heterogeneous data models from industrial automation domain where the highest precision and recall are required. The approach is based on similarity measures aggregation using self-organizing maps and user involvement with the help of active learning and visualization of self-organizing map output layer.

4. Enabling reusability of cyber-physical system components together with effortless configuration based on utilization of Semantic Web technologies. This approach was named as Plug&Play cyber-physical system components.

Anotace

Během posledních desetiletí se průmyslové systémy vyvíjely ve více a více komplexní systémy. Změnily se z centralizovaných řešení na distribuované, více robustní a více flexibilní ekosystémy, které zahrnují velké množství vestavěných (embedded) systémů. Zmíněné změny navíc zahrnují výzkumný trend v oblasti vestavěných systémů, který se týká integrace fyzikálních a výpočetních systémů.

Tato disertační práce se zabývá problémem sémantické integrace komponent (senzorů a akčních členů) kyberneticko-fyzikálních systémů v oblasti průmyslové automatizace a z toho plynoucích výhod. Kyberneticko-fyzikální systémy byly vytvořeny na základě zmíněného výzkumného trendu integrace fyzikálních a výpočetních systémů. Tato integrace zahrnuje infrastruktury odpovědné za řízení, výpočty, komunikaci a snímání. Tyto systémy jsou složeny z mnoha podsystémů vyrobených různými výrobci a tyto podsystémy produkují enormní množství dat. Navíc generovaná data ze všech částí systému mají různé dimenze, vzorkovací frekvence, úrovně detailu, atd. Dále kyberneticko-fyzikální systémy formují systémy, které jsou stavebními bloky čtvrté průmyslové revoluce (Průmysl 4.0), například (průmyslový) internet věcí, chytrá města, chytré továrny. A tak správné porozumění datům (význam dat, daný kontext, účel podsystémů a jejich možné integrace) patří k základním požadavkům pro umožnění vizí Průmyslu 4.0. V této doktorské práci byl navržen způsob, jak se vypořádat se sémantickou heterogenitou pro usnadnění integrace komponent kyberneticko-fyzikálních systémů.

V současné době převládající snaha o vytvoření flexibilní a vysoce zakázkové výroby dále vyžaduje nové metody pro zpracování dat spolu s jejich následným využitím. Ukládání znalostí a dat v ontologii by mělo nabídnout požadované řešení. Použití ontologie přináší například snadnou správu datového modelu systému, zvyšuje efektivitu interoperability kyberneticko-fyzikálních komponent, pokročilé zpracování dat, opakované využití senzorů a akčních členů a použití metod ontologického mapování pro integraci dalších datových modelů. Tato práce se zabývá problémem, jak popsat komponenty kyberneticko-fyzikálních systémů v ontologii, aby mohly být efektivně integrovány. Dále je popsán systém pro mapování ontologií vhodný pro integraci heterogenních datových modelů v průmyslové automatizaci. Navržené řešení sémantické interoperability je demonstrováno na Plug&Play komponentách

kyberneticko-fyzikálních systémů.

Na druhé straně, ukládání dat v ontologii a především zpracování RDF záznamů přináší jednu podstatnou překážku — problém s výkonem. Proto v prezentovaném řešení je využito technologií velkých dat pro překonání tohoto problému spolu s navržením vhodného modelu pro ukládání dat. Celý přístup je demonstrován na navrženém a vyvinutém prototypu s názvem Semantic Big Data Historian.

Hlavní přínosy této doktorské práce jsou následující:

1. Navržení řešení pro nízkoúrovňovou sémantickou integraci kyberneticko-fyzikálního systému na základě technologií Sémantického Webu spolu s ověřením realizovatelnosti navrženého řešení za použití Semantic Big Data Historianu.

2. Překonáním výkonových problémů při zpracování výrobních dat popsaných jako RDF trojice s pomocí technologií velkých dat a vhodných datových modelů pro ukládání — vertikální rozdělování (vertical partitioning) a hybridní SBDH model.

3. Navržení a implementace vhodného způsobu, jak integrovat heterogenní datové modely v oblasti průmyslové automatizace, kde je vyžadovaná nejvyšší možná přesnost. Tento způsob je založen na agregaci měr podobnosti za použití Kohonenových map a zapojení uživatele pomocí aktivního učení a vizualizace výstupní vrstvy Kohonenovy mapy.

4. Umožnění znovupoužitelnosti komponent kyberneticko-fyzikálních systémů spolu se snadnou konfigurací založenou na využití technologií Sémantického Webu. Tento přístup byl pojmenován Plug&Play komponenty kyberneticko-fyzikálního systému.

Contents

I Introduction **1**

1 Introduction and Motivation **3**

1.1 Problem Statement 4

1.2 Goals of the Dissertation Thesis 5

1.3 Structure of the Dissertation Thesis 5

II State of the Art **9**

2 Cyber-Physical Systems **11**

2.1 Cyber-Physical Systems and Industry 4.0 13

2.2 Cyber-Physical Systems Applications 14

2.2.1 Cyber-Physical Systems Variants 16

2.3 Challenges of Cyber-Physical Systems 16

2.4 Summary 17

3 Ontologies **19**

3.1 Concepts 19

3.1.1 Knowledge Representation through Concepts 20

3.2 What Is Ontology? 21

3.2.1 Ontology Non-exact Definition 21

3.2.2 Ontology Definition 22

3.3 Formal Representation of Ontologies 27

3.3.1 Description Logic 27

3.4 Summary 29

4 Languages for Semantic Web **31**

4.1 Resource Description Framework 31

4.2 RDF Schema 33

4.3 Web Ontology Language 34

4.4 SPARQL 35

4.5 Summary 37

5 Information Integration Problem **39**

5.1 Data Heterogeneity 40

5.1.1 Heterogeneity Classification 41

5.1.2 What is Semantic Heterogeneity? 41

5.1.3 How to Find Relations Between Entities? 42

5.2 XML 44

5.3 Neutral Formats for Integration of Industrial Data 45

5.3.1 AutomationML 45

5.3.2 PLM XML 47

5.3.3 OPC Unified Architecture Information model 49

5.4 Ontologies for Integration 50

5.4.1 Integration based on Upper Ontologies 51

5.5 Semantic Sensor Network Ontology 53

5.6 Summary 55

6 Big Data **57**

6.1 Big Data Definition 58

6.2 Big Data Processing Frameworks 59

6.2.1 Batch Processing Frameworks 60

6.2.2 Stream Processing Frameworks 60

6.2.3 Hybrid Processing Frameworks 61

6.3 Summary 61

III Semantic Integration in the Context of Cyber-Physical Systems 65

7 Integration Challenge of CPSs 67

7.1 Low-level Integration 68

7.2 High-level Integration 71

7.3 Integration Process 72

7.4 Summary 73

8 Shared Ontology for Integration 75

8.1 Cyber-Physical System Ontology for Components Integration (COCI) 76

8.1.1 Required Concepts for Cyber-Physical System Components . 76

8.2 Design of Knowledge Base: Experimental System 82

8.3 Summary 85

9 Ontology Matching for CPS 87

9.1 Self-Organizing Map-Based Similarity Measures Aggregation 88

9.1.1 Visualization 89

9.1.2 Initial Classification 90

9.2 User Involvement in Ontology Matching 91

9.2.1 Active Learning 92

9.2.2 Candidate Selection 92

9.3 Experiments 94

9.4 Summary 96

10 Semantic Big Data Historian 97

10.1 Semantic Big Data Historian Architecture 98

10.1.1 Data Storage Layer 100

10.2 Validation Study 104

10.3 RDF Storage for Semantic Big Data Historian 107

10.3.1 Data Models for Triple Store 108

10.3.2 Comparison 110

10.4 Summary 111

11 Plug&Play Components of Cyber-Physical Systems 113
11.1 Data Model 115
11.2 Data Storage 115
11.3 Subsequent Data Processing 116
11.4 Work Flow 117
11.5 Validation Study 119
11.5.1 Semantic Extension of Sensor Data Model 119
11.5.2 Plug&Play CPS Component 121
11.5.3 Localization 122
11.6 Summary 123

IV Conclusions and Discussions 127

12 Conclusions 129
12.1 Summary and Discussion 129
12.1.1 Semantic Integration in the Context of Cyber-Physical Systems 130
12.2 Fulfillment of the Thesis Goals 132
12.3 Contributions of the Dissertation Thesis 133
12.4 Future Work 134

Bibliography 146

List of Author's Publications I

List of Figures

3.1 Example ontology containing concepts, individuals, and literals. . . . 23

4.1 RDF(S) graph. 32

5.1 Architecture of AutomationML, adopted from [1]. 46
5.2 Recommendation how to represent a) sensor b) actuator in AutomationML. 47
5.3 PLM XML information integration and sharing by means of PLM XML pipeline, adopted from [2]. 48
5.4 XSD schema for a sensor model within PLM XML format. 49
5.5 SSN key concepts and their relations, adopted from [3]. 55

6.1 Big Data characteristics — 3V(5V/7V) definition. 58

7.1 Cyber-physical system architecture. 68
7.2 Platform integration by means of corresponding adapters. 69
7.3 Syntactic heterogeneity within cyber-physical system. 70
7.4 High-level integration of cyber-physical systems. 72

8.1 Actuator concept detail in Protégé editor. 77
8.2 Actuating Capability concept detail in Protégé editor. 78
8.3 Action concept detail in Protégé editor. 79
8.4 Part of cyber-physical system ontology for components integration. . 81
8.5 Simplified COCI representation of the control gate actuator. 83
8.6 The first part of simplified COCI ontology. 84

8.7 The second part of simplified COCI ontology. 85

9.1 The SOM with rectangular topology. 88
9.2 MAPSOM — Visualization. 90
9.3 Candidate selection the smallest distance from the boundary. 93
9.4 Candidate selection the nearest samples from different classes. 93

10.1 Architecture overview of Semantic Big Data Historian. 99
10.2 Example of Data Storage Layer architecture combining Apache Spark and Apache Cassandra. 103
10.3 CPSs integration by means of COCI ontology. 106

11.1 Connection of known and unknown sensor. 118
11.2 RDF Representation in the string constant — COCIMetadata. . . . 119
11.3 Plug&Play sensors in Semantic Big Data Historian. 121

List of Tables

6.1 NIST – Big Data characteristics. 59

9.1 Dependencies between different SOM configuration and results (dataset No. 222). 95
9.2 MAPSOM and a real ontology (dataset No. 302). 96
9.3 MAPSOM and comparison with some of the best systems tested on dataset No. 302. 96

10.1 Comparison of different data models - writing data. 110
10.2 Comparison of different data models - reading data. 110
10.3 Vertical partitioning various number of sensors stored in one table — reading of 10,000 sensor samples 111

Part I

Introduction

Chapter 1

Introduction and Motivation

We are witnessing the research trend in the area of embedded systems which concerns the very close integration of physical and computing systems. This research trend constitutes the cornerstone of Cyber-Physical Systems (CPSs). CPSs represent integrated infrastructures responsible for control, computation, communication, and sensing. CPSs arrange a tight integration among controlled physical processes and controlling digital computing systems [4]. Application domain for these systems is quite extensive. For example, CPSs take part in a formation of systems which are building blocks of Industry 4.0 — Smart Buildings, Smart Cities, Smart Factories, Smart Homes [5], and Smart Grids [6].

Furthermore, new trends in customer demands force manufacturers to move from mass production to small batches of customized products. Products are manufactured in a wide variety and small volumes for each product type. Thus, production lines have to be frequently adjusted and more resistant to possible exceptions occurring during manufacturing execution. Flexible and reconfigurable manufacturing systems are essential requirements in a movement towards enabling the already mentioned Industry 4.0 paradigm.

The flexible and reconfigurable manufacturing requires perfect operation of all parts of a production line. This requirement also covers CPSs as well as all CPSs components. The faultless operation of CPSs requires their proper design and development. It cannot be achieved without the full understanding of all sub-systems, parts, and corresponding data formats and models. The main obstacle to achieving this goal is the semantic heterogeneity.

1.1 Problem Statement

Heterogeneity is a natural feature of all kinds of systems including sensors, actuators, and cyber-physical systems as well. It is caused due to the fact cyber-physical systems and their components are produced by various manufacturers with different data models, interfaces, and communication standards. In general, it is both a welcome and an unwelcome feature [7]. The heterogeneity is closely related to the system efficiency — more efficient system is more tailored to the problem. For example, if we design an ad-hoc smart sensor tailored to the given problem, then the sensor should be more efficient than a regular one but a reusability of the sensor for another application will be poor, and an integration of this sensor into a complex system will also be difficult. In other words, the heterogeneity causes significant obstacles for the systems interoperability.

This dissertation thesis deals with the problem how to integrate heterogeneous components of a cyber-physical system for enabling faultless operation of the system as well as how to integrate a new component into the system. A cyber-physical system consists of the two main parts — a physical part and a cyber part. The physical part involves the physical process and physical objects which provide a possibility for process control. The cyber part can be divided for clarity into two layers — the first layer (Platform Layer) represents a system integration of different devices from various manufacturers and the second layer (Computational Layer) represents the computational process which controls the physical process according to an implemented logic. Integral parts of CPSs are sensors and actuators which ensure a possibility to control a physical process. They are located on the boundary of the physical part and the platform layer. This dissertation thesis is focused on a semantic integration of cyber-physical systems components.

As already mentioned, a system designed exactly for a solution to a given problem is more efficient than solutions composed of reusable components. In many cases, a design of a system containing information representation, which is easily understandable, means storage of explicitly described data model's entities as well as their relations. For example, ontologies may be a suitable model for such a representation. Unfortunately, performance issues are still a problem. Thus, many approaches have not been widely accepted in real applications especially because of a poor efficiency concerning data handling and processing in comparison to proprietary solutions. In this dissertation thesis, the described problem is overcomed by a utilization of Big Data technologies.

1.2 Goals of the Dissertation Thesis

To summarize, the goals for this dissertation thesisare as follows:

1. Investigate and propose a possibility of increasing interoperability of heterogeneous cyber-physical system components — (smart) sensors and actuators.

2. Investigate and propose a possibility of increasing reusability of cyber-physical system components with the proposed method of data models representation.

3. Investigate, propose and evaluate a suitable way how to integrate heterogeneous data models in industrial automation domain.

4. Investigate, propose and evaluate methods how to efficiently store shop floor data in respective format.

5. Show and evaluate the feasibility of the integration of cyber-physical system components with the proposed approach.

1.3 Structure of the Dissertation Thesis

The thesis is organized as follows:

1. *Introduction*: Describes the motivation together with the goals of the thesis. There is also a list of contributions of this dissertation thesis.

2. *State-of-the-Art*: Introduces the reader to the necessary theoretical background and surveys the current state-of-the-art. First, the description of the

cyber-physical systems is provided. Next, the ontology definition and a description of Semantic Web is introduced together with formats for their representation. Then, the information integration problem is discussed with the focus on the semantic heterogeneity and ontology matching methods. It is followed by introducing possible utilization of neutral formats and ontologies for the integration. Finally, the Big Data paradigm and Big Data processing frameworks are presented.

3. *Overview of Proposed Approach and Main Results*: Semantic integration in the context of cyber-physical systems is introduced in this part. This part begins with the definition of the cyber-physical systems integration challenge. Next, the developed shared ontology for the integration of cyber-physical system components is introduced. Then, the proposed and developed system for an integration of data models is presented. The system is intended for semi-automatic ontology matching which is required for domains where the highest precision and recall of a matching is needed. The proposed approach is demonstrated with the help of proposed Semantic Big Data Historian. Finally, the concept of Plug&Play cyber-physical system components is introduced.

4. *Conclusions*: Summarizes the results of conducted research, suggests possible topics for further research, and concludes the thesis.

Part II

State of the Art

Chapter 2

Cyber-Physical Systems

Nowadays, we are witnessing the research trend in the area of embedded systems which concerns a very close integration of computing systems and physical systems, especially with the focus on a control. This trend results in a new category of devices named "Cyber-Physical Systems" (CPSs). CPSs may be described as integrated infrastructures involving communication, computation, control, and sensing. The aim of CPSs is a tight integration among controlled physical processes and controlling digital computing systems [8].

Around 2006, the term "Cyber-Physical System" was coined by Hellen Gill at the National Science Foundation (U.S.) to describe the integration of physical and computational processes [9]. An excellent introduction of CPS and CPS history may be found in [10]. Despite the fact the term CPS was coined in 2006, CPSs may be traced back to the 1st Industrial Revolution if we interpret CPSs as physical systems controlled or manipulated in a principled manner through engineering technologies [10].

There are several paths leading to CPSs [10]:

1. The first path to CPSs was started by the development of the first computer ENIAC (in 1946) [11]. Subsequently, computers began to be used to close control loops around physical systems. In the 1990s, greater interest in an integration of physical and computational processes arose with the architecture called "hybrid systems" — using differential equations for the description of

physical processes and discrete models of computations. The hybrid systems and control systems constituted CPSs.

2. The second path may be seen in the integration of communication and computation. Communication between computers started in 1969 when ARPANET [12] was introduced. The integration of communication and computation has been improving with increasing computational and communication capabilities — DARPA, the Internet, WIFI. Around 1998, Smart Dust project [13] introduced a new device (a mote) — a tiny device which is able to sense, communicate, and compute. Motes are connected with nodes which are interconnected by a network.

3. The next possible path is towards the first generation of control systems which provided analog control. These systems were based on the operational amplifier [14]. The system which enables utilization of the amplifier exploits frequency developed for example by Nyquist [15].

All of these paths have converged to the CPS concept. It includes the evolution of technology and several disciplines.

As mentioned, the architecture of CPS consists of three main parts — a cyber part, a physical part, and a network:

- The cyber part represents a computing core where physical system information is transformed into a model of a software system and corresponding rules establishing dependencies and relationships among software model entities together with control algorithms.

- The physical part represents a controlled object. This part involves physical processes and physical objects.

- The network represents a communication medium between a cyber and a physical part.

The reason for CPS development could be found not only in requirements for a decentralized computing system, which should ensure more efficient system operation but also in computing correctness of the system. If a developer designs a control system then he should take into his consideration a processing time of controlling tasks. If a task takes a long time for its processing, then it could be critical to correct functioning of the system. There may be many parallel processes running at once, and their dependencies demand the shortest processing time of computing tasks. Thus, a cyber part is moved from centralized form to a tightly coupled ecosystem together with a physical part.

CPSs could be utilized in various application domains which include vehicular systems and transportation [16], medical systems [17], smart homes and buildings [18], aerospace [19]. Applications will be discussed more detailed in section 2.2. CPSs have not only wide range of applications, but also a wide range of forms — CPSs may represent solutions from the smallest systems such as a pacemaker to complicated systems such as an aircraft safety system.

Another point of view is from the design perspective. A CPS may be described as a system which is responsible for providing following services — sensing service, computing service, actuating service.

2.1 Cyber-Physical Systems and Industry 4.0

The previous three industrial revolutions were started off by technical innovations. The first revolution was triggered by the invention of a steam engine. The second revolution was triggered by electric energy and manufacturing lines, and the third revolution was formed by the rise of electronics and information technology which enable industrial automation.

Industry 4.0 initiative was introduced at Hannover Fair with the presentation “Industry 4.0” [20]. The fourth revolution reflects customers requirements for cost-effective and highly customized production. Industry 4.0 exploits CPSs and Internet of Things and Services to fulfill mentioned requirements [21]. In other words, the new paradigm enables a realization of new business models, production concepts, smart controls, and individual user needs.

Present-day production systems require adaptations to comply with Industry 4.0 paradigm. Therefore, embedded systems (made of control units, sensors, and actuators) need an interface to the Internet, and it may be achieved by an extension of the embedded systems to a CPSs. There are various approaches to such an extension [21]:

- Direct system extension — this solution represents an extension where the embedded system is directly extended by a communication interface to access the Internet, and software is adapted to enable communication over a network.
- System expansion by a micro-controller board — the embedded system is connected to a micro-controller board with various communication interfaces such as CAN, Ethernet, WLAN.
- Extension by smart sensors and actuators — this extension of the embedded system is based on an architecture where communication to the Internet is managed by smart sensors and actuators.

CPSs are intended to be interconnected into more complex units to secure an advanced control of a production. The integration of CPSs means a significant obstacle for effective CPSs utilization within the era of Industry 4.0. Problems during the integration are caused mainly by semantic heterogeneity (more discussed in sec. 5.1.2).

2.2 Cyber-Physical Systems Applications

Cyber-physical systems have been used in many domains regardless it is explicitly named CPS or is not. Domains include automotive, aerospace, healthcare, industrial automation, etc. In following paragraphs, various CPS application domains [22] will be shortly discussed.

Vehicular systems become very complex and capable systems. Their systems provide functionality such as management of car behavior, entertainment, management of energy consumption, and enhanced displays. Here, CPSs may be involved in public transportation, electrical vehicle charging, or road monitoring.

For example, the important field is a fusion of human-centric data in vehicular CPSs. The method, which exploits a human factor for safety applications to assist human drivers, is presented in [23]. This approach helps in avoiding a negative impact of applications on a driver, for example, information overload or distraction.

Medical and healthcare systems — CPSs are crucial in the modern medical systems. The modern medicine has to be based on advanced technologies to be as effective as possible and minimizing the side effects at the same time. CPSs applications cover robot-assisted operation, life-support medical devices, or development of medical applications.

The interesting application is for example discussed in [24]. It describes a design of a CPS for neurally controlled artificial legs.

Smart Buildings — areas for a utilization of CPSs in smart buildings include HVAC control and energy/power management in Smart Buildings.

Furthermore, an approach described in [25] introduces a solution where a CPS is formed from a smart community — networked homes. Smart homes are modeled as multifunctional sensors, and feedback to physical processes is used for improving home security or health care quality.

Manufacturing applications of CPSs are specific because of the domain nature. The industrial domain has several barriers to the full exploitation of the CPS concept because the industry is a rather conservative domain. All of the changes have to be in strategical road maps of manufacturers, and changes in manufacturing have to be addressed as a global issue instead of a local one [26]. Most of us may imagine a utilization of a CPS in a production. A CPS collects information from a production line (from a particular production process). Next, a CPS evaluates information about the process (together with other integrated systems). Finally, a CPS perform a feedback to a physical production process. In flexible manufacturing, it may be represented by a workstation which is able to adjust its operations according to an incoming type of product.

Moreover, there are other more complex applications of CPSs in manufacturing. The most interesting ones are as follows:

- *Adaptive manufacturing* system was introduced for example by FESTO pre-industrial system — MiniProd [27]. The system is based on distributed control with the help of agent-oriented architecture. MiniProd application is able to be reconfigured on-the-fly, i.e., distributed modules may be self-configured thanks to their embedded controllers and the autonomous nature.

- *Model-driven manufacturing* may be represented by a 3D model-driven robot-in-the-loop approach [28]. The approach offers a possibility for remote assembly with the help of a cloud environment — off-site operator is able to manipulate a physical robot instantly via virtual robot control in cyber-space.

2.2.1 Cyber-Physical Systems Variants

Several CPS variants have emerged in the course of time. Mostly, they are distinguished by comprising a new component into the original CPS concept. In the following paragraphs, the most specific variants are introduced — Cyber-Physical-Social Systems and Cloud-based Cyber-Physical Systems.

Very specific type of CPSs is **Cyber-Physical-Social Systems** (CPSS) [29]. These systems consider a human as a part of CPS instead of keeping them separate. It means the common parts of CPS (physical, cyber, and communication part) were supplemented by human knowledge, mental capabilities, and even socio-cultural elements. A CPSS has its place in applications where a human factor is indispensable. Despite the advances of cybernetic technologies, there are still many operations which are unfeasible by automation or automation of an operation may be very expensive. Another requirement for CPSS may be social aspects. Because of various reasons, it may be demanded to preserve or create a new job. Assisting CPSs seem to be a promising way for mentioned problems. From another point of view, a human may serve for sensing, actuating, or computational resource.

Another variant is **Cloud-based Cyber-Physical-System** [30]. These systems provide a possibility to conduct a computation in a cloud. It may be worth when the powerful computational hardware is needed. Furthermore, Cloud-based Cyber-Physical-System may be important for integration of various systems, for example when several CPS share a cloud computational resource.

2.3 Challenges of Cyber-Physical Systems

Expectations from CPS are enormous [31]. It covers for example robustness, self-organization, efficiency, and interoperability.

There are many challenges slightly different in various domains mostly motivated by expectations and requirements. A subset of all challenges is presented in the following listing:

- *Context-aware systems* — this challenge reflects requirements for comprehensive context-aware systems. The context-aware feature is important for recognition, analysis, and interpretation of CPS's intentions and its parts. In other words, a CPS may be purposefully utilized only with a proper understanding of a given context.
- *Cooperative (production) systems* — CPSs may form a complex interconnected unit. The challenge is not in a way, how to connect CPSs but how to make them working cooperatively. As you can see, this challenge is tightly connected to context-aware systems.
- *Prediction of dynamical systems* — An extension of the available prediction methods or a development of new ones are to be provided (especially in the case of interconnected CPSs).
- *Effective and well-operating systems* — CPSs may be assembled from many components (sensors, actuators, control units) which have been produced by many manufacturers. A proper assembly of these components, as well as proper understanding and exploitation of corresponding data models, is one of the most important challenges which is necessary for faultless and effective CPSs operation.

2.4 Summary

The intention of this work is to face the problem which was outlined in previous paragraphs — a semantic integration. The semantic integration is essential mainly for enabling context-aware systems, cooperative systems, and effective and well-operating systems. The semantic integration in the context of CPSs is discussed in section 7 in detail.

Chapter 3

Ontologies

The vast amount of information that surrounds us is stored variously in books, audio and video records, informational systems, various web pages and other media. Furthermore, there is increasing trend of data production caused by a utilization of more capable and automated devices as well as increasing user demands for advanced analytics. Without advanced knowledge representation, a huge amount of data from various sources are inapplicable. Knowledge management is a significant feature for CPS as well. Thus, one of the widely accepted [32] way for knowledge representation and utilization will be introduced in this chapter.

3.1 Concepts

The term *concept* is derived from Latin word *conceptum.* The origin of the classical theory of concepts is in philosophy. The Greek philosopher Aristotle defined a concept within the "theory of definition" by using two characteristics — *genus* (a kind or a family) and *differentiae* (a distinguishing characteristic) [33]. According to this theory, the concept "notebook" could be defined as a "portable computer". The "computer" corresponds to *genus* and "portable" corresponds to *differentiae* as it distinguishes notebooks other computers. Next, many philosophers also tried to define the term concept in different ways, for example, Schopenhauer [34] and Kant [35]. In other words, the meaning of the term concept can be expressed as the very essence of human existence and perception of the world.

A perception of the term concept in other scientific fields is often more pragmatic than in the philosophy domain. Thus, the definitions correspond to the way concepts are used in practice. For example, the term concept is understood (in linguistics) as meaning that is shared in common by the relevant terms in the minds of those who use these terms [36]. An illustrating example is WordNet database[1]. This lexical database contains interconnected English nouns, verbs, and adverbs. Concepts are defined through synsets — sets of synonym words. For example, a synset, which is defined by the following set of synonyms animate being, beast, brute, creature, fauna (a living organism characterized by voluntary movement) and animal, represents a concept that can be lexically expressed by any of the words in the synset.

In contrary to linguistics, the mathematician Carl Boyer expressed mathematical concept as a well-defined abstract mental construct which is beyond the world of sensory experience [37]. Here, concepts overstep the reality that is identified by our intuition.

3.1.1 Knowledge Representation through Concepts

In the knowledge representation, concepts are considered as atomic elements. There are several formalisms for organized knowledge representation, for example, concept maps where concepts are perceived as regularity in events or object, designated by a label (a word or a symbol) [38]. Another formalism for representing logic-based knowledge with the help of concepts, roles and individuals is description logic. Here, concepts represent sets of individual objects [39].

Another technique inspired by the organization of human memory are Frames. They are recursive attribute-value structures used as a general format in accounting for the content of mental concepts. Instead of being taken as atomic units (in contrast to many other formalisms for representing knowledge), concepts came to be understood as classes of highly structured entities describable regarding recursive attribute-value structures [40].

Concepts are basic building blocks of ontologies which are introduced in the following sections.

[1]https://wordnet.princeton.edu

3.2 What Is Ontology?

The word “ontology” has various meanings in different domains. The “Ontology” with the initial upper-case letter usually refers to philosophy. Aristotle defined Ontology as the science of “being qua being” i.e., the study of attributes that belong to things because of their very nature [41]. Oppositely, the term “ontology” (with the initial lower-case letter) as a model of a system in a computational sense is introduced in the following paragraphs and is used in the remaining text of this thesis.

3.2.1 Ontology Non-exact Definition

The term ontology was adopted in the computational sense through the artificial intelligence where it refers to a representation of the real world of computational systems.

The well-known definition of the term ontology is derived from two definitions introduced by Gruber [42] in 1993 and by Borst [43] in 1997. The definition was introduced by Studer [44] in 1998 and is expressed as: “*An ontology is a formal, explicit specification of a shared conceptualization.*” The detailed meaning of this definition is discussed in the following paragraph.

What is a *conceptualization*? A widespread and known notion of conceptualization refers to Genesereth and Nilsson [45]: “A body of formally represented knowledge is based on a conceptualization: the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them. A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose.”

What is a *formal* and *explicit specification*? A conceptualization used during human communication is typically implicit, i.e., in the mind of people. On the contrary, an ontology (concepts, relations, etc.) has to be defined explicitly. Next, explicit specifications of the conceptualization can be done *extensionally* or *intensionally*. However, an extensional definition is impossible in many cases or unsuitable because it would require listing the extensions of every relation for all possible element. On the other hand, the more suitable way is to specify conceptualization intensionally using appropriate axioms (meaning postulates [46]). In other words, an ontology is some set of necessary axioms. Furthermore, the specification has to be formal (i.e., machine-readable) in comparison with natural language which is informal.

Shared conceptualization is necessary because ontologies capture consensual knowledge established by a community of users. In short, an ontology may become useless if it is used incompatible with ontological commitments (established by a given group of interested users).

3.2.2 Ontology Definition

The previously introduced ontology definition is sufficient for many applications. However, the proper definition of the terms such as ontology, instance, knowledge base, lexicon, etc., is needed for avoiding subsequent confusions in terms of misinterpretation. The overview of the main definitions is provided in following paragraphs. All following definitions are discussed in detail in [47, 48].

Definition 3.2.1. (Ontology) An ontology $\mathcal{O}$ is a structure

$$\mathcal{O} := (\mathcal{C}, \leq_{\mathcal{C}}, \mathcal{R}, \sigma_{\mathcal{R}}, \leq_{\mathcal{R}}, \mathcal{A}, \sigma_{\mathcal{A}}, \mathcal{T}) \tag{3.1}$$

where

- disjoint sets $\mathcal{C}, \mathcal{R}, \mathcal{A}$ and $\mathcal{T}$ represent concepts, relations, attributes and data types, respectively,
- a upper semi-lattice $\leq_{\mathcal{C}}$ on $\mathcal{C}$ with top element $root_{\mathcal{C}}$, called concept hierarchy,
- a function $\sigma_{\mathcal{R}} : \mathcal{R} \rightarrow \mathcal{C}^+$ called relation signature,
- a upper semi-lattice $\leq_{\mathcal{R}}$ on $\mathcal{R}$ called relation hierarchy,
- a function $\sigma_{\mathcal{A}} : \mathcal{A} \rightarrow \mathcal{C} \times \mathcal{T}$, called attribute signature, and
- a set $\mathcal{T}$ of datatypes.

For the sake of simplicity, the σ will be used without an index, when it is apparent from the give context whether it is related to a relation or an attribute. Correct applications of binary relations may be maintained with the help of their *domain* and their *range*:

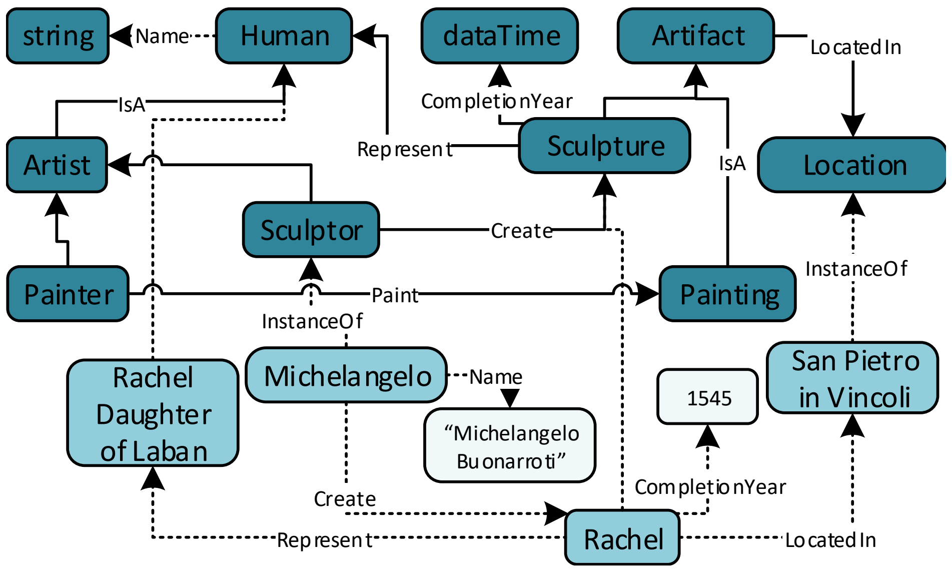


Figure 3.1: Example ontology containing concepts, individuals, and literals.

Definition 3.2.2. (Domain and Range) For a relation $r \in \mathcal{R}$ with $|\sigma(r)| = 2$, domain and range are define by $dom(r) := \pi_1(\sigma(r))$ and $range(r) := \pi_2(\sigma(r))$.

Where $\pi_i(t)$ is the i-th component of a tuple t.

The above definitions are demonstrated on the basis of an example ontology which is illustrated on Fig. 3.1. The ontology contains the set $\mathcal{C}$ of concepts: $\mathcal{C} := \{Human, Artist, Painter, Sculptor, Artifact, Painting, Sculpture, Location\}$, the set $\mathcal{R}$ of relations:
$\mathcal{R} := \{Represent, LocatedIn, Create, Paint\}$, and the set $\mathcal{A}$ of attributes: $\mathcal{A} := \{Name, CompletionYear\}$. Next, the partial order $\leq_{\mathcal{C}}$ is then $\leq_{\mathcal{C}} := \{(Sculptor, Artist), (Painter, Artist), (Artist, Human), (Sculpture, Artifact), etc.\}$.
Furthermore, the example ontology has the following signitures:

- $\sigma_{\mathcal{R}}(Represent) = (Sculpture, Human)$
- $\sigma_{\mathcal{R}}(LocatedIn) = (Artifact, Location)$
- $\sigma_{\mathcal{R}}(Create) = (Sculptor, Sculpture)$
- $\sigma_{\mathcal{R}}(Paint) = (Painter, Painting)$

- $\sigma_{\mathcal{A}}(Name) = (Human, String)$
- $\sigma_{\mathcal{A}}(CompletationYear) = (Sculpture, dateTime)$

For example, the domain and the range of the relation $Paint$ are: $dom(Paint) := \pi_1(\sigma(Paint)) = Painter$, $range(Paint) := \pi_1(\sigma(Paint)) = Painting$.

Now, the core definition of the ontology concept have been introduced. Next, an important axiom system is presented in following definition.

Definition 3.2.3. ($\mathcal{L}$-Axiom System) Let $\mathcal{L}$ be a logical language. A $\mathcal{L}$-axiom system for an ontology $\mathcal{O}$ is a triple

$$S := (AS, \alpha, \mathcal{L}) \tag{3.2}$$

where

- a set AS represents axiom schemata and
- a function $\alpha : AS \to AS_{\mathcal{L}}$ is a mapping from AS to axiom schemata defined over $\mathcal{L}$.

An example of a mapping one axiom schema $\lambda P, Q.disjoint(P, Q)$ to a first-order logic schema is as follows

$$\lambda P, Q.\forall x \left(P\left(x\right) \to \neg Q\left(x\right)\right). \tag{3.3}$$

This axiom linked to the example ontology has following form:

$$\alpha(disjoint)(Painting)(Sculpture), \tag{3.4}$$

with the mapping to a first-order logic schema:

$$\forall x(Painting(x) \to \neg Sculpture(x)). \tag{3.5}$$

The main advantage of the introduction of such an (L)-axiom system to the ontology concept is providing an independence on some concrete knowledge representation. These axioms may be transformed into various languages. Generally, statements representing real-world facts may be in fact intuitively interpreted independently of any certain knowledge representation. A specific interpretation may be then assigned via the α mapping. The given part of the world, which is going to be described in an ontology, contains many axioms. However, the status of an axiom schema should belong to only frequently occurring axioms.

The following definition introduces a Lexicon:

Definition 3.2.4. (Lexicon) A lexicon for an ontology $\mathcal{O}$ is a structure

$$Lex := (S_{\mathcal{C}}, S_{\mathcal{R}}, S_{\mathcal{A}}, Ref_{\mathcal{C}}, Ref_{\mathcal{R}}, Ref_{\mathcal{A}}) \tag{3.6}$$

where

- ○ sets $S_{\mathcal{C}}$, $S_{\mathcal{R}}$ and $S_{\mathcal{A}}$ represent signs for concepts, relations and attributes, respectively,
- ○ a relation $Ref_{\mathcal{C}} \subseteq S_{\mathcal{C}}$ is lexical reference for concepts,
- ○ a relation $Ref_{\mathcal{R}} \subseteq S_{\mathcal{R}}$ is lexical reference for relations, and
- ○ a relation $Ref_{\mathcal{A}} \subseteq S_{\mathcal{A}}$ is lexical reference for attributes.

For $s \in S_{\mathcal{C}} Ref_{\mathcal{C}}$ is represented by

$$Ref_{\mathcal{C}}(s) := \{c \in \mathcal{C} | (s, c) \in Ref_{\mathcal{C}}\} \tag{3.7}$$

and, $Ref_{\mathcal{C}}^{-1}$ is defined for $c \in (C)$

$$Ref_{\mathcal{C}}^{-1}(c) := \{s \in S_{\mathcal{C}} | (s, c) \in Ref_{\mathcal{C}}\}. \tag{3.8}$$

Lexical and inverse lexical references for relations and attributes are defined analogously.

This definition may be utilized to specify that both *Location* and *Place* refer to the same concept Location: $Ref_{\mathcal{C}}^{-1}(Location) = Location, Place$ or that *Represent* and *Symbolize* refer to the same relation Represent: $Ref_{\mathcal{R}}^{-1}(Represent) = Symbolize, Represent$

Ontologies define a conceptualization of a given domain. Moreover, a complementary part of ontologies is represented by a knowledge base which is built on assertions about instances of concepts and relations.

Definition 3.2.5. (Knowledge Base (KB)) A knowledge base for an ontology $\mathcal{O}$ is a structure

$$KB := (\mathcal{I}, \iota_{\mathcal{C}}, \iota_{\mathcal{R}}, \iota_{\mathcal{A}}) \tag{3.9}$$

where

- $\mathcal{I}$ is a set of instance identifiers (or instances or individuals)
- a function $\iota_{\mathcal{C}} : \mathcal{C} \to 2^{\mathcal{I}}$ is concept instantiation,
- a function $\iota_{\mathcal{R}} : \mathcal{R} \to 2^{\mathcal{I}^+}$ is relation instantiation, and
- a function $\iota_{\mathcal{A}} : \mathcal{A} \to \mathcal{I} \times \bigcup_{t \in \mathcal{T}} [[t]]$, where $[[t]]$ are the values of datatype $t \in \mathcal{T}$, is attribute instantiation.

The example ontology contains the set of instances $\mathcal{I} := \{Michelangelo, Rachel, RachelDaughterofLaban, SanPietroinVincoli\}$. Next, the ontology has following instantiation relations:

- $\iota_{\mathcal{C}}(Human) := \{RachelDaughterofLaban\}$
- $\iota_{\mathcal{C}}(Sculpture) := \{Rachel\}$
- $\iota_{\mathcal{C}}(Sculptor) := \{Michelangelo\}$
- $\iota_{\mathcal{C}}(Location) := \{SanPietroinVincoli\}$
- $\iota_{\mathcal{R}}(Create) := \{Michelangelo, Rachel\}$
- $\iota_{\mathcal{R}}(Represent) := \{Rachel, RachelDaughterofLaban\}$
- $\iota_{\mathcal{R}}(LocatedIn) := \{Rachel, SanPietroinVincoli\}$
- $\iota_{\mathcal{A}}(Name) := \{Michelangelo, ``MichelangeloBuonarroti''\}$
- $\iota_{\mathcal{A}}(CompletionYear) := \{Rachel, 1545\}$

Occasionally, it is needed to assign names also to instances:

Definition 3.2.6. (Instance Lexicon) An instance lexicon for a knowledge base KB is a pair

$$IL := (S_{\mathcal{I}}, R_{\mathcal{I}}) \tag{3.10}$$

where

- ∘ $S_{\mathcal{I}}$ is set of signs for instances,
- ∘ $R_{\mathcal{I}}$ is a relation $R_{\mathcal{I}} \subseteq S_{\mathcal{I}} \times \mathcal{I}$ called lexical reference for instances.

3.3 Formal Representation of Ontologies

We know what the term ontology means. During an ontology design, it is needed to choose an appropriate ontology language. There are many various ontology languages. They differ from each other in various expressivity and complexity of their inference capability. The following categorization from [49] is adopted. The first category represents traditional ontology languages and may be divided into several groups:

- ∘ **Based on first-order predicate logic.** Representative languages of this category are KIF [50] and CycL [51]. The cornerstone (the central modeling primitives) of these languages are predicates.
- ∘ **Frame-based languages.** Representative languages of this category are Ontolingua [52] and F-logic [53].
- ∘ **Description logic based languages.** In this category, concepts and their properties are described with the help of description logics.

The second category represents Web standards which were primarily intended for facilitating knowledge interchange on the Internet. In this work, the OWL is used for ontology modeling, and thus this category is described in detail in chapter 4.

3.3.1 Description Logic

In the following paragraphs, description logic will be described because it is an important part of OWL language (i.e., OWL DL or OWL Full).

Description Logics (DLs) [54] is a family of knowledge representation languages that may be used for a representation of knowledge of any application domain. This representation is in a structured and formally understandable way. The name Description Logics is derived from two features — the first feature is the capability to describe a given domain with the help of concept descriptions; the second feature is to provide logic-based semantics in contrast, for example, semantic networks or frames.

Commonly, DLs include a terminological and an assertional formalism. A set of terminological axioms (TBox) is used to describe names (or labels) for complex descriptions. For example, TBox may contain a description of a concept Mother:

$$Human \sqcap Parent \sqcap Woman.$$

On the other hand, a set of assertional axioms (ABox) is used for description of properties of individuals. For example, we can express that individual named Karel is son of individual Pavel:

$$hasChild(Pavel, Karel).$$

DLs offer capabilities to deduce implicit knowledge from the explicitly defined knowledge with the help of TBox and ABox.

The DLs provide a well-defined semantics and powerful reasoning tools. For many years, there was a mismatch between the expressivity of DLs and the efficiency of reasoning. In other words, if a user wants to use a DLs, then he needs to establish a trade-off between the expressivity of DLs and the complexity of their inference capability. It means it is needed to restrict DL appropriately.

The suitability of DLs for ontology definitions has been introduced by web ontology languages, (e.g., OWL which is described in detail in section 4). The cornerstone for OWL design was the expressive DL $\mathcal{SHIQ}$ [55]. In OWL language, the developers tried to find a balance between expressiveness and the complexity of reasoning. Furthermore, more detailed description of DLs may be find in [54].

3.4 Summary

Ontologies represent a promising way for knowledge representation. They have pervaded various domains and we can expect a widening of their utilization in an industrial domain, in forthcoming years as well. In this work, they are considered as suitable for improving interoperability of CPS(s), and we will try to prove this assumption.

Chapter 4

Languages for Semantic Web

The purpose of ontologies was to facilitate a way how shared knowledge may be interchanged among people as well as various systems. Internet technologies are suitable to be exploited for knowledge sharing. From the other point of view, the Internet should be extended to have a structure which allows expressing the meaningful content of web pages. It should be utilized for creating an ecosystem where software agents scan web pages and carry out sophisticated tasks for users. This extension is named Semantic Web [56].

The challenge is to provide languages which express data as well as rules for reasoning about the data. The languages for interchanging ontological data on the Web are for example RDF, RDFS, DAML+OIL, and OWL.

In the rest of this chapter, main concepts of popular languages (RDF, RDFS, OWL) will be introduced together with appropriate querying language — SPARQL.

4.1 Resource Description Framework

The Resource Description Framework[1] (RDF) belongs to World Wide Web Consortium (W3C) specifications and represents a standard model for data publishing or exchanging on the Web. Data and their corresponding properties are expressed in the form of RDF statements (RDF triples) (s - subject, p - property, o - object) and

[1]https://www.w3.org/RDF/

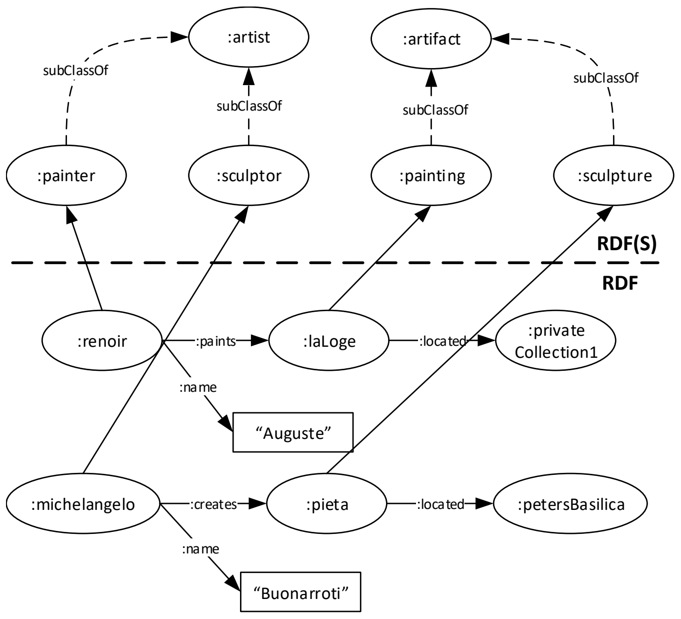


Figure 4.1: RDF(S) graph.

denote that a resource s has a property p with a certain value o. The triples form the RDF graph (See Fig. 4.1) where the subject node S has the property edge P whose value is the object node O.

RDF extends the linking concept of the Web to use Unique Resource Identifiers (URIs) to denote subject, predicate, and object. Usage of URIs allows RDF data to be mixed, exposed, and shared across different applications.

RDF also offers a possibility to represent an incomplete information using blank nodes representing unknown constants or URIs. For example, the author of B is Michelangelo, and the B is located in St. Peters Basilica, for a given and unknown resource B.

In previous paragraphs, a short summarization of basic ideas of RDF was provided. The cornerstones of RDF are as follows:

- Resources — a thing we would like to talk about, e.g., books, apples, people. Every resource is identified by a Universal Resource Identifier (URI), i.e., a URL or another unique identifier.

- Properties — they are also identified by URIs and describe relations between resources, e.g., “isDefinedBy”, age, etc.
- RDF Triples — they assert the properties of resources. Triples are in the form of (subject, property, object). A subject is represented by resources. An object may be represented by resources or literals (atomic values such as strings).

RDF triples may be serialized in various formats including XML document, N3, Turtle. An RDF triple may be perceived as a pair of linked nodes (subject node and object node linked by a property) which form an RDF graph.

4.2 RDF Schema

RDF Schema[2] (RDFS) provides a data modeling vocabulary for RDF data, and it is used to describe classes and relationships between classes (e.g., inheritance). Next, RDFS specifies also properties and corresponding relationships. Relationships may hold between pairs of properties, or between a class and property. RDFS statements are represented as triples as well, and thus RDFS forms an RDF graph. RDFS triple is called schema triple and other triples data triples.

As described, RDF and RDFS represent constructs for defining some ontological knowledge. It is possible to model knowledge in typed hierarchies, i.e., subclasses, subproperties, domain and range restrictions, and instances of concepts. On the other hand, we should be able to express more constructs which are standardized, and thus easily understandable by other data consumers. The constructs include:

- Disjointness of classes — we can say the male is a subclass of the person in RDFS. However, we would like to express that male and female are disjoint classes.
- Cardinality restrictions — constructs which allow saying how many values a property may take.
- Characteristics of properties — a possibility how to say that a property is inverse, transitive, etc.

[2]https://www.w3.org/TR/rdf-schema/

The new language (Web Ontology Language) was proposed to cover additional constructs and is described in the following section.

4.3 Web Ontology Language

Well-known and widely used extension of RDFS is Web Ontology Language[3] (OWL). OWL became a W3C recommendation in February 2004 and was proposed to overcome the weaknesses in RDF/S.

OWL uses RDF's XML syntax (labeled as RDF/XML). OWL may give readers the impression of using RDF/RDFS meaning of classes and properties and those language primitives are added to support the better expressiveness. On the other hand, RDF and RDFS have very voluminous modeling concepts such as rdf:Property and rdfs:Class. Uncontrollable computational properties may be caused because of these constructs which are very expressive. Thus, RDF and RDFS may be restricted when a trade-off between expressive power and efficient reasoning has to be established.

There are three different kinds of OWL because of the appropriate providing trade-off mentioned above. Different sub-languages are described in the following list:

- OWL Full — this kind of OWL represents the entire OWL language. This kind also offers the possibility to combine OWL primitives and RDF and RDFS. Moreover, the meaning of predefined primitives may be changed. OWL Full provides full compatibility with RDF, i.e., every valid RDF document is also valid OWL Full document. On the other hand, ontologies defined in OWL Full may be undecidable.
- OWL DL — this kind of OWL, where DL stands for Description Logic, restricts the application of constructors from OWL and RDF. The restrictions include:
  - Vocabulary partitioning — Resources are allowed to be only one of specific type, i.e., a class, a datatype property, an object property, an individual, etc. In other words, a property cannot be a datatype property and at the same time object property and vice versa.

[3]https://www.w3.org/OWL/

  – Explicit typing of resources.
  – No transitive cardinality restrictions.
  – Restricted anonymous classes.

  The efficient reasoning is preserved because of these restrictions. Furthermore, compatibility with RDF is lost. On the other hand, every valid OWL DL document is a valid RDF document.

- OWL Lite — the last version of OWL represents a restriction of OWL DL. The restrictions are for example excluding enumerated classes, disjointness of classes, and cardinality (except the values 0 or 1).

A user has a possibility to choose an appropriate OWL variety according to his needs.

4.4 SPARQL

There are two different ways how to interact with an RDF storage — a direct data access from some system using an API and with the help of user queries. In this section, query languages will be described in detail, particularly SPARQL[4].

Requirements for query languages may be described as follows [41]:

- *Expressiveness* represents a property how powerful queries may be formulated.
- *Closure* property expresses a requirement that the result of an operation is again elements of the data model.
- *Adequacy* represents a requirement for usage of all concepts of the underlying data model.
- *Orthogonality* stands for a requirement that all operations should be used independently in a given context.
- *Safety* denotes the property that syntactically correct query returns a finite set of results.

[4]https://www.w3.org/TR/rdf-sparql-query/

There are many languages for querying RDF (e.g., SeRQL [57], RQL [58], RDQL [59]). However, only SPARQL become the well-known standard for querying RDF.

The SPARQL constructs are based on matching graph patterns. The simplest graph pattern is the triple pattern which is created from RDF triple by a substitution of an RDF subject and/or object and/or predicate by a variable. An example with a variable instead of the subject is illustrated in Listing 4.1.

Listing 4.1: Simple SPARQL query.

```
SELECT ?x
WHERE
{
   ?x foaf:name "Andreas" .
}
```

Moreover, the pattern may be composed more complex by a conjunction of more patterns. Such a conjunction may represent an implicit or explicit join.

Listing 4.2: Explicit and implicit join.

```
Implicit Join:

SELECT ?x ?y
WHERE
{
   ?x foaf:name "Andreas" .
         ?x foaf:mbox ?y .
}
```

```
Explicit Join:

SELECT ?q
WHERE
{
    ?x rdf:type foaf:Person .
          ?x foaf:name ?q .
    ?c foaf:name "Andreas" .
    ?c foaf:knows ?y .
    FILTER (?x = ?y) .
}
```

Furthermore, SPARQL may represent other interesting constructs, e.g., optional patterns (previous queries represent mandatory patterns), describe queries, ask queries, and construct queries.

4.5 Summary

Ontologies are suitable for knowledge representation. However, they are unusable without an appropriate language. In this chapter, RDF, RDFS, and OWL were introduced together with querying language — SPARQL. These all methods/formats are subsequently exploited for capturing CPSs knowledge as well as the querying language for user queries in this work.

Chapter 5

Information Integration Problem

Heterogeneous data models and their integration is a challenge which we have been solving from the moment when we started to process data from various data sources or producers. Even before the invention of the first computer, people had to integrate information stored in a written form. They had to integrate information from written documents in their minds and transform them into an output. In the era of digitization, this natural and intuitive ability for the information integration had to be transferred from a human brain to computer programs.

The goal of an information integration system is to offer uniform access to a set of autonomous and heterogeneous data sources [60]. Sometimes, an information model of a data source may be covered with system interface, and thus a proper understanding of encapsulated information meaning may be harder. Integrators have to face up to an integration during a runtime, i.e., systems to be integrated are already developed or even deployed. On the other hand, the information integration may be taken into account even during a system design, i.e., a utilization of an appropriate approach and language for information integration which may facilitate understanding of given concepts. The appropriate way of knowledge representation should, for example, facilitate a scenario when a complex system is composed of many interacting components, and these components are independently developed by various developers. Furthermore, different ways of dealing with information integration may be required during an integration of previously known data models or during integration “on-the-fly” when we do not know data models in advance.

In this chapter, details about causes of data heterogeneity are provided followed by the classification of heterogeneity types and a clarification of the term semantic heterogeneity. Next, the introduction to several approaches for information representation will be provided. These approaches may more or less facilitate information integration, i.e., XML (followed by an overview of neutral formats for data integration from the industrial automation domain mainly based on XML) and a utilization of ontologies for integration (especially upper ontologies). Finally, semantic sensor network ontology (SSN) is described. SSN ontology provides means for a description of sensors and related processes and facilitates information integration of their data models and other external data sources.

5.1 Data Heterogeneity

In general, heterogeneity is a feature of all kinds of systems. This feature is both a welcome and an unwelcome feature [7]. On one hand, heterogeneity is welcomed because of close relation to system efficiency — more efficient system is more tailored to the problem. On the other hand, heterogeneity is considered as unwelcomed because it causes significant obstacles for system interoperability. This situation means non-trivial dilemma, and we need to find a balance between efficiency and interoperability.

As mentioned earlier, CPSs are integrations of computational systems, sensors, actuators, and physical processes and therefore it is needed to ensure interoperability. The goal is to reduce heterogeneity. In the following paragraphs, different categories of data heterogeneity are introduced with the focus on semantic heterogeneity. Next, ontology matching methods are described. Ontology matching methods may be used for facilitating information integration.

5.1.1 Heterogeneity Classification

There are many different classifications of types of heterogeneity, e.g., in [61, 62, 63]. In this work, the most obvious types of heterogeneity are adopted according to [64]:

- **Syntactic heterogeneity** — occurs when two data sources are not described in the same knowledge representation formalism (e.g., F-logic and OWL in the case of integration of ontologies).
- **Terminological heterogeneity** — stands for variations in names when referring to the same entity (e.g., different natural language).
- **Semantic heterogeneity** — occurs when different models are used for the same domain of interest (e.g., a utilization of different axioms for defining concepts).
- **Semiotic heterogeneity** — stands for a different interpretation of entities by people.

It is advisable to point out that several types of heterogeneity usually occur together.

5.1.2 What is Semantic Heterogeneity?

In contrast to the other types of heterogeneity, a meaning of semantic heterogeneity is not evident for many people. Furthermore, the semantic heterogeneity means a significant problem for many years and may be found in an integration of various documents, database schemas, device data models, etc. Obviously, resolving semantic heterogeneity is essential for proper and faultless information integration. We would like to make semantic heterogeneity understanding easy, and therefore possible definitions are introduced in the following paragraphs.

How to define semantic heterogeneity correctly? It is a non-trivial and not obvious task, and there is no unique generaly valid definition. However, there are already some useful definitions.

The semantic heterogeneity definition can be derived according to Merriam-Webster dictionary[1] as a quality or a state of being made up of parts that are

[1]www.merriam-webster.com/dictionary

different related to the meanings of words and phrases.

The more complex definition can be found in [65] as differences in the meaning and use of data that make it difficult to identify the various relationships that exist between similar or related objects in different components.

As mentioned, information integration is tightly coupled with resolving semantic heterogeneity of concepts of various data sources. The resolving semantic heterogeneity consists in two following things [66]:

- Determine the relationships between objects that model similar information.
- Detect possible conflicts in their representations that pose problems during the unification of shared data.

A determination of relationships between objects may be ensured (or facilitate) by exploitation of ontology matching techniques. A short overview of ontology matching is provided in section 5.1.3. A relationship between objects may be determined easily when these objects are described in an appropriate format.

5.1.3 How to Find Relations Between Entities?

One of the most challenging tasks during the information integration is to find proper relations between entities from different data models. The methods from ontology matching may be utilized for this task. For example, an adaptation of ontology matching methods for integration of an ontology and a Microsoft Excel document is described in [67]. The goal of ontology matching is to find relations between entities expressed in different ontologies [64]. The determination of relations between objects (e.g., concepts, individuals) from given data sources is an essential prerequisite for subsequent integration of the data sources. The research area of ontology matching is quite extensive, and therefore this work provides only a short overview of ontology matching. A detailed description of ontology matching may be found in [64].

There are many implemented approaches for ontology matching. Most of them are focused on automatic ontology matching, and these approaches are intended for processing a big number of elements because a manual processing could be impossible. Unfortunately, these approaches are unsuitable for utilization in domains such as medicine or industrial automation. Semi-automatic matching methods have to be used for these domains where the highest precision and recall is required. The proposed solution for semi-automatic ontology matching is introduced in section 9.

Ontology matching systems are based on similarity measures. Similarity measures may be divided into five categories:

- **String-based techniques** are methods for comparing strings related to entities for matching — name, label, comments. Examples of these techniques are prefix (suffix) similarity or n-gram.

- **Language-based techniques** are methods based on Natural Language Processing (NLP). NLP is utilized for an extraction of meaningful terms. These methods include linguistic normalization or utilization of external resources as WordNet[2].

- **Structure-based techniques** are methods which compare a structure of entities or complete data models. Example of such techniques is structural topological dissimilarity on a hierarchy.

- **Extensional techniques** are methods which exploit individuals for a comparison of entities. For example, they are based on the assumption if two concepts have a similar set of individuals then they are similar as well.

- **Semantic-based techniques** are deductive methods and are mainly used for verification of already proposed relations. These techniques include modal satisfiability techniques or techniques based on description logic.

Introduced similarity measures are suitable for different dissimilarities. Thus, it is better to consider them as building blocks of more complex solutions. One of the possibilities is to aggregate them. The various approaches may be used for similarity measures aggregation. For example, triangular norms (e.g., weighted product), multidimensional distances (e.g., Minkowski distance), machine learning methods (e.g., support vector machines).

[2]https://wordnet.princeton.edu

5.2 XML

The creation of a standard way for publishing data on the Web was forced by growing needs of the advanced Web technologies. Files published on the Web are usually HTML files with a predefined structure that enables rendering of their contents in a Web browser. The process of a file publishing can be divided into two steps [68] — a creation of a file by a user; a publishing of this file by sharing its URL.

XML is eXtensible Markup Language that is used for transferring data on the Web and has been accepted as a W3C Recommendation in 1998. XML documents are used to store data on the Web, and their content is structured in nested tags. An opening and a closing tag delimit a particular content (called an element), and each tag can be supplement with a set of additional name-value pairs, called attributes. These tags are defined by a user, and an XML document can be supplied by a document that specifies the allowed tags and their structure — XML Schema[3]. In other words, XML Schema defines constraints on XML documents. It provides simple vocabulary and predefined constructs for modeling relations among entities.

Furthermore, XML format is widely accepted and used due to its relatively uncomplicated structure and easy processing. Based on these characteristics, XML format could be understood as a universal format for data exchange and even for data storage.

The XML format is important technology which has been used for information integration in many applications and domains. It was caused by simple and powerful syntax which is versatile enough for information sharing from multiple sources. On the other hand, XML does not address issues of the semantic integration. For example, XML files may be shared with many systems, but they are meaningless outside the application (i.e., without a right context).

The importance of XML for data exchange and integration is obvious from the following section — "Neutral Formats for Integration of Industrial Data", where a brief overview of the most successful formats for a sharing and integration of industrial data is provided. A prevalent part of formats is based on XML. From another point of view, the XML-based formats try to add (more or less successfully) some domain-specific vocabulary as well as constructs for expressing relations between

[3]https://www.w3.org/standards/xml/schema

concepts.

5.3 Neutral Formats for Integration of Industrial Data

As previously mentioned, if we want to obtain a required information then we have to integrate multiple heterogeneous data sources. It is a problem in many domains including industrial automation.

Data integration could be defined as a problem of creating a uniform query interface over data from heterogeneous data sources. It is obvious that the problem is not only in data querying but also in coping with semantic heterogeneity among data sources. A process of data integration is a very complex issue, and therefore it is important to facilitate this process by offering some neutral re-usable data format as well as a way how to handle and manage data.

There are some neutral formats for particular applications. For example, neutral formats for 3D visualization include the 3D XML format [69] developed by Dassault Systemes and the JT format [70] currently owned by Siemens PLM Software.

Next, more important (from our point of view) are formats with the aim to general and versatile information description or integration without any specific specialization. The first and widely used is a utilization of XML with a proprietary schema. In this case, a use of a proprietary schema for data integration makes sense in the scope of for example a company or an initiative which design a given schema. On the other hand, formats such as AutomationML, OPC UA information model, and PLM XML are standardized and contain an additional semantics in contrast to a general XML. Therefore, these formats are used by various companies and are discussed in the following paragraphs.

5.3.1 AutomationML

AutomationML is an XML-based format with the objective to enable seamless automation engineering of production plants [71]. This standard was developed as neutral data exchange format of manufacturing systems by a consortium of leading vendors and users of automation technologies (ABB, Siemens, Rockwell Automation, Kuka, etc.).

The AutomationML format is based on following standards:

- **COLLADA** [72] standard is used for geometry and kinematic modeling.
- **PLCopen** [73] describes plant behavior and control as a sequence of actions.
- **CAEX** [74] standard is the cornerstone of the hierarchical structure of plant objects.

The AutomationML architecture is depicted in the Fig. 5.1.

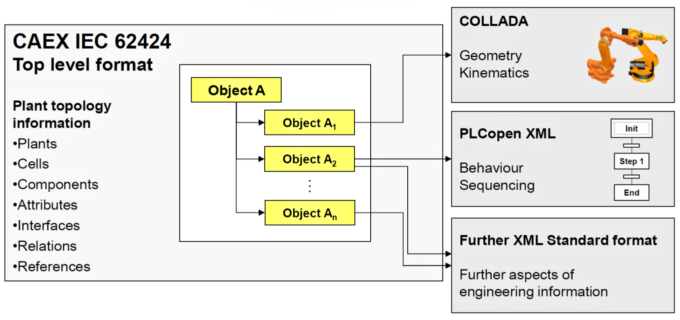


Figure 5.1: Architecture of AutomationML, adopted from [1].

The characteristic feature of the format is the application of CAEX concepts. The following elements are defined by AutomationML:

- **Interfaces** describe relations between objects. There is predefined interface library in AutomationML.
- **Roles** or role classes describe the functionality of a CAEX object within a given context.
- **System units** contain user defined AutomationML classes.
- **Instance hierarchies** store data of concrete project. A hierarchy includes object instances together with properties, interfaces, references, and relations.

AutomationML provides relatively universal architecture how to capture information including for example device concepts such as a sensor or an actuator unit class.

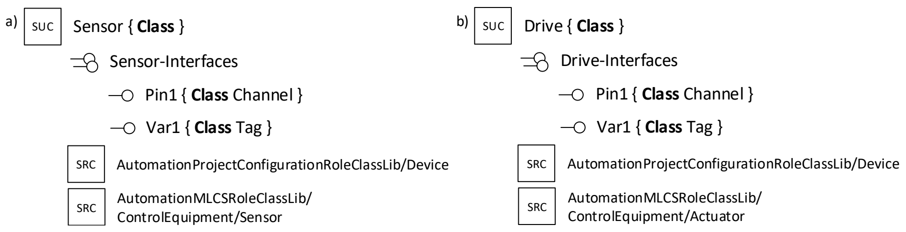

a)
SUC
Sensor { Class }
Sensor-Interfaces
Pin1 { Class Channel }
Var1 { Class Tag }
SRC
AutomationProjectConfigurationRoleClassLib/Device
SRC
AutomationMLCSRoleClassLib/ControlEquipment/Sensor
b)
SUC
Drive { Class }
Drive-Interfaces
Pin1 { Class Channel }
Var1 { Class Tag }
SRC
AutomationProjectConfigurationRoleClassLib/Device
SRC
AutomationMLCSRoleClassLib/ControlEquipment/Actuator

Figure 5.2: Recommendation how to represent a) sensor b) actuator in AutomationML.

A sensor may be represented according to [75] as follows: the corresponding SystemUnitClass contains one Channel interface to model the physical output and one Tag interface to model logical value of the sensor state. The class sensor has the role Device and thus contains inherited attributes, e.g., Name and Type. The sensor class is illustrated in Fig. 5.2.

An actuator may be modeled as a Drive class [75]. The Drive SystemUnitClass contains one Channel interface to model the physical input to the actor and one Tag interface to represent a logical value of the actor state. The Drive class has the role Device similarly as the Sensor class. The drive class is illustrated in Fig. 5.2.

5.3.2 PLM XML

PLM XML format was developed by Siemens PLM Software and is aimed to improve the interoperability of product lifecycle data [2]. The PLM XML schema covers following data characteristics (element types):

- **Product structure** — the format supports both configured product structure (represented by instance graph together with occurrence trees — product views) and unconfigured product structure.
- **Metadata** — the format allows storing various metadata such as an annotation.

- **Geometric representation data** — the format comprises geometric information including points, curves, surfaces, and coordinates. A construction geometry can be represented as well.
- **Data ownership** — the format takes access control into consideration.
- **Visualization properties and features** — the format provides a schema for representation of visualization properties (view directions, ports, and characteristics) and feature description.

Except for data elements, PLM XML provides a delta schema (define a difference between two product structures) for facilitating change management. The XML schema of the format allows extensions to derive new elements from existing ones.

Next benefit of employing PLM XML is a possibility to share high content product data with a community of adopters and enabled applications via PLM XML pipeline [2]. An example of a flexible data sharing is shown in Fig. 5.3.

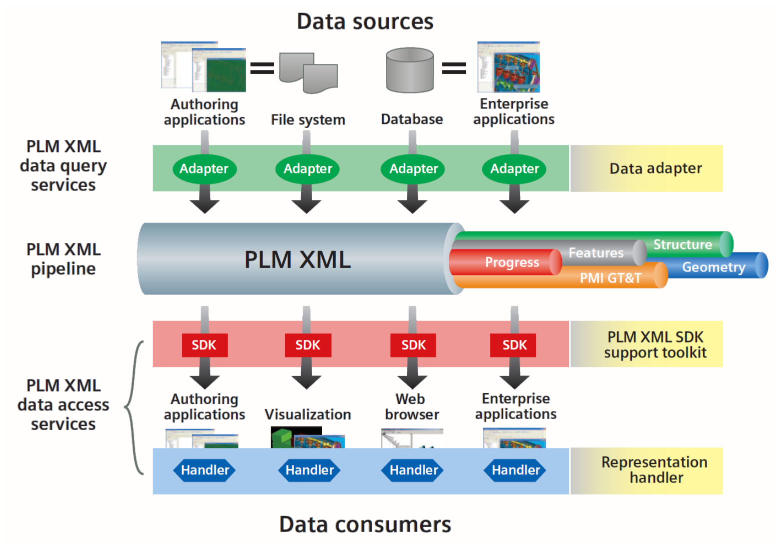


Figure 5.3: PLM XML information integration and sharing by means of PLM XML pipeline, adopted from [2].

Furthermore, PLM XML also describes devices such as sensors. XSD schema for a sensor model is shown in Fig. 5.4. Even though this standard offers quite extensive structures for representation of devices but it may be found not flexible enough in comparison to the following formats — AutomationML and OPC UA Information Model.

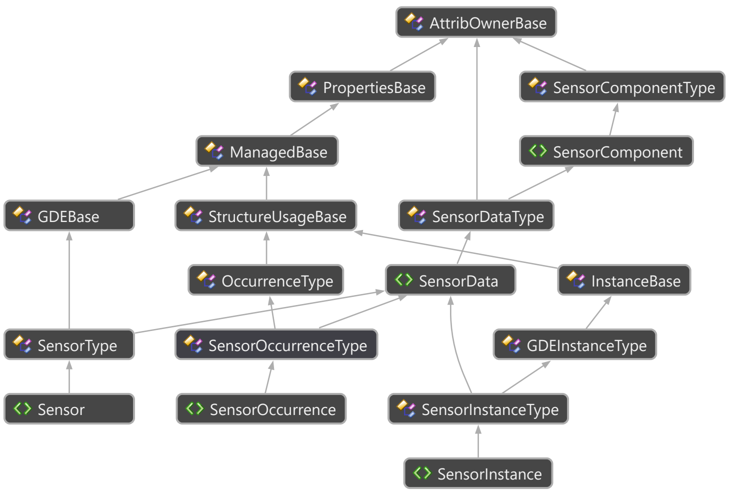


Figure 5.4: XSD schema for a sensor model within PLM XML format.

5.3.3 OPC Unified Architecture Information model

The next very interesting way how to model and even integrate data is by means of OPC Unified Architecture (UA) standard [76]. In general, OPC UA is a secure and open mechanism for exchanging data between servers and clients in industrial automation. OPC UA standard tries to overcome the main obstacle of its predecessor (OPC Data Access together with OPC Historical Data Access and OPC Alarm&Events) — COM[4] dependency of OPC. Therefore, the OPC UA was designed for replacement of all existing COM-based specification to be platform independent with extensible modeling capabilities.

OPC UA is built on two main components [77] — transport mechanisms and data modeling. The transport component offers the possibility to communicate via optimized binary TPC protocol for high-performance intranet communication and

[4]https://www.microsoft.com/com/default.mspx

next possibility to communicate via Web Services. The data modeling component represents rules and building blocks for a creation and exposing information model. It also defines base types to build a type hierarchy.

In original OPC standard, only "raw" data is exchanged, i.e., there was poor information included understanding the semantics of provided data — tag name and some information like engineering unit. In contrary, OPC UA offers more flexible possibility to expose the semantics of the data because of complete object-oriented capabilities including type hierarchies as well as inheritance.

The OPC Foundation has started with standardization of information models of various devices (UA Devices) for the unification of models. Every device vendor may extend these base models with vendor-specific information. This approach is also assumed in other scenarios, e.g., providing data of MES or ERP systems by exposing the ISA 95 model [78].

There are many interesting features described in OPC UA specification — triggering of methods, variable subscriptions, security, device discovery (local as well as global), etc. Because of these features, OPC UA seems to be on of the most promising frameworks for data representation and exchange in automation domain.

5.4 Ontologies for Integration

Ontologies become in recent decades an essential component of many applications in various domains, e.g., natural language processing [79], robots [80], industrial automation [81], and scheduling [82]. The ontologies represent knowledge of the domain of the given application. For many years, relational databases were employed to represent data and partly relevant knowledge in legacy applications. Furthermore, databases will be exploited in some applications in the future ceaselessly. When considering databases, a big portion of knowledge is not stored explicitly, but it is coded in algorithms. Thus, such an approach means a difficult understanding of the knowledge and data, and it could lead to an inconsistent interpretation and use of data. On the other hand, ontologies try to overcome these deficiencies and are supposed to represent more complex and sometimes possibly incomplete models.

We have mentioned that information integration problem may be defined as a creation of a unified querying interface. In other words, there are various systems having different knowledge representation, and we need to acquire relevant information from data sources for a given application.

Thus, the important problem which is related to the mentioned definition is a knowledge representation problem [60]. It means how knowledge is represented in various data sources and how to enable and facilitate their integration. If an architect designs a system and a data model of the system is planned to be used in other systems, then he should take into consideration how to represent data for facilitating the data model integration.

The knowledge representation should offer an adequate expressivity for modeling of given context as well as their relations. Furthermore, a knowledge representation language should facilitate the process of concepts matching and subsequent mapping for information integration. From this point of view, a utilization of ontologies seems to be a suitable approach for knowledge representation. When an ontology is well-designed then modeled concepts of the ontology may be unambiguously understood even without context. Moreover, a utilization of Upper Ontologies should facilitate the process of integration.

5.4.1 Integration based on Upper Ontologies

Providing well-designed and substantial ontologies which stand the test of large application scenarios is a current bottleneck in SemanticWeb research and application development. According to primary intention, the Semantic Web should facilitate a search for suitable ontologies, integrate them with few simple changes and exploit them within a given application. The number of available ontologies is increasing, but well-designed ontologies are rarely available. A utilization of upper ontologies for information integration is not limited only to an integration of ontologies but may be a mean for integration of data sources represented in various formats.

There are many solutions which adopted the methodological approach which utilizes an abstract foundational ontology to facilitate domain ontology integration, e.g., the SmartWeb project[5].

Why to use an upper ontology? The problem may arise when we need to integrate multiple independently developed ontologies. Concepts of ontologies

[5]http://www.smartweb-project.org

and their properties have to be defined precisely with an explicit ontological commitment to avoid semantic ambiguities during integration. Upper ontologies may be understood as axiomatic theories about the high-level as well as domain-independent categories in the real world, e.g., physical object, social object, event, process, etc. The major advantages of an upper ontology employment are as follows [83]:

- Conceptual clarity — Upper ontologies provide a reference point for proper comparison among different ontological approaches and a framework for integrating existing ontologies.
- Design patterns — Ideally, an upper ontology defines "ontology design patterns" for re-occurring modeling needs.
- Modeling basis — Upper ontologies may be understood as guidelines for building a base of new ontologies, instead of modeling from scratch.

On the other hand, understanding of upper ontologies may be difficult because of their abstract nature. Furthermore, a philosophical background is sometimes needed for the proper understanding. The first essential step is to choose the best fitting upper ontology from available ones [84], e.g., BFO, Dolce, OpenCyc, and SUMO.

According to [83], DOLCE and SUMO ontologies are the most promising ones. One of their advantages is the fact that they both meet the following requirements:

- Descriptive requirement — A descriptive ontology tries to capture commonsensical notions based on natural language utilization and human cognition.
- Multiplicative requirement — A multiplicative ontology tries to provide a reliable description of reality by thanks to co-localization different entities in the same spatiotemporal coordinate.
- 4D paradigm (also known as perdurantism) — this requirement assumes that entities extend in space and in time, i.e., entities have both - spatial and temporal parts.

Suggested Upper Merged Ontology — the SUMO is ontology owned by IEEE and available under GNU General Public License. The technical editor of the ontology is Adam Pease.

This ontology contains about 25,000 terms and about 80,000 axioms. This ontology was based on an integration of different ontologies and theories [85]: a formal theory of holes, formal mereotopology, the ontology of boundaries, Process Specification Language, upper ontologies, etc.

The most general concept in the SUMO ontology is *Entity* which is subsequently specialized in *Physical* and *Abstract* concepts. The taxonomy is very extensive. Except concepts concerning industrial automation, there are concepts such as *Hotel*, *Organization*, etc.

SUMO may be difficult to handle because of the enormous number of concepts, their relations as well as many axioms. From the other hand, SUMO's rich taxonomy may be beneficially utilized for modeling domain ontologies.

DOLCE — DOLCE is a part of the WonderWeb library of foundational ontologies [86]. This ontology was successfully used in various domains, e.g., law [87] or agriculture [88].

The term DOLCE abbreviates Descriptive Ontology for Linguistic and Cognitive Engineering. DOCLE is aimed to differ enduring and perduing entities. The main relation between Perdurents (i.e., objects) and Endurants (i.e., events or processes) is that Endurant lives in time by participating in Perdurants. For example, a product (endurant) participates in his production process (perdurant).

DOLCE is aimed at capturing concepts underlying human common sense. Furthermore, DOLCE comprises following advantages — possibility to model 3D (Endurants) and 4D (Perdurants) objects. According to [83], DOLCE is conceptually sound and suitable for reference purposes. On the other hand, a DOLCE exploitation may be difficult because of smaller taxonomy compared to SUMO and its abstract nature may be difficult to handle for some users.

5.5 Semantic Sensor Network Ontology

Neutral formats were introduced in preceding paragraphs. Another possibility is to utilize an existing ontology for a representation of devices such as sensors, etc. In this section, Semantic Sensor Network (SSN) ontology is introduced and this ontology may be utilized for a description of sensing devices as well as related processes.

Since 2002, the Open Geospatial Consortium, more precisely its Sensor Web

Enablement initiative, started with the development of a generic framework for exchanging sensor data, enabling remote-sensing, and in-situ sensing. The developed Sensor Observation Service provides a query interface for observation and sensor data. The output of service is in the form of Ontology&Measurements (O&M) [89] (formerly known as OMXML) and Sensor Model Language (SensorML) [90].

SensorML and O&M represent a different view on data. SensorML is *provider-centric* — it provides information about sensors and raw observation data. SensorML is designed to support serialization of numeric data arrays and is optimized for multiple parallel streams that must be processed together. On the other hand, O&M is more *user-centric* with the focus on the observation and observed property objects. It describes data at a higher level of semantics than SensorML. O&M provides abstract classes for sensors, features of interest, and observable properties. The complementary details are expected to be added by specific applications and domains.

The SSN ontology draws on O&M and SensorML. It is based on concepts of systems, processes, and observations. It offers possibility to describe physical as well as processing structure of sensors. The SSN ontology is based on the ontology design pattern named the Stimulus-Sensor-Observation pattern [91]. The SSO was designed as the corner-stone for heavy-weight ontologies for the Semantic Sensor Web applications. This pattern is also aligned to the Dolce Ultra-Lite ontology[6].

The architecture of SSN ontology together with the dividing to modules is illustrated in Fig. 5.5.

SSN ontology is composed of several modules. The module Skeleton represents the essential conceptualization as a lightweight and minimalistic ontology with a minimal ontological commitment. This part includes the main concepts such as Sensor, SensorOutput, Observation, SensingDevice, and Sensing. Next, the module Process represents processes together with their inputs and outputs. Besides of the main modules, SSN is also composed of following modules — MeasuringCapability, ConstraintBlock, Device, OperatingRestriction, System, Deployment, PlatformSite, and Data.

Nowadays, a new version of SSN ontology is being developed. The main differences are — it also involves Actuators concepts; it is not built on DUL ontology; the

[6]http://www.ontologydesignpatterns.org/ont/dul/DUL.owl

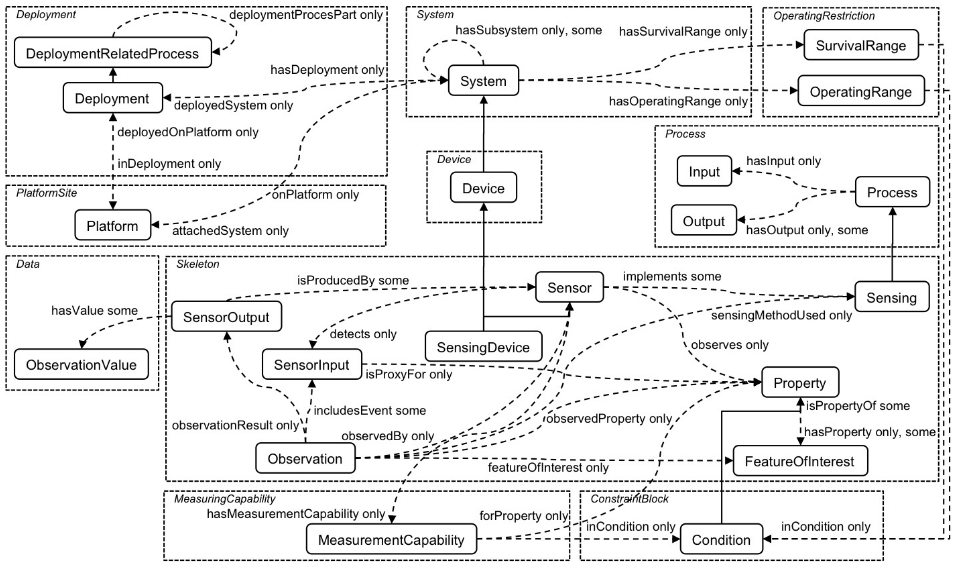


Figure 5.5: SSN key concepts and their relations, adopted from [3].

proposal of the new version (SOSA) is intended to be more lightweight (inspired by Linked Open Data). On the other hand, SOSA ontology is still only W3C Candidate Recommendation.

5.6 Summary

In this chapter, the problem of an information integration was introduced. If we need to integrate different data sources, then the biggest obstacle is a heterogeneity between them. The categorization of various types of heterogeneity is provided, and the semantic heterogeneity was emphasized as the most challenging heterogeneity is the semantic heterogeneity. Next, ontology matching methods were described with the focus on their exploitation for a reduction of semantic heterogeneity during the information integration.

The next part tackles the problem of knowledge representation formats which is a complementary task to the identification of relations between entities. Two possible approaches were considered; the first one is the utilization of a (neutral) formats prevalently based on XML and a corresponding XML schema. On the other hand, an employment of ontologies should benefit from a more syntactically and semantically richer language in comparison with XML-based formats. The information integration based on upper ontologies were introduced together with a short overview the most promising upper ontologies — DOLCE and SUMO. Moreover, SSN ontology which defines important concepts for a description cyber-physical systems (sensors, observations, processes, etc.) was described.

The utilization of ontologies, a representation of a big volume of data in RDF statements more precisely, may cause significant performance issue. This problem may be resolved by a utilization of a suitable technology. In the following chapter, the Big Data paradigm and frameworks which are intended to facilitate processing of heterogeneous data are briefly described.

Chapter 6

Big Data

Semantic technologies represent a very capable way how to enrich conventional automation systems with additional information for competitive data processing. A feasibility of semantic technologies adoption within automation domain consists in the availability of suitable means. Even though there are already frameworks for handling and storing RDF triples, these frameworks have a significant limitation — insufficient performance. It is worthwhile to mention that a suitability of a given system for data handling and storing is related to a specific application. However, the best system is able to handle all possible data types, i.e., big data as well as common datasets.

Another specific feature of (not only) big data applications is batch and streaming data processing. The significant difference of these approaches is dataset availability during computing. Batch data processing assume an availability of the whole dataset for algorithms. On the other hand, streaming processing treats data as it enters a system.

In this section, a description of big data and their specific characteristics is provided. Next, big data processing frameworks are introduced.

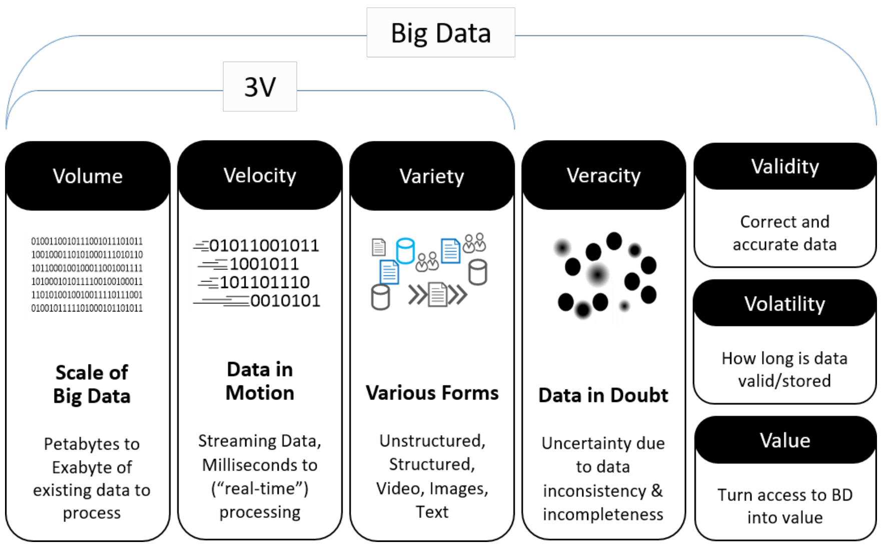

Big Data
3V
Volume
Scale of Big Data
Petabytes to Exabyte of existing data to process
Velocity
Data in Motion
Streaming Data, Milliseconds to ("real-time") processing
Variety
Various Forms
Unstructured, Structured, Video, Images, Text
Veracity
Data in Doubt
Uncertainty due to data inconsistency & incompleteness
Validity
Correct and accurate data
Volatility
How long is data valid/stored
Value
Turn access to BD into value

Figure 6.1: Big Data characteristics — 3V(5V/7V) definition.

6.1 Big Data Definition

What represents the term big data? Unfortunately, there is no one widely accepted definition of big data. In general, the term big data is used for datasets that are growing so that it becomes difficult to manage them using existing database management concepts and tools. According to [92], big data is data that exceeds the processing capacity of conventional database systems. The data is too big, moves too fast, or does not fit the structures of database architectures. To gain value from this data, developers must choose an alternative way to process it.

Widespread "definition" (or description) of big data is with the help of their main characteristics. In origin, it was three characteristics (velocity, variety, volume) and therefore the definition is known as 3V. Furthermore, additional characteristics have been added in the course of time. The additional characteristics are veracity, validity, volatility, and value. 3V definition is illustrated in Fig. 6.1 together with additional characteristics.

A vagueness of 3V definition leads to doubt — what is big data framework and what is not? This problem was solved by NIST. The National Institute of Standards

Table 6.1: NIST – Big Data characteristics.

Volume	Velocity	Variety	Requires Horizontal Scalability	Relational Limitation	Big Data
No	No	No	No	No	No
No	No	Yes	No	Yes	Type 1
No	Yes	No	Yes	Maybe	Type 2
No	Yes	Yes	Yes	Yes	Type 3
Yes	No	No	Yes	Maybe	Type 2
Yes	No	Yes	Yes	Yes	Type 3
Yes	Yes	No	Yes	Maybe	Type 2
Yes	Yes	Yes	Yes	Yes	Type 3

and Technology (NIST) introduced big data characteristics and taxonomy [93] which may solve the previously mentioned issue. There are three main types of big data:

- *Type 1* — represents a problem where non-relational representation is required for effective processing.
- *Type 2* — represents a problem where horizontal scalability is required for effective analysis.
- *Type 3* — represents a problem where non-relational representation, as well as horizontal scalability, is required for effective analysis.

The table 6.1 (introduced by NIST) is derived from the *3V definition* and the big data types. The table can answer the question — are the given data of CPSs Big Data?

6.2 Big Data Processing Frameworks

In many cases, big data processing cannot be managed using legacy systems. Several big data frameworks have been already developed, and they will be introduced shortly in this section. Frameworks differentiate from each other by their focus. There are three different types of frameworks — batch-only, stream-only, and hybrid frameworks. The common feature of all big data processing frameworks is a distributive nature.

6.2.1 Batch Processing Frameworks

Batch processing frameworks are focused mainly on operating over a large and (partly) static datasets. From the nature of batch processing approach, these systems are suitable for processing of datasets which are finite and extremely large. Thus, batch processing frameworks are frequently used to treat with historical data.

One of the most well-known frameworks is Apache Hadoop[1]. This framework is a representative of batch-only frameworks. Hadoop was based on Google File System [94] and data processing algorithm proposed by Google — MapReduce [95]. Advantages of this framework are scalability, reliability, and flexibility. In contrast to many proprietary systems, Hadoop is open source software and runs on low-cost commodity hardware. Important for a deployment is also a set of available expanding tools/libraries. Additional tools[2] for Hadoop are HBase, Hive, Mahout, Pig, Zookeeper, etc.

6.2.2 Stream Processing Frameworks

Stream processing frameworks treat data as it enters a framework deployment. These systems require different processing approach then batch processing framework. In other words, stream processing frameworks use operations that are applied to individual data samples. These operations are applied only on one or very few data (micro-batches) samples.

Apache Storm[3] is a framework which is aimed at extremely low latency processing (near real-time). The processing algorithm utilizes orchestrating DAGs (Directed Acyclic Graphs) called topologies. Topologies describe transformations and operations that are designated for incoming data samples.

Apache Samza[4] framework benefits from Apache Kafka messaging system[5], and

[1]http://hadoop.apache.org
[2]http://wiki.apache.org/hadoop
[3]http://storm.apache.org
[4]http://samza.apache.org
[5]https://kafka.apache.org

this architecture results in fault-tolerant processing and well-managed buffering system. Moreover, Samza uses Hadoop resource management — YARN (Yet Another Resource Negotiator).

6.2.3 Hybrid Processing Frameworks

Some frameworks are able to handle both — batch and stream processing. These systems may be described as their focus is to be general solutions for data processing.

Apache Flink[6] framework considers batches to be data streams with finite range. In other words, batches are considered as subsets of data streams. This approach (primarily focused on streams) are called Kappa architecture. Flink data processing model handles data on an item-by-item basis, i.e., true data streams. An interesting feature is providing snapshots at set points during processing to enable recovery in case of problems.

Apache Spark[7] framework was inspired by Hadoop's MapReduce engine. It differs from Hadoop in offering full in-memory processing. Spark may be deployed as standalone, YARN, or Mesos[8] cluster. While standalone configuration is suitable for small clusters, YARN and Mesos should be used for large clusters. Similarly to Apache Storm, Spark utilizes DAG for representing all operations that must be performed. Spark is primarily focused on batch data processing, and data streams are handled as micro-batches.

6.3 Summary

Big data paradigm becomes very popular in the last decade. Many researchers and developers strictly insist on fulfilling 3V/5V/7V big data definition. However, applications have various characteristics and may be specified according to NIST specification. Thus, an application domain is wider for big data processing frameworks than according to the 3V definition.

Introduced systems represent frameworks covering data batches and data streams processing including hybrid systems. Unfortunately, there is no system which fits

[6]https://flink.apache.org
[7]https://spark.apache.org
[8]http://mesos.apache.org

for every problem or application. For example, Hadoop can manage extremely large datasets but is slow in comparison to Spark. On the other hand, Spark is not able to handle such extremely large datasets and may be considered as more expensive because of required memory hardware in Spark cluster in comparison to commodity hard disks. Therefore a suitability of specific framework utilization should be assessed according to a given application.

In this work, several big data frameworks were tested during conducted experiments, and Apache Spark was found as a best fitting because of the nature of CPSs and Semantic Big Data Historian.

Part III

Semantic Integration in the Context of Cyber-Physical Systems

Chapter 7

Integration Challenge of Cyber-Physical Systems

As mentioned in previous chapters, heterogeneity is a common feature of many systems including cyber-physical systems. In this chapter, the description of the cyber-physical system integration problem is provided, i.e., the problem of cyber-physical system components integration (low-level integration) as well as an integration of cyber-physical systems (high-level integration) into a more complex whole with the focus on their semantic heterogeneity. Nowadays, a common integration of system components relies on *ad hoc* solutions. These solutions can provide very effective systems. However, they may bring many drawbacks — difficult system maintenance, malfunction corrections, adding or adjusting components, re-usability, etc. In the following chapters, the possible solution, how to overcome these drawbacks as well as how to exploit ontology matching methods to facilitate an integration of a new device, is presented. As mentioned in the chapter 2, a cyber-physical system may be a complex system which consists of many sub-parts produced by various manufacturers. Moreover, cyber-physical systems are typically integrated into a more complex system for an improvement of their capabilities — Internet-of-Things, Smart Factories, etc.

Whether we talk about sub-parts of a cyber-physical system or an integration of whole cyber-physical systems, every part maintains specific data model which is

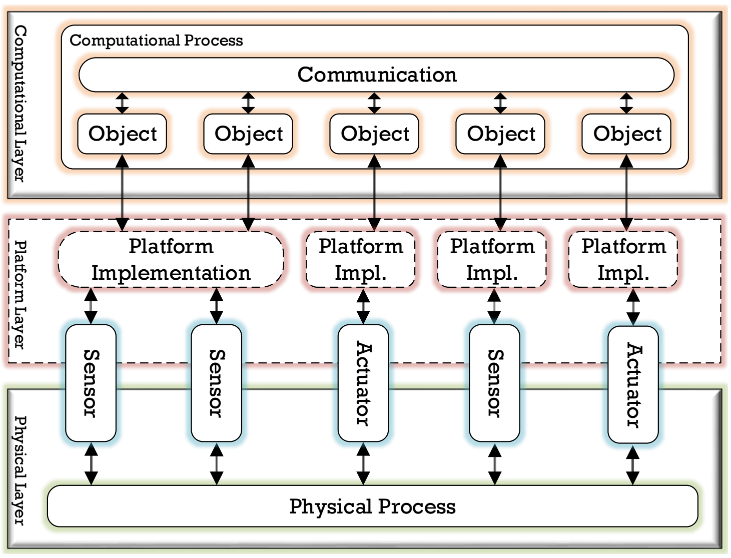


Figure 7.1: Cyber-physical system architecture.

derived from the nature of corresponding physical process or processes. These parts provide data via an interface to other components or other systems. Reversely, they consume data from surrounding systems for an enhancement of their behavior. Furthermore, systems can share joint data storage — local/distributed/in a cloud.

As mentioned, the integration problem can be divided into two distinct problems corresponding to various perspective, and they are introduced in the following sections.

7.1 Low-level Integration

A cyber-physical system is rather a general term. In the simplest case, it may be one sensor, related algorithm, and feedback to the physical process by appropriate modalities (for example employing an actuator). This simple case may be represented by a smartwatch with a pulse sensor and a possible feedback via vibrating. On the other hand, a cyber-physical system may be represented by a very complex system, for example in an aircraft [96] may be used for applications where security and safety-critical aspects are important, or we need predictability in the face of

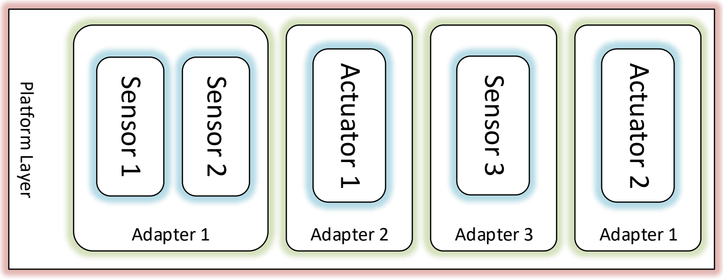


Figure 7.2: Platform integration by means of corresponding adapters.

dynamic environments. All types of cyber-physical systems are based on interoperability of their components. In other words, we face the problem of integration of these components. In following paragraphs, the problem of cyber-physical system components integration concerning system architecture is defined.

The low-level integration represents interconnections among components of a cyber-physical system — sensor(s), data model(s) of the computational process, and an actuator(s). Sample cyber-physical system architecture which is relevant for the low-level integration is illustrated in Fig. 7.1. According to the figure, a cyber-physical system consists of a physical part and a cyber part (represented by a platform and a computational layer). The physical part involves the physical process and physical objects which provide a possibility for process control, e.g., sensors.

The cyber part can be divided for clarity into two layers — a platform layer and a computational layer. The first layer (*Platform Layer*) is responsible for enabling and subsequently ensure proper operation of physical components. As depicted in Fig. 7.2, a technology heterogeneity is usually solved with the help of ad-hoc adapters which serve as interfaces between a cyber-physical system operational platform and physical components. For example, an adapter may be in the form of an OPC UA client, etc. Furthermore, many of systems (not only cyber-physical systems) are limited to a subset of supported devices which are relevant to a given set of adapters.

Next, the second layer (*Computational Layer*) represents the computational process which is able to control the physical process according to an implemented logic. In many cases, the computational layer may face a syntactic heterogeneity issue when it processes data from sensors as well as interacts with actuators. The syntactically heterogeneous sensors, internal cyber-physical system data model, and actuators occur when corresponding data models are represented in different formats — for example, OPC UA and MTConnect[1]. The solution of this issue is similar to the platform heterogeneity, i.e., by means of an appropriate adapter. An example is depicted in Fig. 7.3.

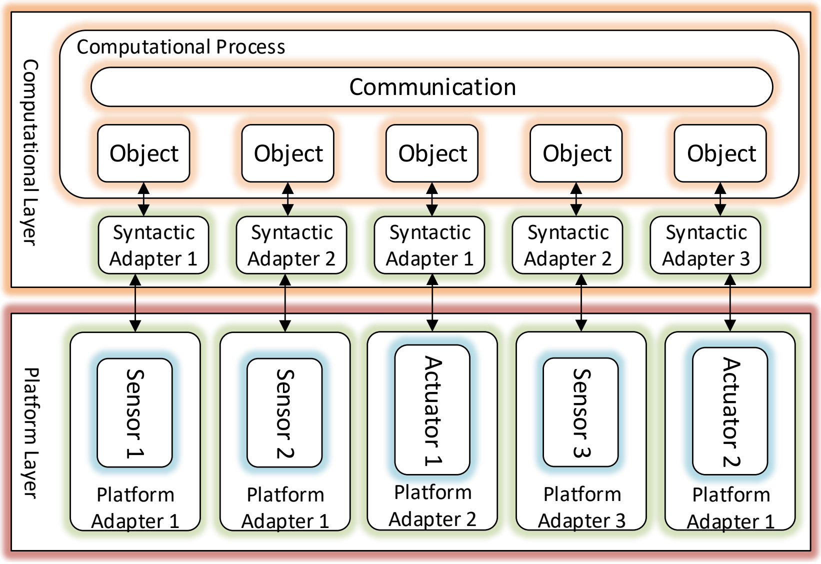


Figure 7.3: Syntactic heterogeneity within cyber-physical system.

If the platform and the syntactic heterogeneity have been solved, then the last obstacle to be able to process data successfully could be semantic heterogeneity. Unfortunately, this kind of heterogeneity cannot be solved easily by some adapter which may be used as a generic solution for all similar problems.

For better understanding, how semantic heterogeneity arises within a cyber-physical system, it is convenient to describe its modeling process. The physical process is modeled first with a physical layer abstraction. Then, the corresponding control system is implemented using a computational (software) layer abstraction. Finally, the control system is deployed on the computation platform modeled with the platform layer abstraction. The different abstraction layers use typically non-

[1]http://www.mtconnect.org/

compatible semantics, which is the cause of semantic heterogeneity.

An example of the cyber-physical system may be considered cabin pressurization and control system which is responsible for regulating the pressure in the aircraft cabin during a flight. There is strict plan how to control cabin pressure [97]. A relation between ambient pressure and cabin pressure is used for regulating outflow pressure valve. A designer of this control system may have two different data models where the pressure sensor is modeled (a model of ambient and cabin pressure sensor), but it may not be a clear detailed specification. Next, the designer has to properly know which is the ambient pressure sensor and which is cabin sensor. In the worst case, the designer could find in the data models some meaningless labels.

Unquestionably, the semantic heterogeneity symbolizes significant obstacle for faultless integration of cyber-physical system components.

7.2 High-level Integration

Although this dissertation thesis is focused on the low-level integration, the high-level integration problem is introduced in this section for the sake of completeness. Additionally, a complex system composed of networked cyber-physical systems shares many of the main characteristics of the low-level integration problem.

High-level integration denotes interconnections of various cyber-physical systems to form for example IoT. The integration of CPSs is depicted in the Fig. 7.4. This problem may be solved by the integration process which is discussed in detail in the following paragraphs mainly aimed to low-level integration.

The example of the high-level integration is presented by Smart Living[2] with *Personal IoT* solution. The combination of components consists in the gateway called ATT IOT. A user has to write a script to automate connected components via the Application Programming Interface (API). This approach solves a platform heterogeneity but does not provide any information about a data meaning.

[2]http://smartliving.io

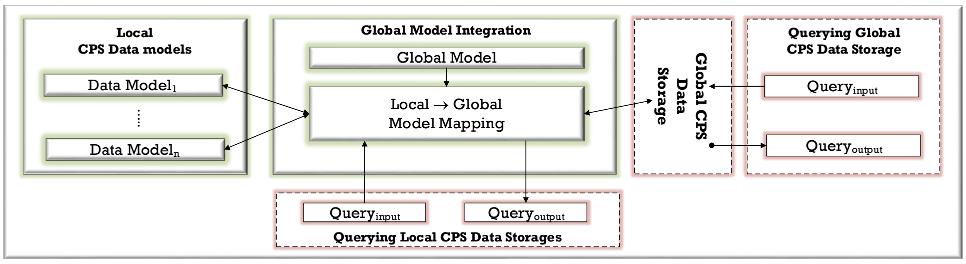


Figure 7.4: High-level integration of cyber-physical systems.

7.3 Integration Process

In following paragraphs, the introduced cyber-physical system integration problem is summarized. "Simply speaking", the integration task consists in the unification of interfaces of devices (high-level integration — cyber-physical systems; low-level integration — sensors, actuators) as well as the unification of corresponding data models. In the following enumeration, the main individual problems are stated together with possible solutions.

1. ◦ *Platform heterogeneity* — the first problem of cyber-physical system component integration lies in the platform heterogeneity. It is caused by different devices used for formation of a more complex system produced by various manufacturers.

   ◦ *Possible solution* — a unification of different interfaces provided by various manufacturers with the help of adapters.

2. ◦ *Syntactic heterogeneity* — platform non-heterogeneous systems may still have problems with an integration. If the system components should communicate within a particular platform, they still may have problem in various format for data representation, i.e., syntactic heterogeneity.

   ◦ *Possible solution* — a unification of different formats used for communication with the help of adapters.

3. ◦ *Semantic heterogeneity* — The last problem may be represented by different data models used by cyber-physical system components, i.e., the

same real-world entities are represented by different concepts, or the same concepts denote different real-world entities.

∘ *Possible solution* — models integration covers the identification of corresponding concepts, relations among concepts and their meaning in a given context. Ontology matching approach may be utilized for elements identification. Next, rules, how to transform data of components to the shared model, have to be modeled.

The low-level, as well as the high-level integration, may be successfully solved with the help of a global model. The model has to map all particular data models and provide proper relations among data concepts. This data model together with implemented transformations acts as a united interface for direct user queries or interaction with other systems in the first case or acts as transformation component for subsequent data storing in global cyber-physical system data storage.

The suitable solution for this problem is ontologies. In the following chapters, a solution for the low-level CPS components integration and the semi-automatic ontology matching system for identification of corresponding entities will be introduced. On the other hand, this solution has a significant deficiency — performance problem. Therefore, this work will show how this problem may be solved by an exploitation of big data technologies.

7.4 Summary

In this chapter, the integration challenge of CPSs was introduced. Next, two different types of the integration were defined — low-level and high-level integration. Subsequently, the general process, how to resolve the integration, was proposed.

Solutions for individual steps of the integration process were identified. The platform and syntactic heterogeneity may be solved by using an appropriate adapter. On the other hand, the semantic heterogeneity means the biggest problem and appropriate solution is application-dependent.

Chapter 8

Shared Ontology for Integration

For dealing with semantic heterogeneity, Semantic Web technologies were chosen for building a suitable solution. Semantic Web Technologies represent one of the promising ways how to ensure faultless interoperability as discussed in chapter 4.

There are various approaches how to face a problem of a representation of a cyber-physical system components data model. According to [98], the solution, where a shared ontology represents given concepts from system components, was chosen instead of having several different distributed data models. It is apt to remark the solution using shared ontology may facilitate an integration of components data models describing concerned area from different perspectives, levels of granularity, or coverages. It is because of the flexibility and high expressivity of ontologies.

This dissertation thesis is mainly about improving interoperability as well as reusability not only within a limited scope of a particular solution but among various devices for example deployed in a cloud by some external provider. Furthermore, a significant contribution of this work is a feasibility of the proposed solution. Thus, a creation of another new ontology is not the intention of this work. Instead, Semantic Sensor Network ontology is reused.

In this chapter, the approach, how to face the problem of semantic heterogeneity using a shared ontology, is introduced, i.e., how to create a data model of a CPS using shared ontology which contains an appropriate description for CPS components and thus enables their integration. It consists of following steps — the adjustment of the shared ontology (and extend as required), the description of a testing scenario, and the creation of a relevant knowledge base.

8.1 Cyber-Physical System Ontology for Components Integration (COCI)

As mentioned, another new ontology will not be developed, but rather proposed solution tries to reuse already existing accepted ontology for integration of cyber-physical system components. Our decision is motivated especially by improving re-usability of designed cyber-physical systems for other future extensions or applications. Unfortunately, there is no suitable existing ontology for describing cyber-physical systems from the component point of view. On the other hand, there is SSN ontology which is able to describe sensor-relevant concepts, i.e., the important part of a CPS. Thus, SSN ontology was chosen to constitute the cornerstone of our ontology.

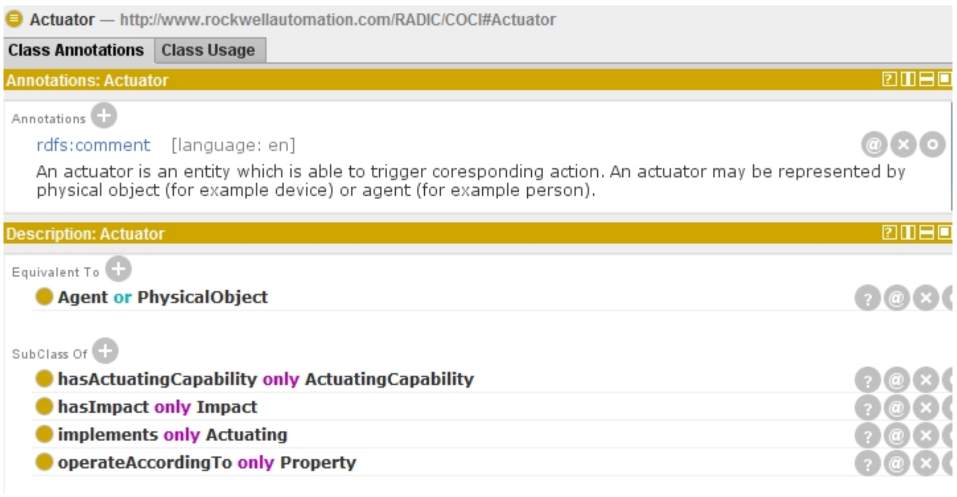
Furthermore, SSN ontology provides the suitable representation for the part of demanded entities. Moreover, it is built on DOLCE ontology, and this property may facilitate a potential future extension.

In following paragraphs, a design of concepts for a description of remaining devices and other relevant entities of a CPS was introduced.

8.1.1 Required Concepts for Cyber-Physical System Components

As already mentioned, SSN Ontology introduced very important concepts for a cyber-physical system — sensing devices, sensors, observations, physical qualities, their relationships, etc. However, it is not fully sufficient for our needs. Thus, required additional concepts has to be designed, i.e., mainly actuators together with their properties and capabilities.

The important part of cyber-physical systems is a feedback to a corresponding physical process. The feedback is provided by means of devices named actuators. An actuator triggers a corresponding action based on a computational process (an


Actuator — http://www.rockwellautomation.com/RADIC/COCI#Actuator

Class Annotations | Class Usage

Annotations: Actuator

Annotations

rdfs:comment [language: en]

An actuator is an entity which is able to trigger coresponding action. An actuator may be represented by physical object (for example device) or agent (for example person).

Description: Actuator

Equivalent To

Agent or PhysicalObject

SubClass Of

hasActuatingCapability only ActuatingCapability

hasImpact only Impact

implements only Actuating

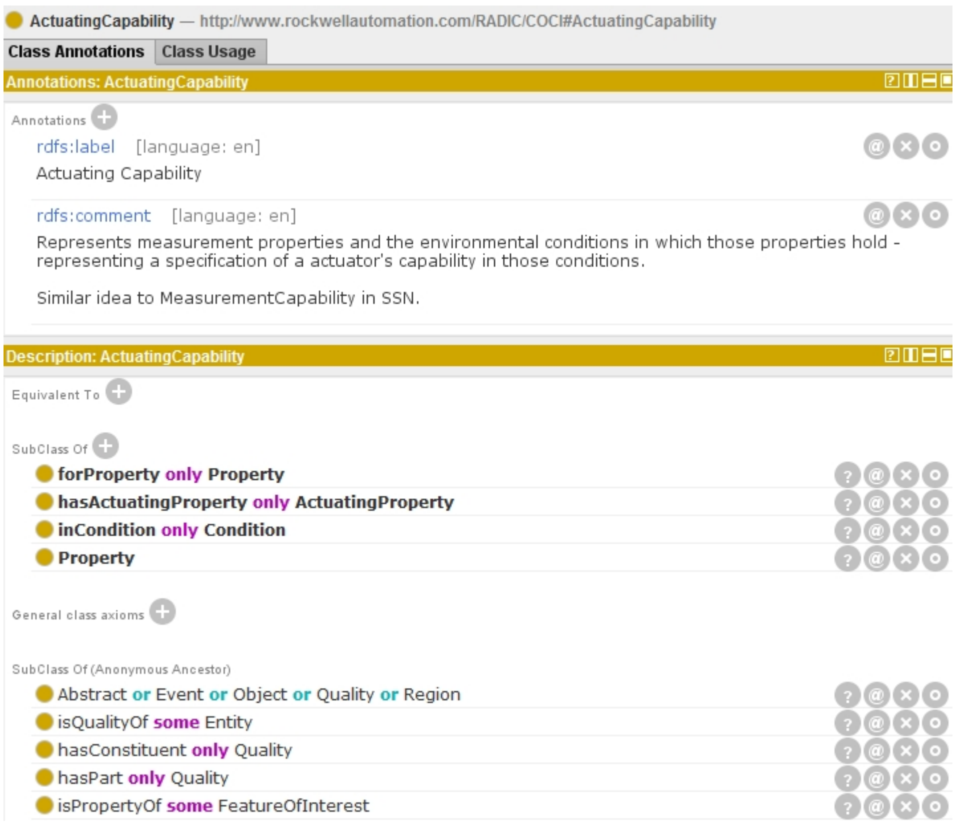
operateAccordingTo only Property

Figure 8.1: Actuator concept detail in Protégé editor.

algorithm). Two essential concepts may be derived from this fact — actuator and action. The most of the important concepts (what were designed) are follows:

- *Actuator* — This concept modeled in Protégé[1] ontology editor is depicted in the Fig. 8.1. Similarly to sensor concept, the *Actuator* concept is modeled as sub-class of *Object* (the concept from Dolce Ultra Light ontology). In general, an actuator is an entity which is able to trigger a given action. An actuator may be represented by actuating device (e.g., an electric motor) but we allow the situation when an actuator is a person (i.e., Cyber-Physical-Social System).
- *Actuating Device* — The *Actuator* concept has a hardware representation named *Actuating Device*. This concept is derived from *SSN:Device* concept which is derived from *DUL:DesignArtifact*.
- *Actuating Capability* — Every actuator has a different capability in specific conditions, and it is modeled by *Actuating Capability* concept. This concept detail is depicted in the Fig. 8.2.
- *Actuating Property* — Directly related to the *Actuating Capability* is *Actuating Property*. This concept represents characteristics of an actuators ability

[1]Protégé ontology editor — https://protege.stanford.edu



ActuatingCapability — http://www.rockwellautomation.com/RADIC/COCI#ActuatingCapability

Class Annotations | Class Usage

Annotations: ActuatingCapability

Annotations

rdfs:label [language: en]
Actuating Capability

rdfs:comment [language: en]
Represents measurement properties and the environmental conditions in which those properties hold - representing a specification of a actuator's capability in those conditions.

Similar idea to MeasurementCapability in SSN.

Description: ActuatingCapability

Equivalent To

SubClass Of
- forProperty only Property
- hasActuatingProperty only ActuatingProperty
- inCondition only Condition
- Property

General class axioms

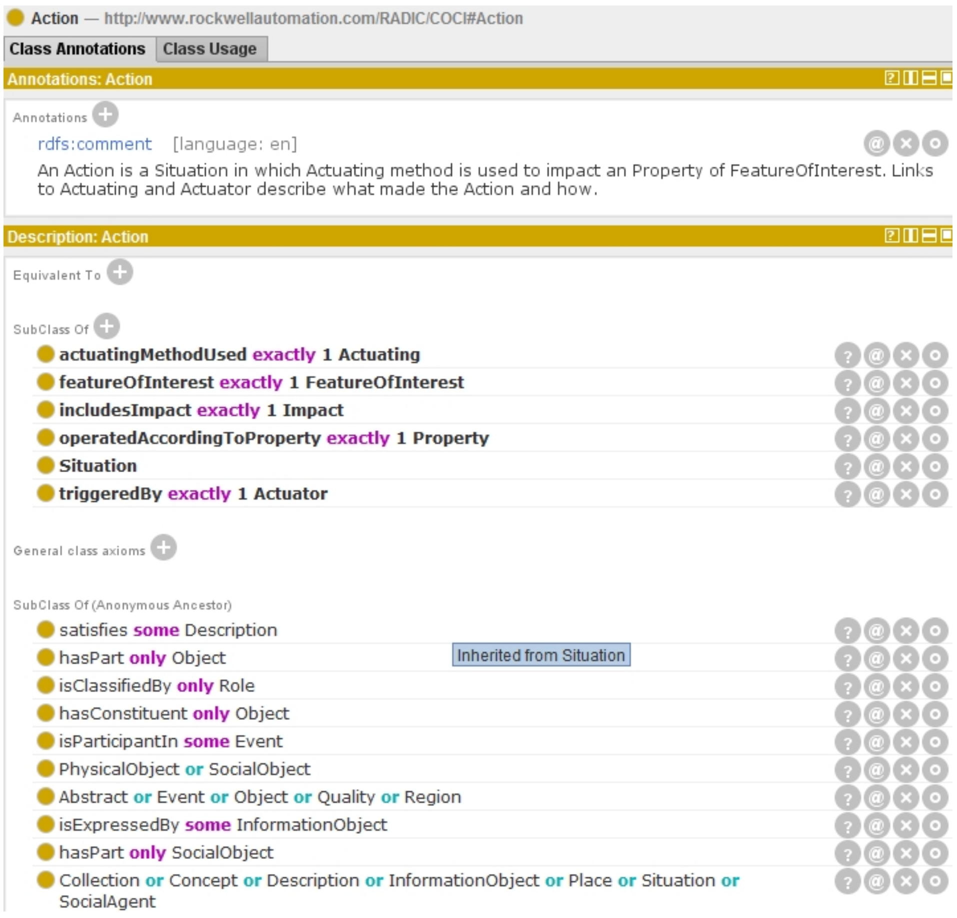
SubClass Of (Anonymous Ancestor)
- Abstract or Event or Object or Quality or Region
- isQualityOf some Entity
- hasConstituent only Quality
- hasPart only Quality
- isPropertyOf some FeatureOfInterest

Figure 8.2: Actuating Capability concept detail in Protégé editor.

to perform an action, e.g., stalling (for example pneumatic actuators may be stalled indefinitely without overheating in contrast to electric actuators), linearity (accuracy of linear motion), thermal stability, etc.

- *Impact/Actuator Output* — the *Impact* concept which is equivalent to the *Actuator Output* concept describes a way of physical process modification which is conducted by an actuator.

- *Actuating* — this concept stands for a process how an actuator influences a given physical process. It may be represented for example by a specific motion (e.g., linear) or by some display modalities.


Action — http://www.rockwellautomation.com/RADIC/COCI#Action

Class Annotations | Class Usage

Annotations: Action

Annotations

rdfs:comment [language: en]

An Action is a Situation in which Actuating method is used to impact an Property of FeatureOfInterest. Links to Actuating and Actuator describe what made the Action and how.

Description: Action

Equivalent To

SubClass Of

- actuatingMethodUsed **exactly** 1 Actuating
- featureOfInterest **exactly** 1 FeatureOfInterest
- includesImpact **exactly** 1 Impact
- operatedAccordingToProperty **exactly** 1 Property
- Situation
- triggeredBy **exactly** 1 Actuator

General class axioms

SubClass Of (Anonymous Ancestor)

- satisfies **some** Description
- hasPart **only** Object
- isClassifiedBy **only** Role
- hasConstituent **only** Object
- isParticipantIn **some** Event
- PhysicalObject **or** SocialObject
- Abstract **or** Event **or** Object **or** Quality **or** Region
- isExpressedBy **some** InformationObject
- hasPart **only** SocialObject
- Collection **or** Concept **or** Description **or** InformationObject **or** Place **or** Situation **or** SocialAgent

Figure 8.3: Action concept detail in Protégé editor.

- *Action* — The last important concept is an *Action*. An action describes a situation in which actuating method is used to impact a property of a feature of interest. The Action concept detail is depicted in the Fig. 8.3.

Next, the most important proposed and designed object properties are as follows:

- *hasActuatingCapability* — describes a relation between *Actuator* and *ActuatingCapability* concepts.
- *hasImpact* — describes a relation between *Actuator* and *Impact* or *Actua-*

torOutput concepts.

- *implements* — describes a relation between *Actuator* and *Actuating* concepts.
- *operateAccordingTo* — describes a relation between *Actuator* and *Property* concepts.
- *hasActuatingProperty* — describes a relation between *ActuatingCapability* and *ActuatingProperty* concepts.
- *inCondition* — describes a relation between *ActuatingCapability* and *Condition* concepts.
- *actuatingMethodUsed* — describes a relation between *Action* and *Actuating* concepts.
- *triggeredBy* — describes a relation between *Action* and *Actuator* concepts.

Before the completion of this dissertation thesis, the work about Semantic Actuator Network (SAN) was published in 2016 [99]. This work was designed as a complement to SSN ontology similarly as our extension. SAN ontology is not used in this work because it does not model all of relevant concepts and relationships. Furthermore, it is not considered as World Wide Web Consortium standard in contrast to SSN.

The important property of sensing and actuating devices is also their location. It is essential for device determination especially for example within a complex production line, etc. Thus, the property *DUL:hasLocation* was added to the *ActuatingDevice* concept. Then, a device location may be specified in some specific part of a production line or specific premises.

If we assume that a cyber-physical system is not separated from surroundings systems, then we may improve a feedback from computational process back to a physical process using external data sources involvement. Thus, the computational process may be more accurate or more capable.

In contrast to sensors and actuators, an external data source is very general term and therefore it is difficult to standardize a corresponding concept. In other words, a valuable external data source may represent very diverse information — from a weather forecast to relevant information from MES/ERP systems or information from accomplished maintenance service. A concept type is strongly dependent on a given application. The best solution is to infer concepts from base DOLCE concepts as recommendation guidelines.

The most important concepts together with their relations are shown in Fig. 8.4. The concepts with the blue edge are from DOLCE Ultralight Ontology and serve as general predecessors of all COCI concepts. There are also several SSN concepts (with the yellow edge) — general concepts from SSN ontology are reused such as *SSN:Property*, *SSN:Process*, and *SSN:FeatureOfInterest* instead of design similar concepts in COCI. Finally, there are shown the essential COCI concepts (with the green edge) representing entities related with an actuator.

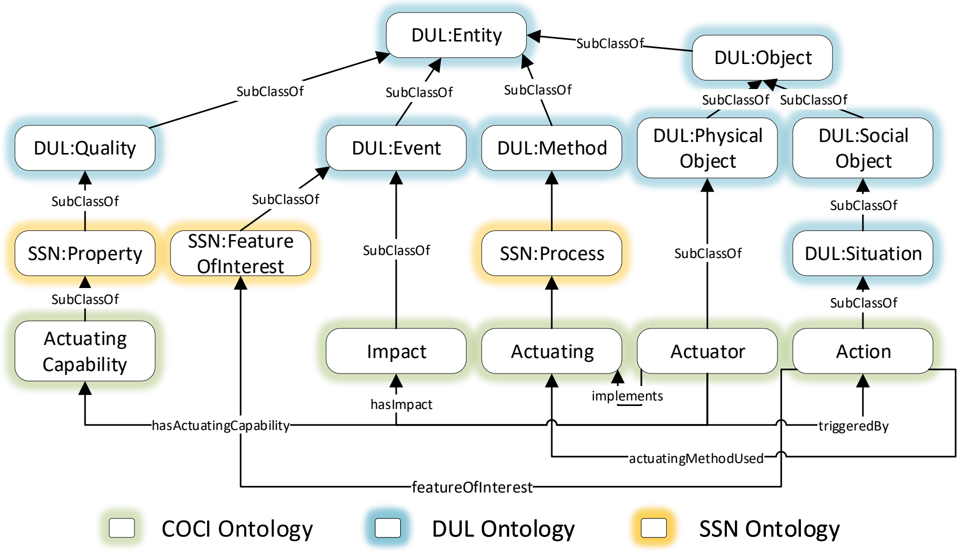


Figure 8.4: Part of cyber-physical system ontology for components integration.

8.2 Design of Knowledge Base: Experimental System

In following paragraphs, the experimental system, which is used for verification and an evaluation of our approach, is introduced. This experimental system also pervades subsequent chapters but from a different point of view. In this section, the system is utilized for testing the COCI conceptualization. In chapter 10, the experimental system is used for an integration of additional data sources and an evaluation of the hybrid SBDH model. Furthermore, Plug&Play concept is demonstrated on the same system in chapter 11.

Thanks to HydroCon company[2], we have access to data (online as well as historical) and a control system of the hydroelectric power plant located in Hluboka nad Vltavou (Czech Republic). As mentioned, this system is used in this section to verify COCI conceptualization. The power plant is equipped with 38 sensors in the power plant including for example measurement of fall of water, frequency, power factor, and real power. All data from power plant sensors are read with 5-second sampling rate.

Moreover, an experimental CPS was composed based on these data and with the help of Semantic Big Data Historian which is introduced in section 10. The experimental cyber-physical system was designed for resolving a hydroelectric power plant "stop problem". The stop-problem is defined as follows: turbine vanes are fouled up with filth during the turbine usage. This fouling causes a decrease in turbine performance. If the turbine is restarted, then a shock wave cleans turbine vanes. The problem is to identify the optimal moment for a restart.

The experimental CPS is composed of sensors (located in the power plant), Semantic Big Data Historian for collecting and processing sensor data, and an actuator which ensures the turbine restart. All experiments were conducted without a direct connection to the power plant ("offline mode") because of the reason of power plant production safety.

The solution of the "stop problem" is not in the scope of this work. Nevertheless, its solution is shortly described in this paragraph. It is based on the classification

[2]http://www.hydrocon.eu

of an abnormal state with the help of neural networks (multi-layered perceptron). The classification consists of two levels — the first level sets aside samples where the power of turbine should be higher; the second level determines a difference between actual and possible achievable power.

The knowledge base (an instantiation of COCI in other words) consists of two segments — an actuator part and a sensor part. Before the modeling of individuals, a specialization of general SSN and COCI concepts corresponding to the specific application has to be added, e.g., particular types of sensors, actuators, observations, etc. The essential concepts related to the actuator (servomotor which is responsible for positioning the control gate) are illustrated in Fig. 8.5. There are shown concepts such as Actuator, Actuating, ActuatorOutput, Action, and ActuatingCapability in the upper part of rectangles. Next, corresponding individuals are depicted in the lower part of rectangles for simplicity.

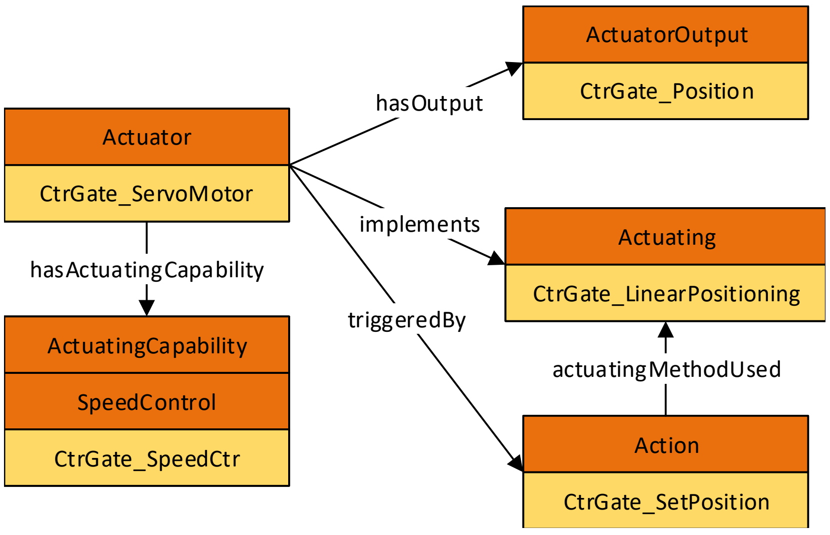


Figure 8.5: Simplified COCI representation of the control gate actuator.

The prevalent part of the experimental system's data model consists of sensor related concepts. For better readability, the simplified figure of the model is divided into two parts which are illustrated in Fig. 8.6 and Fig. 8.7. Besides other things, there are depicted four various features of interests (i.e., fall of water, real power,

power factor, and frequency), corresponding sensors, and observation values.

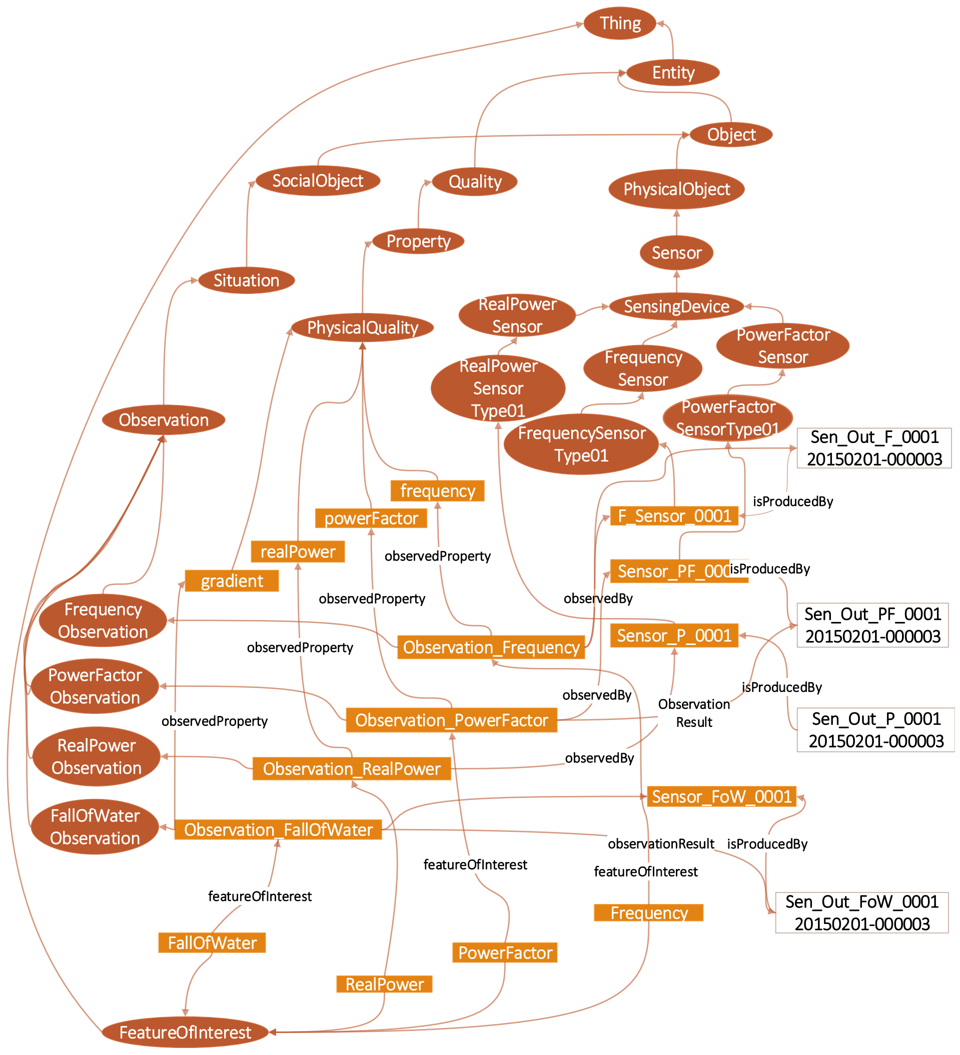


Figure 8.6: The first part of simplified COCI ontology.

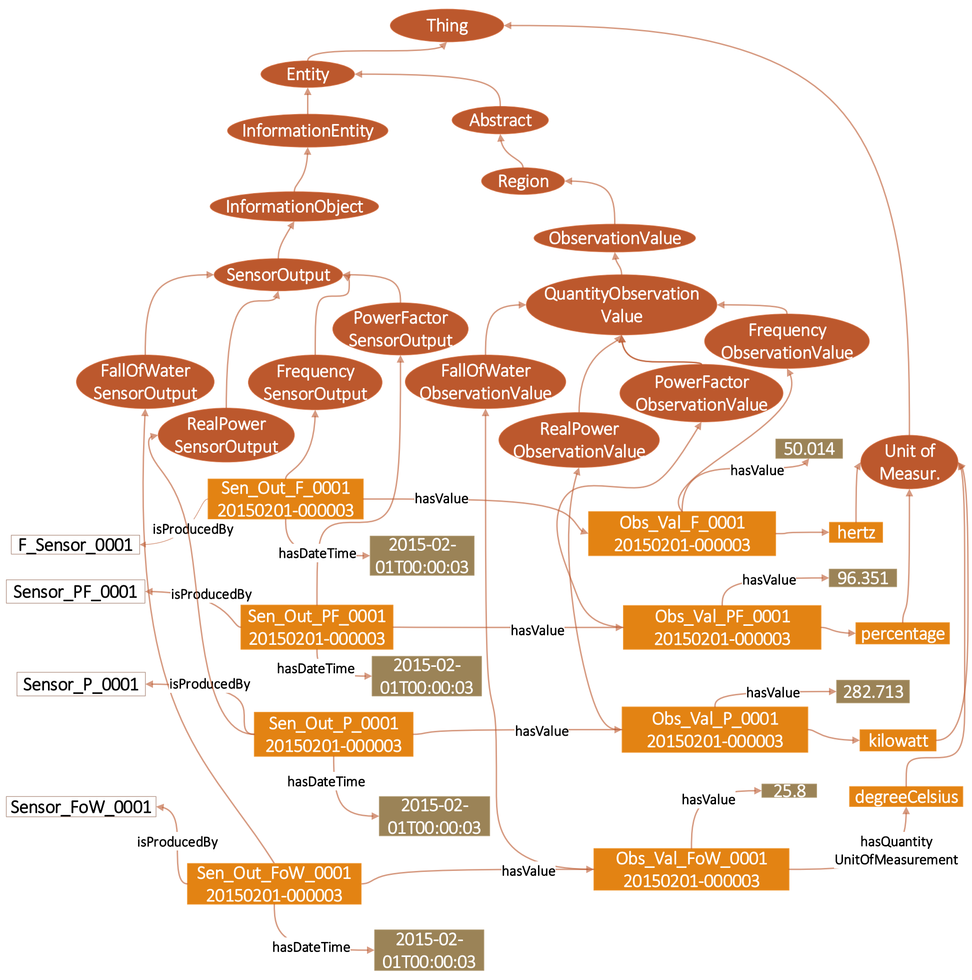

Thing
Entity
InformationEntity
Abstract
Region
InformationObject
ObservationValue
SensorOutput
QuantityObservation Value
PowerFactor SensorOutput
Frequency ObservationValue
FallOfWater SensorOutput
Frequency SensorOutput
FallOfWater ObservationValue
PowerFactor ObservationValue
RealPower SensorOutput
RealPower ObservationValue
50.014
Unit of Measur.
hasValue
Sen_Out_F_0001 20150201-000003
isProducedBy
F_Sensor_0001
hasDateTime
2015-02-01T00:00:03
Obs_Val_F_0001 20150201-000003
hertz
96.351
Sensor_PF_0001
Sen_Out_PF_0001 20150201-000003
Obs_Val_PF_0001 20150201-000003
percentage
Sensor_P_0001
Sen_Out_P_0001 20150201-000003
282.713
Obs_Val_P_0001 20150201-000003
kilowatt
25.8
degreeCelsius
Sensor_FoW_0001
Sen_Out_FoW_0001 20150201-000003
Obs_Val_FoW_0001 20150201-000003
hasQuantity UnitOfMeasurement

Figure 8.7: The second part of simplified COCI ontology.

8.3 Summary

In this chapter, the approach for resolving the semantic heterogeneity of CPS components by the employment of the shared ontology, which is intended to serve as a joint data model for a CPS, was introduced. The proposed Cyber-physical system Ontology for Component Integration (COCI) was described. The ontology employs SSN ontology for a description of sensors, their properties, and relevant processes.

Next, the missing part of CPS components description was designed — actuators, actions, etc. The new concepts serve as a complement of SSN ontology.

Finally, a utilization of COCI ontology on hydroelectric power plant devices was demonstrated.

To conclude this chapter, it is apt to remark that there is new release candidate of SSN ontology. The new version (release candidate) of SSN ontology contains concepts for actuators as described in section 5.5. However, the concept of SSN (respectively SOSA) ontology is heading towards a lightweight ontology inspired by Linked Open Data, and we have to evaluate benefits and deficiencies of such approach to be able to decide for a potential replacement of our COCI.

Chapter 9

Ontology Matching for Cyber-Physical Systems

A data models of sensors and actuators could be easily designed according to the specific application of COCI if the ontology is known during CPS concepts design time. However, the common practice is a usage of existing devices from various manufacturers due to many reasons, e.g., device price. Thus, a way how to facilitate the integration of these devices together with COCI ontology is needed.

Many researchers and developers have the effort to design and develop fully automatic matching systems. Naturally, automatic systems have deficiencies in precision and recall because of impossibility to find corresponding elements automatically. Furthermore, a user is more capable than a computer in finding correspondences. Next, many applications require the best precision and recall more than matching velocity, e.g., in manufacturing or medicine.

Thus, the semi-automatic matching system, which allows involving user as well as balance a trade-off between the matching velocity and the matching precision/recall, was proposed and implemented. The implemented framework is called MAPSOM and is introduced in this chapter. This solution is based on similarity measure aggregation by means of Self-Organizing Map. The process of ontology matching is composed of two steps: 1. step — automatic ontology matching approach; 2. step — optimizing outcomes from the first step by means of active learning.

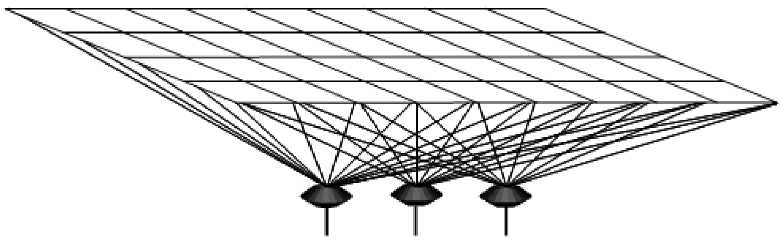


Figure 9.1: The SOM with rectangular topology.

9.1 Self-Organizing Map-Based Similarity Measures Aggregation

The cornerstone of the proposed matching solution is the self-organizing map (SOM). The self-organizing map (SOM) is the neural network introduced by Teuvo Kohonen [100]. The SOM implements a characteristic nonlinear projection from a high-dimensional space onto a low-dimensional array of neurons, and the mapping has the capability to preserve the topological relationships. Furthermore, the SOM has important applications in the visualization of high-dimensional systems and is possible to discover categories and abstractions from raw data.

The SOM usually consists of a two-dimensional regular grid of neurons (see Fig. 9.1). Each neuron in the SOM is a d-dimensional weight vector (codebook vector) where d is equal to the dimension of the input vectors. The neurons are connected to adjacent neurons by a neighborhood relation, which determines the topology of the map, i.e., hexagonal or rectangular topology. The mapping is ensured by the SOM algorithm in the following way: assuming a general distance measure between an input vector x and a codebook m_i denoted as $d(x, m_i)$, then the corresponding output neuron $c(x)$ (winner) is defined as

$$c(x) = arg\ min\ d(x, m_i). \tag{9.1}$$

The learning algorithm in the SOM is called competitive learning. The basic idea of the classical SOM competitive learning algorithm can be expressed as

$$m_i(t+1) = m_i(t) + h_{c(x),i)}(t)[x - m_i(t)], \tag{9.2}$$

where t is the index of the iteration (time), $m_i(t)$ is the weight vector of the corresponding neuron i in the time t, $h_{c(x),i}$ is called the neighborhood function, and x is the input vector.

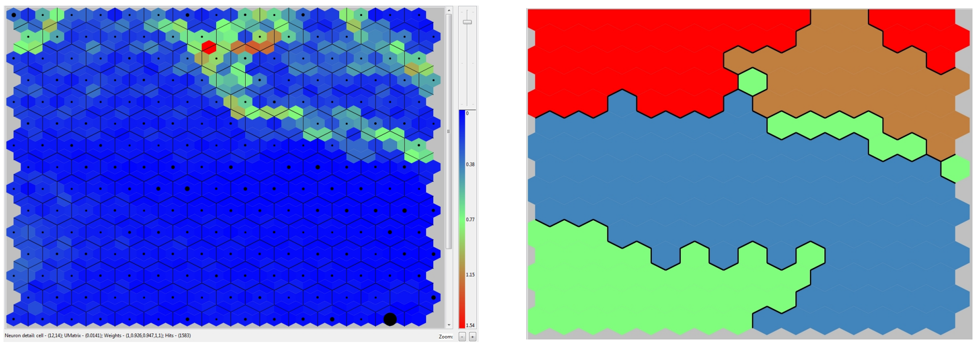
In the proposed solution, the SOM is exploited for similarity measure aggregation. Input vectors for SOM training are composed of different similarity measure values between pairs of concepts from a source and the target ontology (COCI ontology). The main benefit — pairs of concepts with the similar features (which describes used similarity measures) are located in a nearby area of the SOM output layer after training. Then, neurons from the output layer are clustered by ward clustering and classified into two classes — positive or negative. Now, a user can utilize information about neurons from clustering, visualization (see below), and initial classification to prove the classification or a user may tune the classification by means of active learning.

9.1.1 Visualization

A visualization is an important feature of a user interface for enabling easy and meaningful user interaction. Three main different visualization possibilities are implemented in MAPSOM framework — U-Matrix, Ward clustering, and Hit histogram. These visualization methods together offer effective mean during user decision making.

U-Matrix method [101] (see Fig. 9.2a — an example of U-Matrix and hit histogram) visualizes the distances among neurons in a SOM, and thus the U-Matrix is possible to show cluster structure of the SOM. High values indicate a cluster border and areas of low values indicate clusters themselves.

Ward Clustering is implemented for automatic cluster creation. It is possible to operate with the whole group of neurons instead of a single neuron. The cluster method of Ward belongs to the hierarchical agglomerative cluster algorithms (i.e., every single neuron is a cluster in itself, and the clusters with minimal distance are merged in every step). The distance characterizing Ward's method is based on the variance criterion. This distance measure is called the Ward distance and is defined



(a) U-Matrix and Hit histogram. (b) Ward clustering.

Figure 9.2: MAPSOM — Visualization.

as follows:

$$d_{rs} := \frac{n_r \cdot n_s}{n_r + n_s} \cdot \left\| \bar{x}_r - \bar{x}_s \right\|^2 , \tag{9.3}$$

where r and s denote two specific clusters — n_r and n_s denote the number of data points in the two cluster, and $\bar{x}_r$ and $\bar{x}_s$ denote the centers of gravity of the clusters. $\|.\|$ is the Euclidean norm. The number of clusters is variable, and a user can vary this number. Ward clustering of SOM with hexagonal topology and four clusters is depicted in Fig. 9.2b.

Hit Histogram can be combined with Ward clustering, U-Matrix, or with both of them. Hit histogram visualization method allows to find out, how many samples correspond to a neuron. U-Matrix and hit histogram combination of SOM with hexagonal topology and 15 neurons in every dimension is depicted in Fig. 9.2a.

9.1.2 Initial Classification

In the MAPSOM framework, an initial classification of clusters is needed. Thus, *a boolean conjunctive classifier B* and *a linear weighted classifier L* are implemented for this purpose.

Boolean conjunctive classifier B is defined as

$$F_B = \bigwedge_{i=1}^{n} (\sigma_i(s,t) \geq \tau_i), \tag{9.4}$$

where s and t are a certain pair of ontology entities, σ_i is a similarity function, and τ_i is a threshold of i^{th} similarity function.

Linear weighted classifier L has the form

$$F_L = \sum_{i=1}^{n} \omega_i \sigma_i(s,t), \tag{9.5}$$

where s and t are a certain pair of ontology entities, σ_i is a similarity function, and ω_i is a weight of i^{th} similarity function.

The cluster classification is not computed for every single neuron, but it is computed from the center of mass of the cluster. This fact causes the situation that the classification of neurons may vary depending on the number of clusters.

From a different point of view, somebody can wonder about advantages and suitability of the SOM utilization for ontology matching problem. Samuel Kaski describes in [101] that SOM is appropriate for data feature exploration. A usage of the SOM involves following advantages - the possibility of visualization of high-dimensional data, clustering, and non-linear projection capability. These characteristics are essential for data exploration together with user involvement and data advanced visualization of high dimensional data.

9.2 User Involvement in Ontology Matching

Primarily, the target of an ontology matching is to classify pairs from ontologies with the highest accuracy. This effort lies in the best setting of classifier parameters. Unfortunately, there are some limitations especially based on the heterogeneity problem which was described in section 5.1. User involvement in the process of semantic integration is one of the possible approaches to addressing this limitation and allows enhancement of this process.

The already described visualization of the SOM was chosen for user involvement in MAPSOM. Moreover, the active learning is employed for one more way how to involve a user in the process of ontology matching.

9.2.1 Active Learning

Active learning is a special part of semi-supervised machine learning and is frequently used in classification, filtering, information extraction, and speech recognition. Supervised learning algorithms have to be trained on hundred (or thousands) of labeled instances. An acquisition of labeled samples is very difficult, expensive, and time-consuming in many cases:

- Classification — A training task to classify documents requires that an annotator label each document with relevant classes, e.g., “positive” or “negative”.
- Information extraction — Information extraction system must be trained using samples (documents) with detailed labels (annotations). Providing annotations for samples is very time-consuming in these systems, e.g., locating entities and relations can take a half-hour or more for even simple stories [102].
- Speech processing and recognition — A trained linguists are needed for accurate labeling during speech processing. An annotation at the word level takes approximately ten times longer than the actual audio and annotating phonemes takes 400 times than the given audio [103].

Therefore, the main goal of active learning is to overcome a labeling of samples by asking an oracle (a human annotator in many cases).

For many applications, a large number of unlabeled data can be collected at once. This motivates pool-based sampling [104], where data are selectively drawn from the pool, which is usually non-changing. Unlabeled samples are drawn according to their informative contribution. In other words, active learning methods involve evaluating the informativeness of unlabeled samples, which are sampled from a given collection or newly generated.

9.2.2 Candidate Selection

There are many proposed and implemented methods of query strategies. Typically, the most commonly used query method is uncertainty sampling. This method selects

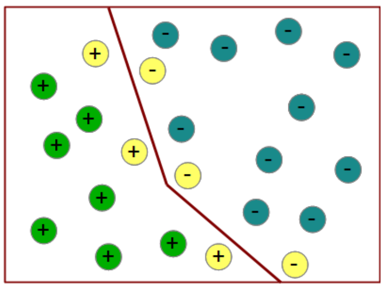


Figure 9.3: Candidate selection: The smallest distance from the boundary.

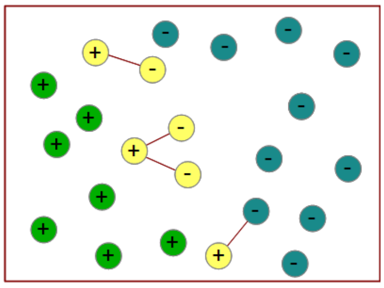


Figure 9.4: Candidate selection: The nearest samples from different classes.

samples with the most uncertain label assignment. For example, when using a probabilistic model for binary classification, uncertainty sampling directly queries the instances whose posterior probability of being positive is nearest 0.5 [105].

A certain number of candidates for a user approval is selected in the first step of the active learning process. The selection of these candidates has to fulfill a selection criterion to ensure an approval of the most uncertain classification. In the proposed system, if any additional domain knowledge is not considered for the selection, then it is easy to see that these candidates are the closest samples to the boundary between positive and non-positive alignment parts.

Two different information sources can be utilized for a distance description between samples with different classification in the selection criterion. The first source is classifier settings and the second source is composed of topological information from output SOM layer. Information from classifier allows finding precise boundary location. Hence, it is needed to find samples with the smallest distance from this boundary (Fig. 9.3). The main drawback of this approach could reside in the necessity of updating classifier settings after every step. In the second case, candidates for user approval can be described as the closest samples which have different classification (Fig. 9.4). The second approach has its drawback in higher computational complexity. The second approach is preferred in our solution because iterative parameters update is difficult or impossible for some specific types of classifiers. It is obvious that outcomes of these two selection algorithms may not be the same.

9.3 Experiments

In this section, the experimental results are analyzed to show the functionality of the proposed semi-automatic framework. The experiments were conducted using benchmark datasets with the reference matching provided by Ontology Alignment Evaluation Initiative (OAEI). The benchmark dataset[1] offers a set of tests which are wide in feature coverage, progressive and stable. It serves the purpose of evaluating the strength and weakness of matchers and measuring the progress of matchers. The conducted tests with MAPSOM framework are divided into two parts, and they are presented in the followings paragraphs. The first part shows how different initial parameters affect matching results and the second part presents MAPSOM functionality on the dataset with real ontologies (dataset No. 302).

The first part of tests shows variability and dependencies between SOM parameters and results. The most important parameters are a number of neurons of the SOM, types of similarity measures, and a type of the initial classifier. The used parameters of the SOM and learning algorithm were: 15 neurons in both dimensions, hexagonal topology, sigma - 0.5, neighborhood size - 8, learning rate - 0.4, and 1500 iterations. Following similarity measures were exploited: n-gram measure, Levenstein measure, Needleman-Wunsch measure [106], and Lin and Wu&Palmer for WordNet [64]. In this part, different initial classifiers were tried, i.e., the boolean conjunctive classifier (described in Eq. 9.4) with the threshold equal to 0.2 for every similarity measure and the linear weighted classifier (described in Eq. 9.5) with the weight equal to 0.2 for every similarity measure and threshold equal to 0.4. There are many suitable datasets in the OAEI Challenge for demonstrating MAPSOM functionality. The dataset No. 222 was chosen from the OAEI benchmark (MAPSOM outcomes are similar with the other datasets and this dataset is sufficient for our demonstration). Table 9.1. shows experimental results for previously mentioned parameters and different number of clusters. It is possible to achieve the best precision and recall with the different initial setting, and the main difference is the number of active learning iterations.

The second part of experiments demonstrates MAPSOM functionality with real ontology and comparison with the best systems of the matching challenge. Dataset

[1] http://oaei.ontologymatching.org/2013/benchmarks/

Table 9.1: Dependencies between different SOM configuration and results (dataset No. 222).

	Bool. Conj. Classifier		Bool. Conj. Classifier		Linear Weight. Classifier	
	4 Clusters		20 Clusters		20 Clusters	
Iteration	Precision	Recall	Precision	Recall	Precision	Recall
1	0.38	1	0.49	1	1	0.74
2	0.39	1	0.51	1	1	0.81
3	0.4	1	0.54	1	1	1
4	0.41	1	0.63	1		
5	0.48	1	0.7	1		
6	0.61	1	0.7	1		
7	0.68	1	0.73	1		
8	0.84	1	1	1		
9	0.84	1				
10	0.97	1				
11	0.99	1				
12	0.99	1				
13	1	1				

No. 302 is composed of finding alignments between reference ontology and real BibTex/UMBC ontology. The result is verified by reference matching from OAEI. We used SOM with 20 neurons in both dimensions, hexagonal topology, and following parameters of training algorithm: sigma - 0.5, neighborhood size - 10, learning rate - 0.5, and 1500 iterations. The similarity measures are n-gram measure, Levenstein measure, Needleman-Wunsch measure, and Lin and Wu&Palmer for WordNet. The initial classifier is boolean conjunctive classifier with thresholds equal to 0.2 for every similarity measure. The number of clusters before the active learning step is 15. The Table 9.2. shows the evolution of the precision and recall depending on the iterations.

Finally, the results of our experiment were compared with some of the best systems also tested on the dataset No. 302 (see Table 9.3.).

Table 9.2: MAPSOM and a real ontology (dataset No. 302).

Iteration	1	2	3	4	5	6	7	8	9
Precision	0.72	0.88	0.96	0.96	1	1	1	0.83	0.83
Recall	0.46	0.46	0.46	0.46	0.46	0.46	0.46	0.5	0.5
Iteration	10	11	12	13	14				
Precision	0.83	0.83	0.84	0.81	0.85				
Recall	0.5	0.5	0.52	0.54	0.61				

Table 9.3: MAPSOM and comparison with some of the best systems tested on dataset No. 302.

ASMOV		Lily		n-Harmony	
Precision	Recall	Precision	Recall	Precision	Recall
0.71	0.56	0.84	0.65	0.93	0.55
MAPSOM					
Precision	Recall				
0.85	0.61				

9.4 Summary

In this section, MAPSOM was described — a system for semi-automatic ontology matching. A user involvement is providing by visualization of an output layer of SOM as well as by an active learning. Thus, a user may achieve the excellent outcomes in reasonable time.

Furthermore, it is possible to use this system for integration of various formats, not only ontologies. For example, this system is employed for integration of data stored in excel sheet [67]. MAPSOM is suitable for an integration of data models in the industrial automation domain including CPSs because of the capabilities of the system.

Chapter 10

Semantic Big Data Historian

The Semantic Big Data Historian (SBDH) was proposed and developed to manage the semantic integration of CPS components. SBDH is an extension of common historian software. In general, a historian software is used in the industrial automation to gather data and then to provide access to the data and possibly also analytics of them. The historian software is usually optimized to allow fast and compressed storage of data, but not much attention is paid to analytics or heterogeneous data integration. Moreover, a capacity of a historian storage has in common limited size. Thus, the original architecture of historians becomes insufficient with advances in industrial informatics, and thus new technologies are applied, e.g., Big Data technologies or a cloud.

As already mentioned, an ontology is a suitable data model representation for sharing knowledge among all CPS parts. An ontology provides an unambiguous and clear understanding of modeled concepts and their properties. Unfortunately, the performance deficiencies are caused mainly by poor efficiency of triple stores and non-effective data representation which on the other hand ensures clear knowledge interpretation.

Big Data approach was employed to preserve unambiguous and clear knowledge representation and overcome the performance problems of processing large-scale data represented in ontologies. Data managed by a CPS are usually of big volume and are produced with high speed. Thus, the data velocity and volume refer to the second type of big data (according to big data taxonomy introduced by NIST — described in chapter 6) where horizontal scalability is required for effective analysis, and there could be a limitation for a relational database according to NIST review. Furthermore, there is also the last criterion (variety) of data as well. In many cases, a CPS has to process data produced by sensors from various manufacturers as well as a CPS has to control various actuators. Moreover, the variety increases when a CPS has to integrate data from other sources, e.g., MES/ERP systems or external sources such as weather forecast.

In this chapter, the mentioned SBDH, which benefits from the employment of Big Data and Semantic Web Technologies, is introduced. This solution was proposed to overcome common deficiencies of available solutions in case of handling (i.e., storing, processing and querying) of RDF triples.

In following paragraphs, the architecture of SBDH is described with the focus on the storage layer whose architecture and implementation is essential for a proper operation of SBDH. Next, two versions of SBDH storage layer are introduced — the first one based on Apache Hadoop and Jena Elephas and the second one which employs Apache Spark together with Apache Cassandra. After the presentation of the storage layer representations, an integration of the experimental CPS model with external data sources using COCI ontology is demonstrated. Finally, the proposed data model for storage RDF triples produced by SBDH is introduced followed by several experiments.

10.1 Semantic Big Data Historian Architecture

Semantic Big Data Historian was proposed to facilitate data acquisition, processing, integration, storage, and analyzing. Therefore, it should provide a flexible environment for enabling implementation of various data connectors to sensors/actuators or other sources (e.g., OPC DA, OPC UA, PROFINET, JDBS, etc.). Next, SBDH has to provide high-performance data processing and storage together with means for subsequent data analysis.

The architecture of Semantic Big Data Historian (SBDH) is introduced in the following paragraphs, and the architecture is illustrated in the Fig. 10.1.

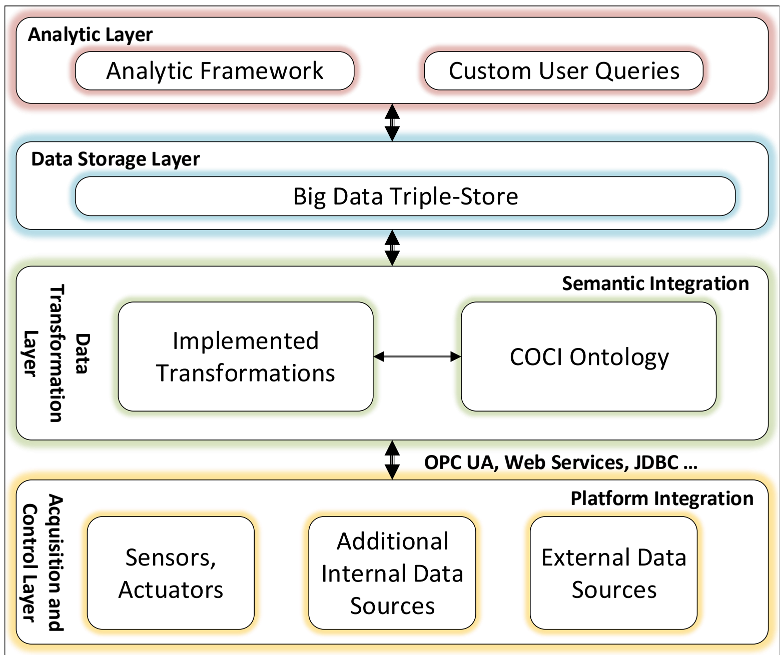


Figure 10.1: Architecture overview of Semantic Big Data Historian.

The historian architecture is divided into four main layers — data acquisition and control layer, transformation layer, data storage layer, and analytic layer.

- *Data acquisition and control layer* — collects data from sensors (or actuators), other systems related to a given application (for example from MES/ERP systems — information about shifts, supply chain, etc.), and relevant external data sources (e.g., weather forecast, traffic information). Various data sources are gathered and connected mainly via OPC UA [107]. Furthermore, this layer provides a possibility to control actuators. The platform heterogeneity (various developers and manufacturers) has to be resolved by this layer.

- *Transformation Layer* — transforms data to the unified semantic form according to COCI ontology. This layer is responsible for data pre-processing (corrections of damaged data, etc.) if needed. Created triples are subsequently stored in the corresponding storage system. The semantic heterogeneity is solved by this layer.

- *Data storage layer* — several triple stores were evaluated during SBDH development. The most promising solutions were 4Store[1], CumulusRDF[2], and Hadoop[3] together with Jena Elephas[4]. Every mentioned solution has a certain limitation (performance issues, limitations caused by design) and thus the Data storage layer was implemented by means of Apache Spark[5] and Apache Cassandra[6]. The data storage layer is described in the section 10.1.1 in detail.
- *Analytic layer* — this layer provides access to directly connected storage layer for custom analytic programs or custom user queries. In the current implementation, SBDH provides the possibility to perform analysis with the help of Apache Spark and MLlib[7].

10.1.1 Data Storage Layer

Several prerequisites were identified for enabling proper utilization of RDF data form in industrial automation domain especially concerning Industry 4.0. The most important prerequisites are as follows:

- A suitable and correct ontology for representing required knowledge. Apparently, this is the cornerstone of the overall proposed solution, and this approach has more drawbacks than benefits without properly captured knowledge in the ontology.
- A modular and scalable way how to interconnect system parts. There are several suitable ways for an interconnection of all parts of a distributed system. A suitable way is strongly dependent on a given application domain. Nowadays, some promising and versatile standards are coming to the fore — for example, OPC UA [108]. This standard offers versatile approach how to design information model as well as a way how to communicate across various operating systems and from a shop floor to highest enterprise levels (e.g., ERP[8]).

[1]http://4store.org
[2]https://code.google.com/p/cumulusrdf/
[3]http://hadoop.apache.org
[4]https://jena.apache.org
[5]http://spark.apache.org
[6]http://cassandra.apache.org
[7]https://spark.apache.org/mllib/
[8]ERP — Enterprise resource planning system

- Realization of the data storage layer — the right selection of the technology which will be used together with proper data model definition according to a given application should ensure efficient and faultless system operation. In this section, the possible solution is discussed and presented in detail.
- Analytical and querying tools. An appropriately operating system is valueless without any reasonable tool which is able to access information stored in the system. It means the way how to query data by a user or another system via a proper API as well as the basic/advanced tools for conducting analytical tasks.

This section is focused on the realization of the data storage layer of the SBDH. It is the essential component (together with a corresponding ontology) of the system which is intended for handling and storing RDF data. The realization of the data storage layer influences an efficiency of data management and processing itself as well as a scalability of possible applications.

Many different systems for the storage layer implementation were tested during years of the SBDH development. Solutions based on available triplestores have several drawbacks, but the main insufficiency is their performance. Thus, the proposed prototype was built upon some Big Data framework.

The First Version of SBDH Storage Layer.

The first version of SBDH prototype has the storage layer based on Apache Hadoop together with Jena Elephas. This solution has many advantages, for example, it offers the very robust environment for massive distributed parallel data processing and very efficient cluster management with the help of YARN[9] — providing the computational resources (e.g., CPUs, memory, etc.) needed for application executions. Furthermore, there are available many additional tools for extending data processing and conducting various analytical tasks — Hive[10], HBase[11], Mahout[12], KNIME[13] connected by means of Hive connector, etc.

[9]YARN — Yet Another Resource Negotiator
[10]https://hive.apache.org
[11]https://hbase.apache.org
[12]http://mahout.apache.org
[13]https://www.knime.org

The Hadoop Distributed File System (HDFS) is designed to work with sequence files [109]. The SequenceFile is a flat file consisting of binary key/value pairs and is used in MapReduce [110] as input/output formats. This input/output format has many benefits — more compact than text files, offers support for data compression (particular records or whole blocks of records), are designed for parallel processing, etc. Unfortunately, SequenceFiles have one main disadvantage — they are append-only [111]. The "append-only" mode helps maintain easy data consistency. On the other hand, it is not sufficient for our realization of the SBDH storage layer. The prototype encountered fundamental problems during integrating (storing and processing) various data compared to simple storing of sensor data. Thus, the architecture of the SBDH storage layer has been changed and re-implemented with the help of Apache Spark together with Apache Cassandra.

Data Storage Layer Based on Apache Spark and Apache Cassandra.

The more suitable solution for the data storage layer for SBDH seems to be the combination of Apache Spark together with Apache Cassandra.

Apache Spark is a fast and general-purpose computing system which provides high-level APIs in Java, Scala, Python, and R. It also provides a set of additional tools including SparkSQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming. Spark may be deployed in three different modes depending on the used cluster manager — Standalone, Apache Mesos, and Hadoop YARN. The standalone deployment mode uses a simple cluster manager included in Spark which is sufficient for clusters that are not big. On the other hand, Mesos and YARN cluster managers should be utilized for huge clusters for improving the cluster performance.

Apache Cassandra is a NoSQL database project which originated at Facebook and is maintained by Apache Software Foundation. It is built on Amazon DynamoDB[14] and Google Big Table[15]. Cassandra was designed as a distributed database for managing large amounts of structured data across many commodity servers and for offering high availability. In comparison to the common NoSQL databases, the Cassandra uses a hybrid model between key-value and column-oriented data-

[14]https://aws.amazon.com/dynamodb/
[15]https://cloud.google.com/bigtable/

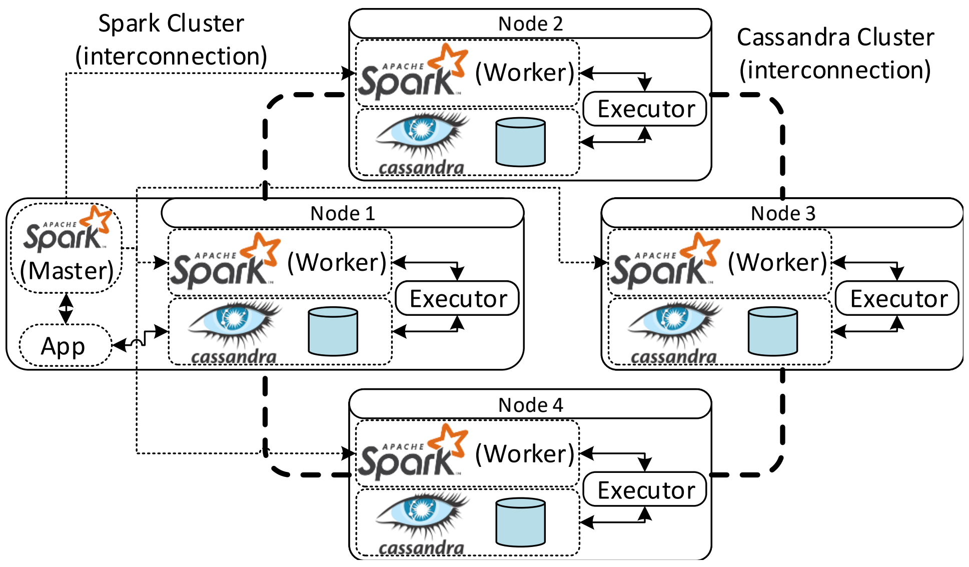


Figure 10.2: Example of Data Storage Layer architecture combining Apache Spark and Apache Cassandra.

base — based on defining super-columns and column-families. The topology of a Cassandra cluster is "masterless ring" due to overcoming a legacy master-slave architectures. The advantages of the Cassandra database could be summarized as follows — continuous availability, linear scale performance, operational simplicity and easy data distribution across multiple data centers.

In the proposed data storage layer, Spark and Cassandra clusters are deployed to the same set of machines. Cassandra serves as data storage, and Spark worker nodes are co-located with Cassandra and perform the data processing tasks. When a job is created, the Spark workers load data into memory and perform the required data processing. Very important fact of such processing is that there is no overhead with superfluous network traffic. Finally, the results are written back to the Cassandra tables or propagated to other systems. The architecture of the SBDH data storage layer is illustrated in the Fig. 10.2.

Data are stored in Cassandra tables by means of the Hybrid SBDH Model (see section 10.3). In detail, data corresponding to general entities are vertically partitioned by a predicate, e.g., the table named hasQuantityUnitOfMeasurement contains corresponding subjects and objects — :CO2ds048 :parts-per-million, etc. Next, time-related data are stored according to the hybrid SBDH model, e.g., the table with the composite name (subject#object) CO2ds048#hasQuantityValue contains objects representing measurements together with their timestamp — 2012-04-29T00:00:10 355.0, etc. The emerging problem is how to automatically recognize the right model for given entities in this approach. This problem is handled as follows — if the concept is connected with some object with the type timestamp, then the data are stored in hybrid SBDH model.

10.2 Validation Study

The semantic integration by the employment of shared ontology COCI (described in section 8) is discussed together with the Semantic Big Data Historian in this section. The already mentioned experimental system is utilized (the hydroelectric power plant) for the demonstration. As mentioned, data for processing include data measured by 38 sensors in the power plant including for example measurement of fall of water, frequency, power factor, and real power. All data from power plant sensors are read with 5-second sampling rate. These data sources are connected via implemented adapters to comply with the COCI ontology.

The significant problem is the performance issue in the context of sensors data processing — especially in the case when data are stored as RDF triples. Our sensors produce 656,640 samples per day. If these data are transformed into triples, then the volume of data is equal to 5,253,120 triples per day and it corresponds to 1,917 mil. triples per year.

Moreover, it is needed to involve additional data sources for the improvement of analytic results. The data integration involves available online information covering meteorological data (temperature and precipitation) and hydrological data (rate of flow, water level, and water temperature) for relevant locations. These data consist of measured samples as well as the prediction. The sampling rate is 1 hour for meteorological data and 10 minutes for hydrological data. These external data add negligible volume in comparison to data from the power plant, but they bring important information. Additionally, a plan is to extend the data by the purchase prices of electricity, by who is present at power plant premises, etc. All of these data may be utilized for an improvement of prediction accuracy in the case of the experimental CPS and the stop-problem.

The Fig. 10.3 represents partially the COCI ontology model of CPS components integration used by the historian. There is the concept model (including concepts from DOLCE ontology[16]) in the upper part and individuals representing one sensor and its measurement for every data source in the lower part of the figure. During SBDH operation, all of the incoming samples from all data sources (power plant sensors and external sources) are transformed into RDF triples, e.g., *Sensor_RP_0001 hasLocation Generator*. Next, the triples are stored in the corresponding triple store. When data are transformed and stored, then they may be utilized for subsequent analysis or direct user queries.

The advantage of this way of the integration represented by the COCI ontology and the SBDH are as follows — the ontology is able to describe (with the help of axioms) the reality in its representation. On the contrary, classical schemas (as a database schema) are representation mechanisms that are designed to meet the requirements of a particular application and when the requirements change it is difficult to change the schema and the implementation as well. Next, data can be easily queried in SPARQL. Relationships within data are explicitly described and directly accessible, and therefore SPARQL queries are significantly closer to a user understanding of the problem. For example, the sample query for listing all sensors located in the generator is described in Query 11.1.

Listing 10.1: Listing all sensors in generator.

```
PREFIX : <http://www.loa-cnr.it/ontologies/DUL.owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX coci: <http://www.rockwellautomation.com/RADIC/COCI#>
PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
SELECT ?sensor
        WHERE {?sensor :hasLocation coci:Generator.
        ?sensor a ?sensorType.
        ?sensorType  rdfs:subClassOf+ coci:Sensor}
```

[16]http://www.loa.istc.cnr.it/old/DOLCE.html

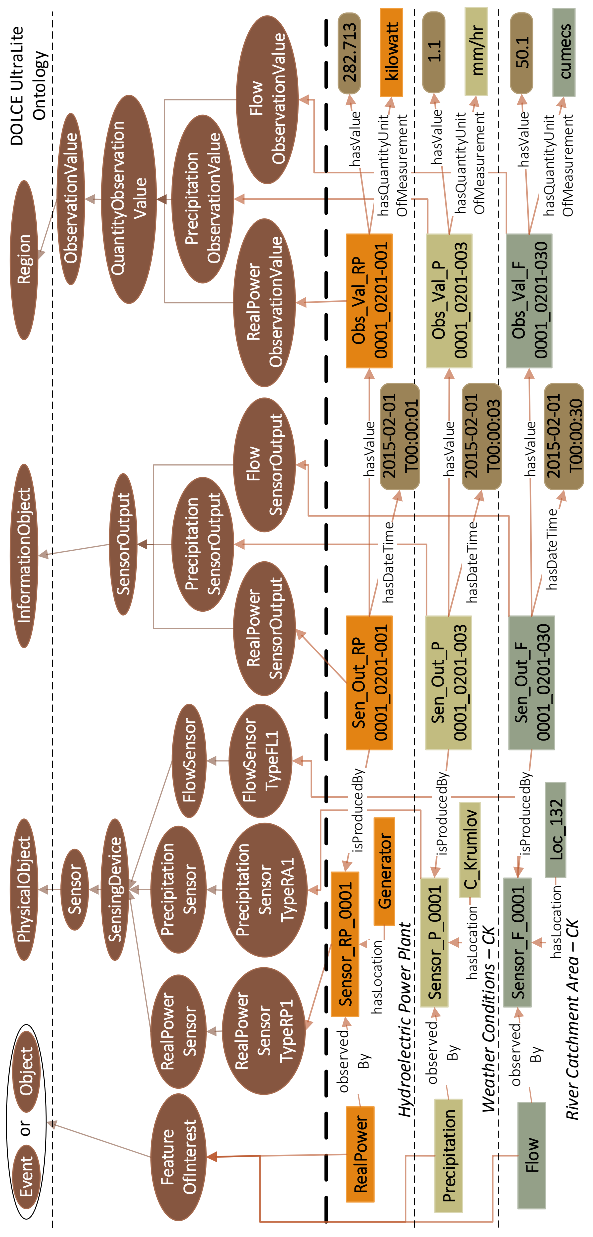


Figure 10.3: CPSs integration by means of COCI ontology.

As mentioned, the exploitation of ontology offers the possibility to check data consistency and reasoning. Furthermore, the expressivity can be significantly increased by utilization of the Semantic Web Rule Language[17]. The equation 10.1 illustrates a sample SWRL rule for the malfunction detection caused by unacceptable rotations of the generator.

$$\begin{aligned} &Generator(?g) \ \wedge \ hasRotation(?g, ?rot) \\ &\qquad \wedge \ swrlb : greaterThan(?rot, 1500) \ \rightarrow \ hasFailure(?g) \end{aligned} \tag{10.1}$$

10.3 RDF Storage for Semantic Big Data Historian

As already mentioned, a utilization of triplestores was chosen to store, query and retrieve the data, such as time series of measured sensor data. For processing of huge amounts of data, including their analysis, a scalable approach is needed. Let us discuss the solutions for our Semantic Big Data Historian.

There are many various already existing triplestores which offer mainly "database" for RDF triples (based on different technologies) and subsequently different support of data querying, inferring, etc.

The important triplestores characteristics include triplestore performance (required time for query processing) and the capability to store as many triples as possible. Widespread and well-known RDF triplestores [112, 113, 114, 115] are based on a centralized approach. These solutions have become unsatisfactory in the current trend of increasing data production due to their limited scalability. In addition to those centralized approach solutions, there are several distributed solutions (Apache Accumulo [116] or HadoopRDF [117]).

The available triplestores optimize the triple storage for general data models. On the other hand, when a deployment within industrial automation domain is considered then it is obvious that the prevalent amount of data are sensor measurements. Thus, the possible solutions of how to optimize RDF triple storage model for the time series data are introduced in the following paragraphs.

[17]https://www.w3.org/Submission/SWRL/

10.3.1 Data Models for Triple Store

Three different approaches were identified for storing RDF data in a distributed way according to the method of data model handling:

- **Single file/table model** preserves the triple construct of standard RDF.
- **Vertical partitioning model** splits RDF triples according to their properties.
- **Entity class-based model** utilizes high-level entity class graph to create RDF partitions [118]. First, similar entities (subjects) are grouped (according to similarity measure) into an entity class. Corresponding entity class graph is then partitioned. This model is not discussed in detail in the following sections because it is not used in our Semantic Big Data Historian.

Single Table Model

The single table model preserves the RDF triples in the form (subject, predicate, object). In other words, data are stored within a database system in one file/table. The database system is then responsible for splitting the tables into blocks, replicating the blocks, etc.

The system based on this approach and HDFS (Hadoop Distributed File System) is for example PigSPARQL [118]. Furthermore, SHARD [119] uses a variation of the single file model where triples with the same subject are merged into one line of a file/table.

```
:CO2ds048 rdf:type :CO2ObsValue :hasQuantityValue   3 5 5 .0
        :hasQuantityUnitOfMeasurement :parts-per-million
```

Vertical Partitioning Model

The previously described single table model is easy to implement but has some disadvantages. The main obstacle is the I/O cost during query processing. A more suitable model is represented by vertical partitioning model. In this model, triples are partitioned concerning their property and stored in files/tables named according to the corresponding property name. The vertical partitioning model is employed for example in [120]. In the case of SBDH, the table `hasQuantityUnitOfMeasurement` contains the following data:

```
:CO2ds048 :parts-per-million
:THSds075 :percentage
:THSds075 :degreeCelsius
:PRSds032 :hectopascal
```

This model overcomes deficiencies of the single table model but data are not homogenously distributed in tables in some cases (e.g., the type table is usually a very big table). In the case of SBDH, the biggest table would be `hasQuantityValue` as this relation is the most used one.

Further table splitting can be performed for ensuring homogeneous data distribution among files. HadoopRDF creates partitions according to data property and object as well. For example, the triple (`:THSds075 :hasQuantityUnitOfMeasurement :percentage`) would be stored in a file named `hasQuantityUnitOfMeasurement#percentage`.

Hybrid SBDH Model

Our current realization of the SBDH storage architecture is based on combining single table model and vertical partitioning-like model. This hybrid model replaced previously used single table model which had insufficient performance due to the high I/O costs during query processing. The single table model is unsuitable for time-series data storage. Especially for queries with range filter expressions and order constraints the acceptance was not acceptable.

The vertical partitioning is used for all sensors measurements where the partitions are created with respect to subject and property accompanied by a timestamp. For example, the table `CO2ds048#hasQuantityValue` contains the following data:

```
2012-04-29T00:00:10 355.0
2012-04-29T00:00:40 355.1
2012-04-29T00:01:10 355.0
```

Other triples are stored according to the vertical partitioning model. The different data handling of sensors measurements reflects the fact that an amount of measurements is significantly bigger than the rest of data.

10.3.2 Comparison

Several tests were conducted for demonstrating the suitability of the proposed hybrid SBDH model on data from the hydroelectric power plant. The hardware for these tests was the computer with two hard disks (SSD + magnetic HDD), 32 gigabytes of memory and CPU was Intel Core i7-7700T.

The cluster of Spark and Cassandra nodes was deployed using Docker containers. First, tests with two Spark workers and two Cassandra nodes were conducted. The importance of two separated hard disk resides in ensuring independent data storage for each cluster node. Shared data storage affects the speed of reading/writing operations. The objective of this test was the performance comparison of single table model, vertical partitioning model, and hybrid SBDH model. The test was focused on writing and reading sensor measurement sequence.

	Single Table	**Vertical Partitioning**	**Hybrid SBDH**
Write 1000 sensor samples	36.522s	29.612s	13.027s
Write 10000 sensor samples	94.428s	70.298s	34.412s
Write 100000 sensor samples	299.647s	296.767s	149.293s

Table 10.1: Comparison of different data models - writing data.

	Single Table	**Vertical Partitioning**	**Hybrid SBDH**
Read 3000 sensor samples	60.022s	20.012s	9.716s
Read 10000 sensor samples	113.428s	22.298s	10.231s

Table 10.2: Comparison of different data models - reading data.

The performance comparison of the different data models during writing sensor samples is presented in Table 10.1. Next, the performance comparison during reading sensor samples is presented in Table 10.2.

The measured times of experiments are mainly influenced by a different number of tables which are required to be accessed during the reading/writing operations. Different tables access according to the particular data model are summarized in the following list:

- **Single table model** — it is needed to access only one table, but filtering is very demanding operation. Whole records have to be parsed during filtering.
- **Vertical partitioning model** — two tables have to be accessed.
- **Hybrid SBDH model** — only one table has to be accessed.

Furthermore, it is important to be aware that the single table and vertical partitioning models store data from more sensors into one table. Therefore, next experiments were conducted to find out how the number of sensors stored in a table affects reading performance.

# of sensors	**Vertical Partitioning.**
1 sensor	22.298s
2 sensors	24.321s
3 sensors	28.982s
4 sensors	30.842s

Table 10.3: Vertical partitioning various number of sensors stored in one table — reading of 10,000 sensor samples

One of the experiments demonstrating dependency between the number of sensors and the reading time is described in Table 10.3. It shows that the reading time is not increasing together with the sensor number in a linear way according to our experiments. This verification is important for SBDH because this model is utilized for storage of non-time related data in the proposed historian.

10.4 Summary

In this section, proposed Semantic Big Data Historian, which is able to store and process a huge amount of RDF data, is introduced. First, the overview of SBDH architecture is provided together with the description of the storage layer in detail.

Next, the integration of a CPS sensor and external data sources is demonstrated. Then, several data models for RDF storage including the hybrid SBDH data model, which is used for storage of sensors measurements, were described.

Furthermore, experiments (the integration of various data sources and the evaluation of data models characteristics) proved the feasibility of the semantic integration of CPS components by employment a shared ontology.

Chapter 11

Plug&Play Components of Cyber-Physical Systems

For many years, manufacturers desire to have an automatic system where an employee connects a new part/device without any additional configuration. Nevertheless, no contemporary system offers such feature. The majority of industrial systems require configuring new device which has been connected to the system — a configuration of the connections (e.g., IP address of the server where data will be stored), a configuration of a corresponding database, and configuration of systems responsible for subsequent data processing.

In this section, an example of the semantic integration of CPS components is introduced. The example faces the problem of adding/removing a component (a sensor or an actuator) to/from a complex CPS more or less automatically. This approach is named as "Plug&Play CPS components" after the similarity with the well-known Plug&Play concept designed by Microsoft company.

The idea of Plug&Play CPS components was motivated by complex industrial systems (e.g., a complex production workstation corresponding to a CPS) where the connection of a new part may cause a non-trivial task from many perspectives including a configuration or a data model adjustment. There may be distinguished the following issues related to the connection of a device to a CPS:

- Collect data from a sensor or provide feedback to a physical process by executing an action by an actuator.
- Identify and interpret collected data.
- Add a new previously unknown CPS concept into a system.
- Process/Analyze collected data.

Obviously, some of the mentioned issues depend on each other or overlap — data cannot be processed without their acquisition; data cannot be processed without proper data understanding, etc. In the case of a CPS component removal, all system components have to be aware of the missing device to prevent unplanned and dangerous system behaviors.

Our proposed solution to this problem is built on the following concepts:

- Development of a versatile data model which will serve as a knowledge base for the whole system and will also provide an easy way how to extend the model.
- A versatile format for a knowledge representation of device's data model (inside the device).
- A suitable way for communication between a device and a CPS.
- An appropriate description of a device which ensures an unambiguous device identification.
- Providing a flexible schema of CPS to ensure automatic integration of a new CPS component.
- An approach which allows utilizing a new CPS component without any additional configuration of surrounding systems.

This approach is illustrated and discussed on our developed Semantic Big Data Historian and experimental CPS which was proposed to resolve the stop-problem of the hydroelectric power plant.

11.1 Data Model

As already mentioned, the semantic web technologies are suitable for enabling suggested Plug&Play feature. Properly designed ontology facilitates easy incorporation of changes (modify/add/remove ontology elements reflecting real-world entities) and re-usability of the model thus it should be suitable for a description of complex systems.

Every sensor or actuator is represented as a particular individual of a given concept. Then, only one triple (expressing that the individual belongs to the concept) is required for full identification of a given device. The example of such triple accompanied by triples defining corresponding concepts are listed on the Lst. 11.2. With this simple identification, a user or a system has a full overview of available CPS parameters, its methods as well as their proper meanings. Furthermore, if a suitable concept is not available for some sensor or actuator then the new device (i.e., the triples representing the missing concept in the ontology) may be added automatically in certain case — a manufacturer of the device has to use some of the already existing ontology concepts as a predecessor of the new concept. This approach is demonstrated on the proposed COCI ontology which is described in section 8.1.

11.2 Data Storage

The important part of Plug&Play CPS component is the data storage. This storage has to be based on a general and flexible schema which makes the dynamic device management possible. The common problem comes up when a new device is added, and a corresponding table/object for data storage is missing in a database. The automatic or at least semi-automatic table management has to be solved for enabling Plug&Play feature. This task poses problems in many legacy database systems due to system, data, and user security policies. Schemas of legacy database systems are usually rigid and even a small change may cause big problems. On the other hand, RDF has no such limitations.

If ontology model is considered, then data are represented as RDF triples and may be stored in RDF database, i.e., a triple store. An internal structure of every triple store may vary according to a provider, but triple stores have the common feature —

a user does not take care of where to store data. All data (RDF triples) are upload to the triple store in the same way with the same command. A triple store itself is in charge of subsequent distribution to corresponding tables or files. It is important to mention here that the majority of triple stores have the insufficient performance for industrial applications. This deficiency may be solved for example by the utilization of Big Data approach [121] or by advanced triple storage model [122].

11.3 Subsequent Data Processing

The whole idea is more beneficial with an appropriate connection to the other CPS components or the surrounding systems which use data from the CPS for subsequent processing. When the new table with data from a new device is added to legacy database systems, the corresponding data are not used for processing because the surrounding systems are not aware of the new structure. This requirement may be expressed as a support for an ontology-driven application development for accepting changes in the number of individuals as well as accepting new concepts. For example, if one developed his application following common approaches then he needs to re-generate the code of applications in the case of changing data model (especially when new classes/concepts are added).

On the other hand, these drawbacks may be overcome using the semantic web technologies. Ontology query language — SPARQL — together with inferencing offers means how to keep the current view of the relevant parts of the modeled world. SPARQL provides options how to acquire current model structure easily and allows to assemble queries close to the human thinking. The query for finding out temperature sensors contained in a particular generator together with a simple reasoning is illustrated in Listing 11.1.

Listing 11.1: Listing all sensors in generator.

```
PREFIX : <http://www.loa-cnr.it/ontologies/DUL.owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX COCI: <http://www.rockwellautomation.com/RADIC/COCI#>
PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
```

```
SELECT ?sensor
WHERE {?sensor :hasLocation COCI:Generator .
  ?sensor a ?sensorType .
  ?sensorType  rdfs:subClassOf+ COCI:temperatureSensor
}
```

This way of querying is more straightforward than the usage of legacy database systems where you need the right tables for the query. Furthermore, the reasoning offers other helpful possibilities as consistency checking, subsumption of concepts, retrieval of individuals, etc.

11.4 Work Flow

The workflow during the connection of CPS component to the system for clarity and a better understanding of the Plug&Play principle is summarized in the following paragraphs. The workflow may be described as follows:

- Development of CPS component.
- Connection of CPS component.
- Initialization of communication.
- Relevant operation between component and the CPS.
  - Data retrieval.
  - Data storage.
  - Calling methods of a given CPS component.
  - Etc.
- Data querying and processing.

The essential prerequisite is the compatible device itself. It lies in ensuring compatibility with communication API of the corresponding CPS and in defining semantic metadata in the device. The prototype is built on OPC UA standard which offers means for interconnection of devices as well as versatile object model

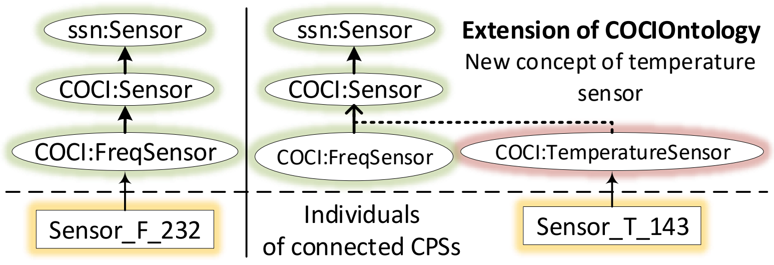


Figure 11.1: Connection of known and unknown sensor.

for modeling corresponding data. The possible way, how to store semantic metadata, is described in the next section.

The configured component is then connected to a CPS. There are two options as they were already described — connection of previously defined and not defined device in the ontology. If a previously known device is connected then data may be directly moved forward storage or another processing. On the other hand, if the unknown device is connected and the new concept included in the semantic metadata is compatible with the ontology then the ontology could be extended, and the device may be used as needed. In this case, security aspects have to be taken into consideration. The connection of not verified data could cause unpredictable situations. These two options are illustrated in Fig. 11.1.

Next, the communication has to be initialized. Thus, a user needs to set a connection string to the CPS. This may be done in the configuration phase of a device or before connection of the device to the CPS. This step may be skipped in the case of utilization of OPC UA server/client communication — OPC UA standard provides a possibility for easy device discovery, i.e., local and global discovery servers[1]. The following task is an identification of the device according to the ontology. The metadata of the device is parsed and compared with the ontology.

If a device is connected then data of a device may be correspondingly processed and stored. Finally, stored device data may be used for processing by the CPS, surrounding systems or queried by a user.

[1]https://opcfoundation.org/developer-tools/specifications-unifiedarchitecture/part-12-discovery/

11.5 Validation Study

In this section, a prototype implementation of the Plug&Play CPS component, and CPS component deployment with the help of the Semantic Big Data Historian is described. The prototype implementation resides in an extension of a sensor data model by a metadata with RDF triples defining the sensor. Adapted sensors are subsequently used in the experimental CPS which is applied to the hydroelectric power plant "stop-problem".

11.5.1 Semantic Extension of Sensor Data Model

Two different approaches to representing OWL triples defining a given device were proposed. The first approach extends the common way of data representation and communication. The extension resides in the creation of a string variable where annotation of the device is stored. The second approach is based on modeling of the correspondent data model within the particular format. This method assumes that the used data format has adequate expressional capabilities.

RDF Representation in String Variable

The first way, how to include a semantic annotation in a device data model is to store a corresponding part of the RDF file directly into a string variable. The variable is named *COCIMetadata* in our case. The extended data model corresponding to a *COCISensor* is depicted in the Fig. 11.3. This concept of COCI ontology named *COCISensor* is a general concept and acts as a superclass of particular sensor concepts (e.g., *FrequencySensor* which is also shown in the Fig. 11.3).

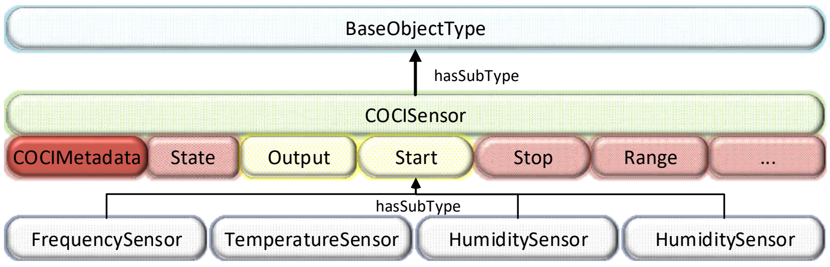


Figure 11.2: RDF Representation in the string constant — COCIMetadata.

Next, the example of the COCIMetadata content for a specific frequency sensor is shown in the Listing 11.2. The frequency sensor definition is presented in the turtle language format for better readability. Besides definition of the sensor class, the COCIMetadata variable contains information about individual attributes, e.g., individual name.

Listing 11.2: Frequency sensor definition in turtle language.

```
<http://www.rockwellautomat...COCI#FrequencySensor> rdf:type :Class;
            "rdfs: subClassOf <http://www.purl...ssn#SensingDevice>,
                                             [rdf: type :Restriction;
                         :onProperty <http://www.purl...ssn#observes>;
           :hasValue <http://www.rockwellautomat...COCI#frequency>].

<http://www.ro...COCI#Sensor_F1> rdf:type <...COCI#FrequencySensor>,
                                                     :NamedIndividual.
```

This approach brings many advantages. The implementation is easy even in conventional communication standards and this realization is possible everywhere the string variables are allowed. Next, no special adapters for a semantic information processing are needed — a content of COCIMetadata is processable by RDF tools and libraries (e.g. owlapi[2]). Moreover, this approach reflects the original intention of OPC UA (sharing common data models), i.e., specifications of node sets such as OPC UA Field Device Integration (FDI), OPC UA MTConnect Companion Specification (MTConnect), OPC Analyzer Devices Integration (ADI), etc. On the other hand, the expression of semantic information is hard to understand in comparison with the approach which is introduced in the following paragraphs. A disadvantage of this way may be seen in a redundant data storing - entities and their relations are modeled in an ontology as well as in an OPC UA model. It may cause a possible tendency to lose a data integrity during an incorporation of model changes.

[2]http://owlapi.sourceforge.net

OPC UA Model of the Ontology

The second possible approach is to model the related part of an ontology directly in a suitable standard — for example, OPC UA. A conversion from an ontology model to OPC UA data model and vice versa are described in [123, 124]. It is important to remark there are some limitations in the conversion from ontology to OPC UA caused by different expressivity. However, it is sufficient for proper modeling of all relevant entities which are used for the definition of CPS components as well as their identification according to the COCI ontology.

The main advantage of this approach is easier data understanding compared to the previously mentioned representation. On the other hand, an adapter has to be implemented for transforming data from OPC UA model to the RDF format.

11.5.2 Plug&Play CPS Component

As mentioned, the Plug&Play property of a device is not ensured only by a given device but also by a whole CPS which is responsible for device identification according to a shared ontology. It is represented by SBDH in our case. The overall architecture of SBDH with connected Plug&Play components is illustrated in the Fig. 11.3.

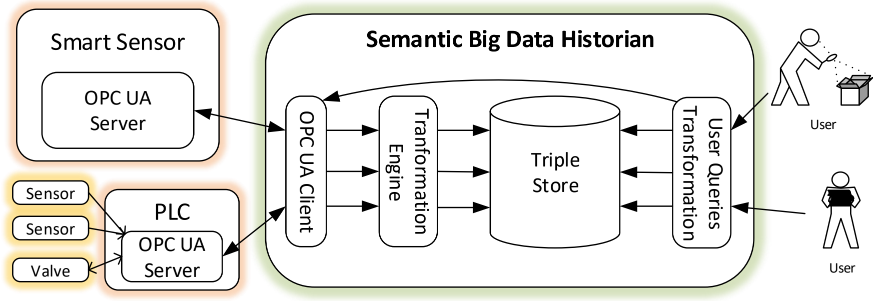


Figure 11.3: Plug&Play sensors in Semantic Big Data Historian.

The Plug&Play approach has been modeled and verified with the help of previously described stop-problem of the hydro-electric power plant. The power plant is equipped with 38 sensors in the power plant including for example measurement of fall of water, frequency, power factor, and real power. The Plug&Play approach facilitates from mentioned COCI ontology which describes all of the power plant sensors and actuators together with data from external data sources like online information covering meteorological data (temperature and precipitation) and hydrological data (rate of flow, water level, and water temperature) for relevant locations.

All data from power plant sensors are read with 5-second sampling rate. The amount of data means significant problem — the performance issue of processing huge amount of RDF data. Our sensors produce 656,640 samples per day. If these data are transformed into triples, then the volume of data is equal to 5,253,120 triples per day and it corresponds to 1,917 mil. triples per year. Such an amount of fast-produced sensor data should be considered as a big data according to NIST, and horizontal scalability is for efficient processing [93]. The processing of such amount of RDF triples is very time consuming or impossible in the case of common triple stores. A way how to face the performance issue of processing huge amount of RDF-triples is presented in [122].

COCIMetadata is represented by means of string variable based on conducted experiments. This approach has the main advantage in COCIMetada processing by available libraries (e.g., OWL API[3]).

Smart sensors are connected using OPC UA standard. Milo[4] OCP UA library is used for implementing OPC UA servers (located in the CPSs) and OPC UA client (located in SBDH). Smart sensor (frequency sensor) information model of OPC UA servers is illustrated in the Fig. 11.2.

Next, the transformation engine identifies COCIMetadata (see Lst. 11.2) of the connected smart sensor according to COCI ontology. If the smart sensor concept is not defined in the ontology and the super concept of smart sensor is in the ontology, then the ontology is correspondingly extended. The connections of a known sensor (frequency sensor) and unknown sensor (temperature sensor) are illustrated in the Fig. 11.1. Subsequently, measurements are stored in the historian triple store.

11.5.3 Localization

Very important and interesting information to be provided by the self-describing sensors is their localization. This issue is beyond the scope of this work. However, the

[3]http://owlapi.sourceforge.net
[4]https://projects.eclipse.org/proposals/milo

localization of CPSs is supposed to be implemented according to a given application. Unfortunately, there is no versatile solution suitable for all possible applications.

At first, a location of CPSs is stored in COCI Ontology with the help of concepts defined in DOLCE ontology. It provides properties for a location, an approximate location, a descriptive place, a participant place, a situation place, etc.

The appropriate localization method depends on a given application as well as on a used connection technology. If localization of CPSs is considered within a complex machine or production line and CPSs are connected via an ethernet connection, then localization may be easily done by static IP addresses and predefined location of ethernet plugs stored in the ontology.

The problem becomes more complex when CPSs are connected via wireless technologies. There are many already implemented and promising solutions. For example, NextMe [125] solution is localization based using cellular traces in an internet of things which may be generalized for a wider set of applications. A large set of solutions use for localization wireless networks [126, 127] including industrial wireless sensor networks.

The most straightforward and the easiest solution is the storage of a location of the CPS in COCIMetadata directly in RDF. On the other hand, this approach makes CPS interchangeability more difficult and goes against the idea of Plug&Play devices.

11.6 Summary

In this chapter, the concept of Plug&Play CPS components was introduced. This concept may be understood as a low-level integration of CPSs, but the solution concepts may be utilized for high-level integration of CPSs as well. The proposed solution was verified on a solution of the stop-problem of the hydroelectric power plant.

The proposed Plug&Play concept facilitates the change management in the context of industrial CPSs. The configuration of a new CPS component is time-consuming as well as error-prone. In the perspective of Plug&Play devices, the whole configuration responsibility is moved to the design phase and is mostly repeatable for a given sensor or actuator. Therefore it is less expensive, more robust and resistant to configuration errors and very importantly more modular — CPS components may be easily reused.

Based on the conducted research, it is obvious the Plug&Play concept does not rely only on the utilization of OWL format for data representation and changes of devices data model but also changes of an overall CPS data model and a specific way of analyzing the data from a connected device. In detail, the research achievements may be summarized as follows:

- OWL was chosen as the suitable format for data representation. This format has two main advantages — a meaning of a particular device and its related properties may be properly understood without any context. A reasoning may be employed for advanced analytics.
- After experiments, the COCImetadata representation as a string variable seems to be more promising than the other one. The main advantages are a wide spectrum of data formats which can handle data model extension. Next, a utilization of existing software of OWL files processing.
- The last significantly important aspect is data analytics. The analytic systems have to be driven by purposes of relevant devices instead of hard-coded algorithms.

Part IV

Conclusions and Discussions

Chapter 12

Conclusions

In this chapter, the content of the dissertation thesis is summarized together with the evaluation of the thesis goals fulfillment, and the summary of the main thesis contributions. Next, future research directions are introduced.

12.1 Summary and Discussion

The introduction specified the goals of this dissertation thesis and is followed by the state-of-the-art relevant to the thesis.

The term Cyber-Physical System becomes well-known with the help of Industry 4.0, but this work attempted to show that the origin of these devices is older than the fourth industrial revolution. CPSs are not only a simply interconnected physical and computational processes but may be represented by more complex devices composed of many integrated components, and moreover, CPSs may also include a human or a cloud part. The complex nature of CPSs demands a complete understanding of all CPS components. It may be a difficult requirement, especially in the case when components are produced by various designers. Thus, the main obstacle for perfect integration of CPS components is resolving of the semantic heterogeneity.

An integration of a CPS components may be facilitated by ontologies and Semantic Web technologies which may serve as a data model of a CPS. This thesis introduced and explained the well-known ontology definition: "An ontology is a formal, explicit specification of a shared conceptualization." This definition is supplemented with the rigorous ontology definition which should offer a better understanding of the main ontology constituents. Furthermore, various ontology representations with the focus on languages for Semantic Web were discussed.

The problem of an information integration was discussed together with an introduction of the main already mentioned obstacle in the information integration process — data heterogeneity. Ontology matching methods were described because they may be helpful in finding relevant elements from a given data models, i.e., in a reduction of the semantic heterogeneity. Next, suitable formats for a representation of CPS related data were described. First, special attention was paid to a usage of upper ontologies (mainly SUMO and DOLCE) for supporting the task of an information integration. Next, XML and derived neutral formats for representation and interchanging information related to CPS components were described.

An employment of an ontology for a knowledge representation causes in performance problems during data handling. It is one of the most significant obstacles which slow down adoption of semantic technologies within the industrial domain. Thus, Big Data paradigm and corresponding frameworks, which may offer a solution to the performance problems, were introduced.

12.1.1 Semantic Integration in the Context of Cyber-Physical Systems

Two different levels of the integration challenge of CPSs were identified — the low-level and the high-level integration. These different integrations suffer from similar problems, and therefore the defined steps for the solution of the integration process (resolving of the platform, syntactic, and semantic heterogeneity) may be applied on both of them.

This work is focused on the low-level integration problem, i.e., an integration of CPS components. The platform and the syntactic heterogeneity may be resolved by an implementation of corresponding adapters straightforwardly. Unfortunately, a solution of the semantic heterogeneity is more complex, and the essential requirement for its correct solution is based on the precise identification of corresponding elements. Thus, a solution, which benefits from a utilization of ontologies and ontology matching methods to deal with the semantic heterogeneity of various CPS components, was proposed.

As the first requirement for resolving the semantic heterogeneity, Component Ontology for Cyber-physical system Integration (COCI) was developed. This ontology is able to describe sensors, actuators, and related processes and actions. The main benefit of the ontology utilization is the capability of unambiguous identification of devices which may be not misinterpreted even without a given context. Moreover, the proposed solution also offers additional benefits:

- Ontology matching methods may be exploited more precisely for identification of corresponding entities.
- A reasoner may be employed for providing consistency checking, a possibility of inferring not explicitly described information as well as a simplification of user queries.
- A possibility to easily re-use a particular solution in other applications.

The MAPSOM framework was proposed to enable utilization of ontology matching methods in the industrial automation domain. A self-organizing map and active learning were successfully employed for providing semi-automatic ontology matching framework which may achieve better precision and recall then automatic matching systems and also less strenuous task than manual matching. The system was evaluated using OAEI benchmarks, and the system proved to be a suitable tool for matching various entities, e.g., concepts, individuals, etc.

After the verification of basic concepts, Semantic Big Data Historian was designed and implemented to ensure the feasibility of the proposed semantic heterogeneity reduction of a CPS. The historian is based on Apache Spark and Apache Cassandra, which provide adequate flexibility and performance for the satisfaction of our demands. Furthermore, the hybrid SBDH model was defined for effective handling of RDF statements.

The overall approach was tested on the experimental CPS which was proposed to resolve the hydroelectric power plant “stop-problem”. It was verified that the proposed solution for semantic integration of CPS components is meeting all requirements for successful deployment, i.e., sufficient expressivity for the description of the components and related processes as well as sufficient efficiency.

Moreover, this dissertation thesis defines the Plug&Play concept which benefits from the exploitation of the Semantic Web technologies for the description of CPS components and utilization of OPC UA. It represents an example of the integration of CPS components based on the unambiguous description of devices together with the shared data model. The Plug&Play concept enables a connection a new device to the CPS with minimal configuration. Furthermore, a newly connected device may automatically extend the shared ontology of CPS when the device is unknown.

12.2 Fulfillment of the Thesis Goals

In this dissertation thesis, all of the five demanding goals, which were described in the introduction, were achieved.

1. The problem of CPSs integration was analyzed, and two different levels of a CPS integration were determined — the low-level and the high-level integration. Solutions for the relevant types of heterogeneity were identified. First, the platform heterogeneity and a correspondent solution by means of adapters which unifies different interfaces. Next, the syntactic heterogeneity and a solution by a unification of different formats with the help of suitable adapters. Finally, the semantic heterogeneity with a solution concerning an identification of corresponding elements by means of ontology matching methods.

2. An ontology was identified as a suitable format for a knowledge representation of a CPS components. The reusability of components is possible only because of a proper understanding of components' data models and accepting data models within a correspondent community. The proposed solution was built on SSN ontology, and SSN ontology was extended by required concepts for actuators. The easy reusability of the ontology is also ensured by its interconnection with DOLCE ontology.

3. The solution for an integration of heterogeneous data models was proposed. The solution is based on semi-automatic ontology matching which may be gen-

eralized for matching various formats not only ontologies. The proposed system employed self-organizing map for similarity measures aggregations. User involvement is ensured by various visualization methods (hit histogram, U-Matrix, visualization of clusters) as well as by active learning. The solution was evaluated with the help of OAEI benchmarks, and the outcomes were very promising even before user involvement, i.e., after the initial phase.

4. This work identified that the solution concerning a representation of CPS components in RDF triples may cause performance problems when a huge number of RDF statements has to be processed. Thus, the framework named Semantic Big Data Historian, which employes a Big Data framework (Apache Spark together with Apache Cassandra) for overcoming performance problems, was proposed and implemented. The proposed solution was deployed as Spark and Cassandra cluster and subsequently was evaluated on the hydroelectric power plant. Furthermore, the solution includes also Hybrid SBDH Model for time-series related data storage.

5. The proposed solution was utilized for an implementation of the concept of Plug&Play CPS components. It demonstrates how semantic technologies may be successfully used for integration within a CPS. Plug&Play concept cornerstone resides in a description of the device (CPS component) with the help of the ontology and subsequent immediate device utilization without any additional configuration of a system.

12.3 Contributions of the Dissertation Thesis

In the dissertation thesis, the solution for CPS low-level semantic integration was proposed, and subsequently, this work proved that the integration based on Semantic Web technologies is feasible. It is due to the concept of Semantic Big Data Historian which is based on the Big Data framework (Apache Spark) and the suitable data storage (Apache Cassandra).

This thesis also verified that the feasibility of such an integration is not only about computational power (i.e., Big Data employment) but also about data representation — hybrid SBDH model for storage of RDF statements.

Furthermore, it was shown that ontology matching methods are applicable for an integration even within industrial automation domain where the highest precision and recall are required. The approach is based on similarity measures aggregation using self-organizing maps and active learning for user involvement.

Finally, it was shown that ontologies together with appropriate technologies (e.g., OPC UA) may facilitate not only the process of integration but also processes such as configuration. This concept is called Plug&Play CPS components.

12.4 Future Work

The presented work is the first step on the way for improving interoperability of cyber-physical systems using ontologies. Nevertheless, many further obstacles have to be overcome for an adoption of this approach within the industrial domain. The author of the dissertation thesis suggests to explore the following:

- A possible extension of the solution to be capable involve humans for an exploitation of their knowledge, mental capabilities, and operational capabilities. This research direction is important because a human involvement within a CPS should reduce costs of a CPS, preserve jobs, and provide operations which are not achievable by devices.

- An additional improvement could be provided by an implementation of the remaining SPARQL constructs. Translations between simple SPARQL queries and CQL (Cassandra Query Language) were already implemented but providing complete SPARQL interface should ensure easier utilization of SBDH.

- The implementation of the proposed approach could be further improved by a formalization how to utilize a local or/and a global OPC UA discovery servers. Primary experiments with OPC UA discovery servers were conducted, but proper methodology and formalization of these servers together with their exploitation could exploit all servers capabilities.

- Information about a localization of a CPS may be crucial in many applications. There are two different research problems — a suitable representation of localization in a data model and an automatic determination of a CPS location.

Bibliography

[1] e.V. c/o IAF, A. AutomationML - story. Available from: `https://www.automationml.org/o.red.c/story.html`

[2] Siemens, P. Software: Open product lifecycle data sharing using XML. *White Paper*, 2011.

[3] W3C Semantic Sensor Network Incubator group. Report Work on the SSN ontology. Available from: `https://www.w3.org/2005/Incubator/ssn/wiki/Report_Work_on_the_SSN_ontology`

[4] Cao, X.; Liu, L.; Shen, W.; et al. Real-Time Misbehavior Detection and Mitigation in Cyber-Physical Systems over WLANs.

[5] Paredes, A. C.; Malfaz, M.; Salichs, M. A. Signage system for the navigation of autonomous robots in indoor environments. *IEEE Transactions on Industrial Informatics*, volume 10, no. 1, 2014: pp. 680–688.

[6] Georg, H.; Müller, S. C.; Rehtanz, C.; et al. Analyzing cyber-physical energy systems: The INSPIRE cosimulation of power and ICT systems using HLA. *IEEE Transactions on Industrial Informatics*, volume 10, no. 4, 2014: pp. 2364–2373.

[7] Visser, P. R.; Jones, D. M.; Bench-Capon, T. J.; et al. Assessing heterogeneity by classifying ontology mismatches. In *Proceedings of the FOIS*, volume 98, 1998.

[8] Cao, X.; Liu, L.; Shen, W.; et al. Real-time misbehavior detection and mitigation in cyber-physical systems over WLANs. *IEEE Transactions on Industrial Informatics*, 2015.

[9] Lee, E. A.; Seshia, S. A. *Introduction to embedded systems: A cyber-physical systems approach.* MIT Press, 2016.

[10] Kim, K.-D.; Kumar, P. R. Cyber–physical systems: A perspective at the centennial. *Proceedings of the IEEE*, volume 100, no. Special Centennial Issue, 2012: pp. 1287–1308.

[11] Eckert, J.; Mauchly, J. Outline of plans for development of electronic computers. *Available: http://archive. computerhistory. org/resources/access/text/2010/08/102660910-05-01-acc. pdf*, 1946.

[12] Leiner, B. M.; Cerf, V. G.; Clark, D. D.; et al. A brief history of the Internet. *ACM SIGCOMM Computer Communication Review*, volume 39, no. 5, 2009: pp. 22–31.

[13] Kahn, J. M.; Katz, R. H.; Pister, K. S. Next century challenges: mobile networking for Smart Dust. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, ACM, 1999, pp. 271–278.

[14] Black, H. S. Stabilized feed-back amplifiers. *Electrical Engineering*, volume 53, no. 1, 1934: pp. 114–120.

[15] Nyquist, H. Regeneration theory. *Bell Labs Technical Journal*, volume 11, no. 1, 1932: pp. 126–147.

[16] Ahmadi, H.; Abdelzaher, T.; Han, J.; et al. The sparse regression cube: A reliable modeling technique for open cyber-physical systems. In *Proceedings of the 2011 IEEE/ACM Second International Conference on Cyber-Physical Systems*, IEEE Computer Society, 2011, pp. 87–96.

[17] Pajic, M.; Jiang, Z.; Lee, I.; et al. From verification to implementation: A model translation tool and a pacemaker case study. In *2012 IEEE 18th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, IEEE, 2012, pp. 173–184.

[18] Kleissl, J.; Agarwal, Y. Cyber-physical energy systems: Focus on smart buildings. In *Proceedings of the 47th Design Automation Conference*, ACM, 2010, pp. 749–754.

[19] Zhang, W.; Kamgarpour, M.; Sun, D.; et al. A hierarchical flight planning framework for air traffic management. *Proceedings of the IEEE*, volume 100, no. 1, 2012: pp. 179–194.

[20] Ferber, S. Industry 4.0 Germany takes first steps toward the next industrial revolution. 2012. Available from: `http://blog.bosch-si.com/categories/manufacturing/2012/10/industry-4-0-germany-takes-first-steps-toward-the-next-industrial-revolution/`

[21] Jazdi, N. Cyber physical systems in the context of Industry 4.0. In *2014 IEEE International Conference on Automation, Quality and Testing, Robotics*, IEEE, 2014, pp. 1–4.

[22] Khaitan, S. K.; McCalley, J. D. Design techniques and applications of cyber-physical systems: A survey. *IEEE Systems Journal*, volume 9, no. 2, 2015: pp. 350–365.

[23] Wagh, A.; Li, X.; Wan, J.; et al. Human centric data fusion in vehicular cyber-physical systems. In *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2011, pp. 684–689.

[24] Huang, H.; Sun, Y. L.; Yang, Q.; et al. Integrating neuromuscular and cyber systems for neural control of artificial legs. In *Proceedings of the 1st ACM/IEEE international conference on cyber-physical systems*, ACM, 2010, pp. 129–138.

[25] Li, X.; Lu, R.; Liang, X.; et al. Smart community: an internet of things application. *IEEE Communications Magazine*, volume 49, no. 11, 2011.

[26] Wang, L.; Törngren, M.; Onori, M. Current status and advancement of cyber-physical systems in manufacturing. *Journal of Manufacturing Systems*, volume 37, no. Part 2, 2015: pp. 517–527.

[27] Ferreira, P.; Lohse, N.; Ratchev, S. Multi-agent architecture for reconfiguration of precision modular assembly systems. *Precision Assembly Technologies and Systems*, 2010: pp. 247–254.

[28] Wang, L.; Wang, X. V.; Gao, L.; et al. A cloud-based approach for WEEE remanufacturing. *CIRP Annals-Manufacturing Technology*, volume 63, no. 1, 2014: pp. 409–412.

[29] Liu, Z.; Yang, D.-s.; Wen, D.; et al. Cyber-physical-social systems for command and control. *IEEE Intelligent Systems*, volume 26, no. 4, 2011: pp. 92–96.

[30] Colombo, A. W.; Bangemann, T.; Karnouskos, S.; et al. Industrial cloud-based cyber-physical systems. *The IMC-AESOP Approach*, 2014.

[31] Monostori, L. Cyber-physical production systems: Roots, expectations and R&D challenges. *Procedia CIRP*, volume 17, 2014: pp. 9–13.

[32] Neches, R.; Fikes, R. E.; Finin, T.; et al. Enabling technology for knowledge sharing. *AI magazine*, volume 12, no. 3, 1991: p. 36.

[33] Berg, J. Aristotles theory of definition. *ATTI del Convegno Internazionale di Storia della Logica*, 1982: pp. 19–30.

[34] Schopenhauer, A.; Payne, E. F. J. *Parerga and Paralipomena: short philosophical essays*, volume 1. Oxford University Press, 1974.

[35] Kant, I. Logic. Translated by Robert S. Hartman and Wolfgang Schwarz. 1988.

[36] Smith, B. Beyond concepts: ontology as reality representation. In *Proceedings of the third international conference on formal ontology in information systems (FOIS 2004)*, IOS Press, Amsterdam, 2004, pp. 73–84.

[37] Boyer, C. B. *The history of the calculus and its conceptual development.* Dover, 1959.

[38] Novak, J. D.; Cañas, A. The Theory Underlying Concept Maps and How to Construct Them. 2006. *Florida Institute for Human and Machine Cognition: Pensacola, FL*, 2010.

[39] Baader, F.; Nutt, W. Basic description logics. In *Description logic handbook*, 2003, pp. 43–95.

[40] Petersen, W. Representation of concepts as frames. *The Baltic international yearbook of cognition, logic and communication*, volume 2, 2007: pp. 151–170.

[41] Staab, S.; Studer, R. *Handbook on ontologies.* Springer Science & Business Media, 2010.

[42] Guber, T. A Translational Approach to Portable Ontologies. *Knowledge Acquisition*, volume 5, no. 2, 1993: pp. 199–229.

[43] Borstw, N. Construction of engineering ontologies. *Enschede: University of Twente*, 1997.

[44] Studer, R.; Benjamins, V. R.; Fensel, D. Knowledge engineering: principles and methods. *Data & knowledge engineering*, volume 25, no. 1, 1998: pp. 161–197.

[45] Garrido, A. Logical Foundations of Artificial Intelligence. *BRAIN. Broad Research in Artificial Intelligence and Neuroscience*, volume 1, no. 2, 2010: pp. 149–152.

[46] Carnap, R. *Meaning and necessity: a study in semantics and modal logic.* University of Chicago Press, 1988.

[47] Stumme, G.; Ehrig, M.; Handschuh, S.; et al. The Karlsruhe view on ontologies. *Rapport technique, University of Karlsruhe, Institute AIFB, Germany.(Cité en pages 5, 8 et 13.)*, 2003.

[48] Buitelaar, P.; Cimiano, P.; Magnini, B. *Ontology learning from text: methods, evaluation and applications*, volume 123. IOS press, 2005.

[49] Corcho, O.; Gómez-Pérez, A. A roadmap to ontology specification languages. In *EKAW*, volume 2000, Springer, 2000, pp. 80–96.

[50] Genesereth, M. R.; Fikes, R. E.; et al. Knowledge interchange format-version 3.0: reference manual. 1992.

[51] Fensel, D. Ontologies. In *Ontologies*, Springer, 2001, pp. 11–18.

[52] Gruber, T. R. Ontolingua: A mechanism to support portable ontologies. 1992.

[53] Kifer, M.; Lausen, G. F-logic: a higher-order language for reasoning about objects, inheritance, and scheme. In *ACM SIGMOD Record*, volume 18, ACM, 1989, pp. 134–146.

[54] Baader, F. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.

[55] Horrocks, I.; Sattler, U.; Tobies, S. Practical reasoning for very expressive description logics. *Logic Journal of IGPL*, volume 8, no. 3, 2000: pp. 239–263.

[56] Berners-Lee, T.; Hendler, J.; Lassila, O.; et al. The semantic web. *Scientific american*, volume 284, no. 5, 2001: pp. 28–37.

[57] Broekstra, J. Storage, querying and inferencing for semantic web languages. 2005.

[58] Karvounarakis, G.; Christophides, V.; Plexousakis, D.; et al. Querying community web portals. 2000.

[59] Seaborne, A. Rdql-a query language for rdf. *W3C Member submission*, volume 9, no. 29-21, 2004: p. 33.

[60] Doan, A.; Halevy, A.; Ives, Z. *Principles of data integration*. Elsevier, 2012.

[61] Goh, C. H. *Representing and reasoning about semantic conflicts in heterogeneous information systems*. Dissertation thesis, Massachusetts Institute of Technology, 1996.

[62] Wache, H.; Voegele, T.; Visser, U.; et al. Ontology-based integration of information-a survey of existing approaches. In *IJCAI-01 workshop: ontologies and information sharing*, volume 2001, Citeseer, 2001, pp. 108–117.

[63] Bouquet, P.; Euzenat, J.; Franconi, E.; et al. D2. 2.1 Specification of a common framework for characterizing alignment. 2004.

[64] Euzenat, J.; Shvaiko, P. *Ontology matching*, volume 18. Springer, 2007.

[65] Hammer, J. Resolving Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous database Systems. Technical report, Stanford, 1994.

[66] Hammer, J.; McLeod, D. An approach to resolving semantic heterogeneity in a federation of autonomous, heterogeneous database systems. *International Journal of Intelligent and Cooperative Information Systems*, volume 2, no. 01, 1993: pp. 51–83.

[67] Jirkovský, V.; Kadera, P.; Rychtyckyj, N. *Semi-automatic Ontology Matching Approach for Integration of Various Data Models in Automotive*. Cham: Springer International Publishing, 2017, ISBN 978-3-319-64635-0, pp. 53–65, doi:10.1007/978-3-319-64635-0_5. Available from: `https://doi.org/10.1007/978-3-319-64635-0_5`

[68] Abiteboul, S.; Buneman, P.; Suciu, D. *Data on the Web: from relations to semistructured data and XML*. Morgan Kaufmann, 2000.

[69] Systemes, D. 3D XMLa universal, lightweight XML-based format. 2011.

[70] Katzenbach, A.; Handschuh, S.; Vettermann, S. JT Format (ISO 14306) and AP 242 (ISO 10303): The Step to the Next Generation Collaborative Product Creation. In *Digital Product and Process Development Systems*, Springer, 2013, pp. 41–52.

[71] Drath, R.; Luder, A.; Peschke, J.; et al. AutomationML-the glue for seamless automation engineering. In *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, IEEE, 2008, pp. 616–623.

[72] Arnaud, R.; Barnes, M. C. *COLLADA: sailing the gulf of 3D digital content creation*. CRC Press, 2006.

[73] van der Wal, E. PLCopen. *IEEE Industrial Electronics Magazine*, volume 3, no. 4, 2009: p. 25.

[74] Commission, I. E.; et al. IEC 62424. *Representation of process control engineering-Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools*, 2008.

[75] AutomationML consortium. Application Recommendations: Automation Project Configuration. 2016.

[76] Foundation, O. Unified Architecture. Available from: `https://opcfoundation.org/about/opc-technologies/opc-ua/`

[77] Mahnke, W.; Leitner, S.-H. *OPC Unified Architecture*. Springer, 2009.

[78] Brandl, D.; Consulting, B. What is ISA-95. *Industrial Best Practices of Manufacturing Information Technologies with ISA-95 Models, BR&L Consulting*, 2008.

[79] Busch, J. E.; Lin, A. D.; Graydon, P. J.; et al. Ontology-based parser for natural language processing. Apr. 11 2006, uS Patent 7,027,974.

[80] Auer, P.; Billard, A.; Bischof, H.; et al. A research roadmap of cognitive vision. *IST project IST-2001-35454*, 2005.

[81] Obitko, M.; Vrba, P.; Kadera, P.; et al. System And Method For Implementing A User Interface To A Multi-Agent Distributed Control System. Feb. 28 2013, uS Patent 20,130,055,115.

[82] Lee, C.-S.; Jiang, C.-C.; Hsieh, T.-C. A genetic fuzzy agent using ontology model for meeting scheduling system. *Information Sciences*, volume 176, no. 9, 2006: pp. 1131–1155.

[83] Oberle, D.; Ankolekar, A.; Hitzler, P.; et al. DOLCE ergo SUMO: On foundational and domain models in the SmartWeb Integrated Ontology (SWIntO). *Web Semantics: Science, Services and Agents on the World Wide Web*, volume 5, no. 3, 2007: pp. 156–174.

[84] Oberle, D. *Semantic management of middleware*, volume 1. Springer Science & Business Media, 2006.

[85] Niles, I.; Pease, A. Towards a standard upper ontology. In *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*, ACM, 2001, pp. 2–9.

[86] Masolo, C.; Borgo, S.; Gangemi, A.; et al. The wonderweb library of foundational ontologies. 2002.

[87] Gangemi, A.; Sagri, M.-T.; Tiscornia, D. A constructive framework for legal ontologies. In *Law and the semantic web*, Springer, 2005, pp. 97–124.

[88] Gangemi, A.; Fisseha, F.; Keizer, J.; et al. An overview of the FOS Project: towards a fishery Semantic Web. *Internal Project Report, ISTC-CNR, Laboratory for Applied Ontology, Rome, Italy*, 2002.

[89] Cox, S. Observations and Measurements (O&M). 2011. Available from: `http://www.opengeospatial.org/standards/om`

[90] Botts, M.; Robin, A. SensorML: Model and XML Encoding Standard 2.0. 2014. Available from: `http://portal.opengeospatial.org/files/55939`

[91] Janowicz, K.; Compton, M. The stimulus-sensor-observation ontology design pattern and its integration into the semantic sensor network ontology. In *Proceedings of the 3rd International Conference on Semantic Sensor Networks-Volume 668*, CEUR-WS. org, 2010, pp. 64–78.

[92] Wilder-James, E. What is big data? 2012. Available from: `https://www.oreilly.com/ideas/what-is-big-data`

[93] Mell, P.; Cooper, M. Tackling Big Data. 2012. Available from: `https://beta.csrc.nist.gov/Presentations/2012/Tackling-Big-Data`

[94] Ghemawat, S.; Gobioff, H.; Leung, S.-T. The Google file system. In *ACM SIGOPS operating systems review*, volume 37, ACM, 2003, pp. 29–43.

[95] Dean, J.; Ghemawat, S. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, volume 51, no. 1, 2008: pp. 107–113.

[96] Derler, P.; Lee, E. A.; Vincentelli, A. S. Modeling cyber–physical systems. *Proceedings of the IEEE*, volume 100, no. 1, 2012: pp. 13–28.

[97] Aircraft pressurization systems. Available from: `http://www.tpub.com/ase2/75.htm`

[98] Noy, N. F. Semantic integration: a survey of ontology-based approaches. *ACM Sigmod Record*, volume 33, no. 4, 2004: pp. 65–70.

[99] Szilagyi, I.; Wira, P. Ontologies and Semantic Web for the Internet of Things-a survey. In *Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE*, IEEE, 2016, pp. 6949–6954.

[100] Kohonen, T. The self-organizing map. *Proceedings of the IEEE*, volume 78, no. 9, 1990: pp. 1464–1480.

[101] Kaski, S.; Kohonen, T. Exploratory data analysis by the self-organizing map: Structures of welfare and poverty in the world. In *Neural networks in financial engineering. Proceedings of the third international conference on neural networks in the capital markets*, Citeseer, 1996.

[102] Settles, B.; Craven, M.; Friedland, L. Active learning with real annotation costs. In *Proceedings of the NIPS workshop on cost-sensitive learning*, 2008, pp. 1–10.

[103] Zhu, X.; Lafferty, J.; Rosenfeld, R. *Semi-supervised learning with graphs.* Carnegie Mellon University, language technologies institute, school of computer science, 2005.

[104] Lewis, D. D.; Gale, W. A. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, Springer-Verlag New York, Inc., 1994, pp. 3–12.

[105] Lewis, D. D.; Catlett, J. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the eleventh international conference on machine learning*, 1994, pp. 148–156.

[106] Needleman, S. B.; Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, volume 48, no. 3, 1970: pp. 443–453.

[107] Girbea, A.; Suciu, C.; Nechifor, S.; et al. Design and Implementation of a Service-Oriented Architecture for the Optimization of Industrial Applications. *IEEE Trans. Ind. Informat.*, volume 10, no. 1, Feb 2014: pp. 185–196, ISSN 1551-3203, doi:10.1109/TII.2013.2253112.

[108] Commission, I. E.; et al. IEC 62541: OPC Unified Architecture. *all parts), February*, 2010.

[109] SequenceFile. 2009. Available from: `https://wiki.apache.org/hadoop/SequenceFile`

[110] MapReduce. 2011. Available from: `https://wiki.apache.org/hadoop/MapReduce`

[111] Vohra, D. *Practical Hadoop Ecosystem: A Definitive Guide to Hadoop-Related Frameworks and Tools.* Apress, 2016.

[112] Aasman, J. Allegro graph: RDF triple database. *Cidade: Oakland Franz Incorporated*, 2006.

[113] Harris, S.; Gibbins, N. 3store: Efficient bulk RDF storage. 2003.

[114] Kolas, D.; Emmons, I.; Dean, M. Efficient linked-list rdf indexing in parliament. *SSWS*, volume 9, 2009: pp. 17–32.

[115] Wilkinson, K.; Sayers, C.; Kuno, H.; et al. Efficient RDF storage and retrieval in Jena2. In *Proceedings of the First International Conference on Semantic Web and Databases*, CEUR-WS. org, 2003, pp. 120–139.

[116] Accumulo. 2011. Available from: `https://accumulo.apache.org`

[117] Du, J.-H.; Wang, H.-F.; Ni, Y.; et al. HadoopRDF: A scalable semantic data analytical engine. In *International Conference on Intelligent Computing*, Springer, 2012, pp. 633–641.

[118] Schätzle, A.; Przyjaciel-Zablocki, M.; Lausen, G. PigSPARQL: Mapping SPARQL to pig latin. In *Proceedings of the International Workshop on Semantic Web Information Management*, ACM, 2011, p. 4.

[119] Rohloff, K.; Schantz, R. E. High-performance, massively scalable distributed systems using the MapReduce software framework: the SHARD triple-store. In *Programming Support Innovations for Emerging Distributed Applications*, ACM, 2010, p. 4.

[120] Husain, M.; McGlothlin, J.; Masud, M. M.; et al. Heuristics-based query processing for large RDF graphs using cloud computing. *IEEE Transactions on Knowledge and Data Engineering*, volume 23, no. 9, 2011: pp. 1312–1327.

[121] Obitko, M.; Jirkovský, V. *Big Data Semantics in Industry 4.0.* Cham: Springer International Publishing, 2015, ISBN 978-3-319-22867-9, pp. 217–229, doi: 10.1007/978-3-319-22867-9_19.

[122] Jirkovský, V.; Obitko, M. Enabling Semantics within Industry 4.0. In *International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, Springer, 2017, pp. 39–52.

[123] Pessemier, W.; Deconinck, G.; Raskin, G.; et al. Why Semantics Matter: a Demonstration on Knowledge-Based Control System Design. In *Proceedings, 15th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2015): Melbourne, Australia*, 2015.

[124] Rohjans, S.; Fensel, D.; Fensel, A. OPC UA goes semantics: Integrated communications in smart grids. In *2011 IEEE 16th Conference on Emerging Technologies & Factory Automation (ETFA)*, IEEE, 2011, pp. 1–4.

[125] Zhang, D.; Zhao, S.; Yang, L. T.; et al. NextMe: Localization using cellular traces in internet of things. *IEEE Trans. Ind. Informat.*, volume 11, no. 2, 2015: pp. 302–312.

[126] Farooq-I-Azam, M.; Ni, Q.; Ansari, E. A. Intelligent energy efficient localization using variable range beacons in industrial wireless sensor networks. *IEEE Trans. Ind. Informat.*, volume 12, no. 6, 2016: pp. 2206–2216.

[127] Elwischger, B. P. B.; Sauter, T. Efficient Ambiguity Resolution in Wireless Localization Systems. *IEEE Trans. Ind. Informat.*, 2017.

List of Author's Publications

Related to the Thesis Topic

Publication in Journal with Impact Factor

[1] V. Jirkovský (80%), M. Obitko (10%) and V. Mařík (10%). Understanding Data Heterogeneity in the Context of Cyber-Physical Systems Integration. In: *IEEE Transactions on Industrial Informatics*, 2017, 13.2: 660-667.

The paper has been cited in:

- Bellavista, Paolo, Carlo Giannelli, and Riccardo Zamagna. The PeRvasive Environment Sensing and Sharing Solution. In: *Sustainability*, 2017, 9.4: 585.
- Trunzer, E., Kirchen, I., Folmer, J., Koltun, G., and Vogel-Heuser, B. A flexible architecture for data mining from heterogeneous data sources in automated production systems. In: *IEEE International Conference on Industrial Technology (ICIT) 2017*, 2017, 1106-1111.

Publications Excerpted in ISI

[2] V. Jirkovský (90%) and R. Ichise (10%). MAPSOM: User Involvement in Ontology Matching In: *Joint International Semantic Technology Conference*, Springer, Cham, 2013. p. 348-363.

The paper has been cited in:

- Cruz, Isabel F., et al. Quality-based model for effective and robust multi-user pay-as-you-go ontology matching1. In: *Semantic Web*, 2016, 7.4: 463-479.

- ◦ Balasubramani, Booma S., Aynaz Taheri, and Isabel F. Cruz. User involvement in ontology matching using an online active learning approach. In: *OM*, 2015. p. 45-49.
- ◦ Abbes, Hanen, and Faiez Gargouri. Big data integration: A MongoDB database and modular ontologies based approach. In: *Procedia Computer Science*, 2016, 96: 446-455.
- ◦ El Alaoui, Mohamed, and Hussain Ben-Azza. Generalization of the weighted product aggregation applied to data fusion of intuitionistic fuzzy quantities. In: *2017 Intelligent Systems and Computer Vision (ISCV)*, IEEE, 2017. p. 1-6.

[3] V. Jirkovský (25%), M. Obitko (25%), P. Novák (25%) and P. Kadera (25%). Big Data analysis for sensor time-series in automation In: *19th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Barcelona, 2014, pp. 1-8.

The paper has been cited in:

- ◦ Ringsquandl, Martin, et al. Semantic-guided feature selection for industrial automation systems. In: *International Semantic Web Conference*, Springer, Cham, 2015. p. 225-240.
- ◦ Kamola, Mariusz. Analytics of industrial operational data inspired by natural language processing. In: *2015 IEEE International Congress on Big Data (BigData Congress)*, IEEE, 2015. p. 681-684.
- ◦ Moussa, Sherin. A Location-Based Framework for Mobile Blood Donation and Consumption Assessment Using Big Data Analytics. In: *Asian Journal of Information Technology*, 2016, 15.22: 4475-4481.
- ◦ Panigrahy, Sandeep, Sarang Karpate, and Saurabh Wahile. HYFRAMAL: An innovative Automation Framework for Internet of Things. In: *Proceedings of the Sixth International Conference on Computer and Communication Technology 2015*, ACM, 2015. p. 86-90.
- ◦ Ringsquandl, Martin, Steffen Lamparter, and Raffaello Lepratti. Graph-based predictions and recommendations in flexible manufacturing systems. In: *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*, 2016. p. 6937-6942.
- ◦ Balusamy, Balamurugan, et al. BALUSAMY, Balamurugan, et al. Predictive Analysis for Digital Marketing Using Big Data: Big Data for Predictive Analysis. In: *Handbook of Research on Advanced Data Mining Techniques and Applications for Business Intelligence. IGI Global*, 2017. p. 259-283.

- ◦ Matta, Purnima, and Akash Tayal. Supplier Performance Evaluation for Manufacturing Industries: Re-exploring with Big Data Analysis. In: *International Conference on Advances in Computing and Data Sciences*, Springer, Singapore, 2016. p. 516-526.
- ◦ Novák, Petr. Design and Integration of Simulation Models for Industrial Systems. *PhD Thesis* 2016, Czech Technical University in Prague.

[4] M. Obitko (25%), P. Vrba (25%), P. Kadera (25%) and V. Jirkovský (25%). Visualization of ontologies in multi-agent industrial systems In: *2011 IEEE 16th Conference on Emerging Technologies & Factory Automation (ETFA)*, IEEE, 2011. p. 1-8.

The paper has been cited in:

- ◦ Park, Seoung-Hun, and Young-Guk Ha. Visualization of resource description framework ontology using Hadoop. In: *Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, IEEE, 2013. p. 228-231.
- ◦ Park, Seonghun, Kim, Sungmin, HA, Youngguk. Scalable visualization for DBpedia ontology analysis using Hadoop. In: *Software: Practice and Experience*, 2015, 45.8: 1103-1114.
- ◦ Tanajura, Ana Paula M., et al. Distributed Manufacturing System in a Multi-Agent Approach: An Application for Oil Field Management. In: *Advances in Sustainable and Competitive Manufacturing Systems*, Springer, Heidelberg, 2013. p. 1347-1357.
- ◦ Tanajura, Ana Paula M., Valdir Leanderson C. Oliveira, and Herman Lepikson. A Multi-agent Approach for Production Management. In: *Industrial Engineering, Management Science and Applications 2015*, Springer, Berlin, Heidelberg, 2015. p. 65-75.

Other Publications

[5] V. Jirkovský (60%), Petr Kadera (20%), and Nestor Rychtyckyj (20%). Semi-automatic Ontology Matching Approach for Integration of Various Data Models in Automotive. In: *International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, Springer, Cham, 2017. p. 53-65.

[6] V. Jirkovský (70%) and M. Obitko (30%). Enabling Semantics within Industry 4.0. In: *International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, Springer, Cham, 2017. p. 39-52.

[7] M. Obitko (34%), V. Jirkovský (33%) and J. Bezdíček (33%). Big data challenges in industrial automation. In: *International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, 305-316, 2013.

The paper has been cited in:

- Zhong, Ray Y., et al. A big data approach for logistics trajectory discovery from RFID-enabled production data. In: *International Journal of Production Economics*, 2015, 165: 260-272.
- Zhang, Lichen. A framework to model big data driven complex cyber physical control systems. In: *2014 20th International Conference on Automation and Computing*, Cranfield, 2014, pp. 283-288.
- Afshari, Hamid and Peng, Qingjin. Modeling and quantifying uncertainty in the product design phase for effects of user preference changes. In: *Industrial Management & Data Systems*, 2015, 115.9: 1637-1665.
- Atya, Ahmed Osama Fathy, et al. Bolt: Realizing high throughput power line communication networks. In: *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, ACM, 2015. pp. 39.
- MOoguel, Enrique, et al. Multilayer big data architecture for remote sensing in Eolic parks. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2015, 8.10: 4714-4719.
- Afshari, Hamid and Peng, Qingjin. Using Big Data to minimize uncertainty effects in adaptable product design. In: *Proceedings of the International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2015*, Boston, MA, August 2. 2015.
- Lu, Ningyun, Bin Jiang, and Jianhua Lu. Data mining-based flatness pattern prediction for cold rolling process with varying operating condition. In: *Knowledge and information systems*, 2014, 41.2: 355-378.
- Ahmed, Firas D., et al. Agent-based big data analytics in retailing: a case study. In: *2015 4th International Conference on Software Engineering and Computer Systems (ICSECS)* IEEE, 2015. p. 67-72.
- Queiroz, Jonas, Paulo Leito, and Eugnio Oliveira. Industrial Cyber Physical Systems Supported by Distributed Advanced Data Analytics. In: *Service Orientation in Holonic and Multi-Agent Manufacturing*, Springer, Cham, 2017. p. 47-59.
- Atya, Ahmed Osama Fathy Mahmoud. Performance and security problems in today's networks. *PhD Thesis* 2016, University of California, Riverside.

- Queiroz, Jonas, and Paulo Leito. Agent-Based Data Analysis Towards the Dynamic Adaptation of Industrial Automation Processes. In: *Doctoral Conference on Computing, Electrical and Industrial Systems*, Springer, Cham, 2016. p. 99-106.
- Novák, Petr. Design and Integration of Simulation Models for Industrial Systems. *PhD Thesis* 2016, Czech Technical University in Prague.

[8] M. Obitko (50%) and V. Jirkovský (50%). Big data semantics in industry 4.0 In: *International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, Springer, Cham, 2015. p. 217-229.

The paper has been cited in:

- Grangel-Gonzlez, Irln, et al. Towards a semantic administrative shell for industry 4.0 components. In: *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)*, IEEE, 2016. p. 230-237.
- Lu, Yang. Industry 4.0: A Survey on Technologies, Applications and Open Research Issues. In: *Journal of Industrial Information Integration*, 2017.
- Domingo Galindo, Laura. The Challenges of Logistics 4.0 for the Supply Chain Management and the Information Technology. *MS thesis*, NTNU, 2016.
- Grangel-Gonzlez, Irln, et al.: An RDF-based approach for implementing industry 4.0 components with Administration Shells. In: *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2016. p. 1-8.
- Yi, Bing, Xiongbing Li, and Yue Yang. Heterogeneous model integration of complex mechanical parts based on semantic feature fusion. In: *Engineering with Computers*, 2016, 1-9.

[9] V. Jirkovský (50%) and M. Obitko (50%). Semantic Heterogeneity Reduction for Big Data in Industrial Automation In: *14th conference on Information Technologies - Applications and Theory*, 2014.

The paper has been cited in:

- Kamola, Mariusz. Analytics of industrial operational data inspired by natural language processing. In: *2015 IEEE International Congress on Big Data (BigData Congress)*
- Abbes, Hanen, and Faiez Gargouri. Big data integration: A MongoDB database and modular ontologies based approach. In: *Procedia Computer Science*, 2016, 96: 446-455.

- ◦ Moravec, Jakub. Transformace uivatelskch dotaz pro analzu dat v prmyslov automatizaci. *Bachelor's Thesis*, 2016, Czech Technical University in Prague.
- ◦ Wang, Lidong, and Randy Jones. Big Data Analytics for Disparate Data. In: *American Journal of Intelligent Systems*, 2017, 7.2: 39-46.

[10] V. Jirkovský (60%) and M. Obitko (40%). Ontology Mapping Approach for Fault Classification in Multi-Agent Systems In: *7th IFAC Conference on Manufacturing Modelling, Management, and Control*, IFAC, 2013. p. 951-956.

[11] V. Jirkovský (50%), M. Possolt (25%) and M. Obitko (25%). RDF Storage for Semantic Big Data Historian In: *Proceedings of WIKT & DaZ*, STU, 2016. p. 179-182.

Other Publications Unrelated to the Thesis Topic

[12] V. Jirkovský (25%), P. Kadera (25%), M. Obitko (25%) and P. Vrba (25%). Diagnostics of distributed intelligent control systems: Reasoning using ontologies and hidden markov models In: *IFAC Proceedings Volumes*, Elsevier, 2012. p. 1315-1320.

[13] W. Lepuschitz (25%), V. Jirkovský (25%), P. Kadera (25%) and P. Vrba (25%). A Multi-Layer Approach for Failure Detection in a Manufacturing System Based on Automation Agents In: *2012 Ninth International Conference on Information Technology: New Generations (ITNG)*, IEEE, 2012. p. 1-6.

[14] P. Vrba (20%), P. Kadera (20%), V. Jirkovský (20%), M. Obitko (20%) and V. Mařík (20%). New trends of visualization in smart production control systems In: *Holonic and Multi-Agent Systems for Manufacturing*, Springer, Cham, 2011. p. 72-83.

[15] P. Novák (20%), P. Kadera (20%), V. Jirkovský (20%), P. Vrba (20%) and S. Biffl (20%). Engineering of Coupled Simulation Models for Mechatronic Systems In: *Service Orientation in Holonic and Multi-agent Manufacturing*, Springer International Publishing, 2015. p. 3-11.

[16] V. Jirkovský (34%), P. Kadera (33%) and P. Vrba (33%). Semantics for self-configurable distributed diagnostics In: *2012 IEEE 17th Conference on Emerging Technologies & Factory Automation (ETFA)*, IEEE, 2012. p. 1-6.

[17] P. Kadera (34%), P. Vrba (33%) and V. Jirkovský (33%). Deviation Detection in Distributed Control Systems by Means of Statistical Methods In: *Proceedings of the 38th Annual Conference on IEEE Industrial Electronics Society*, Qubec: cole Polytechnique, 2012, pp. 4709-4714.

[18] P. Kadera (70%), P. Novák (20%), V. Jirkovský (5%) and P. Vrba (5%). Performance models preventing multi-agent systems from overloading computational resources In: *Automation, Control and Intelligent Systems*, Volume 6, pp. 96-102, (2014).

Patents of the Author

[19] Jirkovský, V. (20%), Obitko, M. (20%), Mavrov, R. (20%), Cherpakov, A. B. (20%), Barrancotta, A. C. (20%) (2015), P&ID and control system synchronization, U.S. Patent No. 20,160,161,930 Washington, DC: U.S. Patent and Trademark Office.

- European Patent Application EP3029535.

[20] Obitko, M. (25%), Vrba, P. (25%), Kadera, P. (25%), Jirkovsky, V. (25%) (2013), System and method for implementing a user interface to a multi-agent distributed control system, U.S. Patent No. 20,130,055,115. Washington, DC: U.S. Patent and Trademark Office.