Swansea University
Prifysgol Abertawe

Knowledge-driven Artificial Intelligence in Steelmaking: Towards Industry 4.0

Sadeer Beden

Submitted to Swansea University in fulfilment
of the requirements for the Degree of Doctorate of Philosophy

Department of Computer Science
Swansea University

December 2023

Declaration

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed: .....

Date: 01/12/23 .....

Statement 1

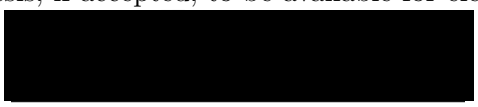
This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed: .....

Date: 01/12/23 .....

Statement 2

I hereby give consent for my thesis, if accepted, to be available for electronic sharing.



Signed: .....

Date: 01/12/23 .....

Statement 3

The University's ethical procedures have been followed and, where appropriate, that ethical approval has been granted.

Signed: .....

Date: 01/12/23 .....

Acknowledgements

Inspiration means being in the spirit of something, and often, it is our role models that ignite this spirit within us. I have a great deal of respect and appreciation for the following people who have not only inspired me to produce my best work but have influenced higher levels of personal and professional growth—from small habits like the choice of language or clothing—to more meaningful aspects like life ambitions and sense of purpose.

First and foremost, I would like to thank Arnold Beckmann, my primary supervisor. His passion for research and teaching has resonated with me, even when receiving emails from him at midnight on weekends. His work ethics and unwavering dedication leaves me greatly inspired. I am grateful to have such an engaging and pleasant supervisor. I would also like to extend my gratitude to Cinzia Gianetti, who took the role of second supervisor during the early days of the project when COVID hit. Her support and guidance were invaluable during a challenging period of the research; alongside the assistance provided by my ex-colleague and friend Qiushi Cao. A special thank you to the folks over at Tata Steel, especially Steve Thornton as the primary point of contact, and Mark Hamman, the domain expert who described the steelmaking processes so vividly that I could dream about them.

Returning to the trajectory of role models, from an academic standpoint, I wish to acknowledge and extend my gratitude to Markus Roggenbach for his mentorship. His cheerful outlook on life is something I'd like to say I adopted from him. His encouragement influenced me to apply for the Ph.D. in the first place, so this document would not exist if not for him. Additionally, I would like to give heartfelt thanks to Bertie Muller and John Tucker, two figures who continually inspire me.

Outside University, I have two families who I wish to thank. Firstly, my beautiful mother, father, and older siblings, Atheer and Maisam, who left Wales for London. They have been very loving and supportive throughout my academic journey; I feel like I've known them my whole life. Also, a big shout out to my oldest brother, Hadeer, who stayed in Wales and is probably the smartest one in the family (it was all downhill after the first child, really). Secondly, I want to thank my Swansea family, a group of people who have made my Ph.D. experience exponentially, astronomically (I'm running out of big words), more agreeable, and are the reason why this journey has been so memorable. I extend my sincerest gratitude to my two housemates, Nigel and Jeremy, whose combined age is, *opens calculator app*, 140. It is a rare and beautiful honour to have such great inter-generational friends, whom fill my life with wisdom and joy.

I would also like to thank my peers and friends in the department. There are far too many of you, so I'm going to name the ones who I think are going to be the most successful... In no particular order: Kévin Spinicci (I always have to Google the special character), Harry Bryant, Zuha (I still don't know your surname), Kira Pugh, Sam Oliver, Sam Gue, Jamie Duell, Xavier Crean, Harry Bevan, Tonicha Crook, Alec Critten, Connor Blake Hurley (give one of your surnames to Zuha), Filippos Pantekis, and Hassan Eshkiki. Much love, success, and happiness to you all. I would like to thank Zuha once more for taking the time to proofread this thesis.

Finally, a big thank you to my secret family: Peepwood, Honk, Cpt. Humperdink "*Peg-leg*" Plankton III, JD, Saint *"Felipé Flop"* Brendan, Ronald Ducksworth, and Donald Pip-squeeze.

Abstract

With the ongoing emergence of the Fourth Industrial Revolution, often referred to as Industry 4.0, new innovations, concepts, and standards are reshaping manufacturing processes and production, leading to intelligent cyber-physical systems and smart factories. Steel production is one important manufacturing process that is undergoing this digital transformation. Realising this vision in steel production comes with unique challenges, including the seamless interoperability between diverse and complex systems, the uniformity of heterogeneous data, and a need for standardised human-to-machine and machine-to-machine communication protocols.

To address these challenges, international standards have been developed, and new technologies have been introduced and studied in both industry and academia. However, due to the vast quantity, scale, and heterogeneous nature of industrial data and systems, achieving interoperability among components within the context of Industry 4.0 remains a challenge, requiring the need for formal knowledge representation capabilities to enhance the understanding of data and information. In response, semantic-based technologies have been proposed as a method to capture knowledge from data and resolve incompatibility conflicts within Industry 4.0 scenarios.

We propose utilising fundamental Semantic Web concepts, such as ontologies and knowledge graphs, specifically to enhance semantic interoperability, improve data integration, and standardise data across heterogeneous systems within the context of steelmaking. Additionally, we investigate ongoing trends that involve the integration of Machine Learning (ML) techniques with semantic technologies, resulting in the creation of hybrid models. These models capitalise on the strengths derived from the intersection of these two AI approaches. Furthermore, we explore the need for continuous reasoning over data streams, presenting preliminary research that combines ML and semantic technologies in the context of data streams.

In this thesis, we make four main contributions: (1) We discover that a clear understanding of semantic-based asset administration shells, an international standard within the RAMI 4.0 model, was lacking, and provide an extensive survey on semantic-based implementations of asset administration shells. We focus on literature that utilises semantic technologies to enhance the representation, integration, and exchange of information in an industrial setting. (2) The creation of an ontology, a semantic knowledge base, which specifically captures the cold rolling processes in steelmaking. We demonstrate use cases that leverage these semantic methodologies with real-world industrial data for data access, data integration, data querying, and condition-based maintenance purposes. (3) A framework demonstrating one approach for integrating machine learning models with semantic technologies to aid decision-making in the domain of steelmaking. We showcase a novel approach of applying random forest classification using rule-based reasoning, incorporating both meta-data and external domain expert knowledge into the model, resulting in improved knowledge-guided assistance for the human-in-the-loop during steelmaking processes. (4) The groundwork for a continuous data stream reasoning framework, where both

domain expert knowledge and random forest classification can be dynamically applied to data streams on the fly. This approach opens up possibilities for real-time condition-based monitoring and real-time decision support for predictive maintenance applications. We demonstrate the adaptability of the framework in the context of dynamic steel production processes. Our contributions have been validated on both real-world data sets with peer-reviewed conferences and journals, as well as through collaboration with domain experts from our industrial partners at Tata Steel.

Keywords: Industry 4.0, RAMI 4.0, Semantic Web, Ontology, Machine Learning, Rule-based Reasoning, Continuous Streaming, Continuous Reasoning, Predictive Analytics, Semantic Interoperability.

Table of Contents

List of Figures

List of Tables

Part I

Introduction and Background

CHAPTER 1

Introduction

Contents

This chapter introduces the main themes and structure of the thesis, as well as the context and motivations behind the research. The research questions and contributions are presented, briefly introducing the peer-reviewed journal and international conference publications in later sections.

1.1 Introduction

Over the years, there have been two distinct approaches to Artificial Intelligence (AI). One approach, known as logical 'Symbolic' AI, focuses on representing knowledge symbolically for programs to reason on. An example of symbolic AI is Knowledge Representation and Reasoning (KRR), which utilises Semantic Technologies for these tasks. These technologies, introduced by the World Wide Web Consortium (W3C) in 2001 [33], transform the traditional web—originally designed for humans to store and share documents—into a platform where explicit annotations are embedded into documents, providing semantic meaning to the information. This concept, pioneered by Tim Berners-Lee, makes the content on the web interconnected and machine-readable. This form of AI is commonly referred to as knowledge-driven AI, where the core techniques involve the construction of a knowledge base, logical rules, and reasoning mechanisms.

On the other hand, statistical Machine Learning (ML) models focus primarily on algorithms that enable computers to learn concepts without explicit definitions [8]. ML is often referred to as data-driven AI, where patterns in data are exploited to produce an output from a set of input variables, often excluding the context of the data. Both of these AI approaches have an extensive set of use cases and applications across a wide range of industries, including the domain of manufacturing.

The modern era of manufacturing, often termed the fourth industrial revolution or Industry 4.0, is undergoing digital transformation, leveraging AI, among other digitisation trends such as the Internet of Things (IoT) and Cloud Computing (CC) [112]. In this vision, physical components are digitally interconnected and contain the capabilities to comprehend data, share information, and make actionable decisions, autonomously. This level of digitisation is propelling the manufacturing industry as one of the main contributors of data creation and is anticipated to grow exponentially with the ongoing transformation towards Industry 4.0. Figure 1.1 displays different domains and their quantity of data.

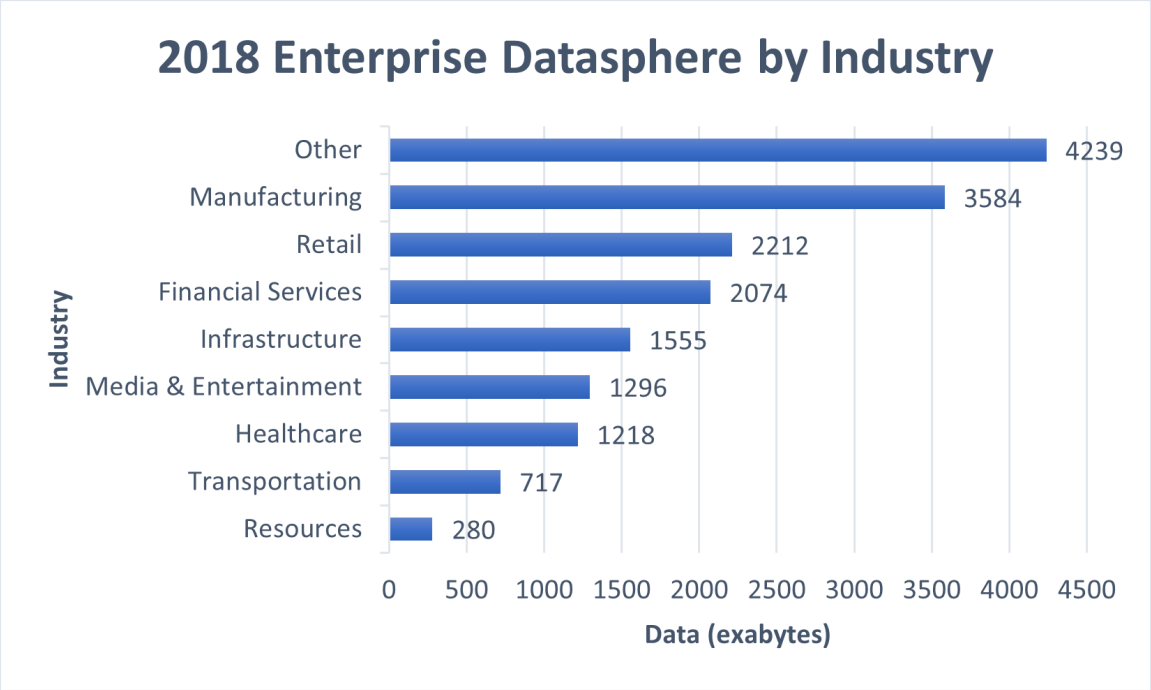


Figure 1.1: The 2018 Enterprise Datasphere statistics by the IDC [269].

To provide some statistics, the International Data Corporation (IDC), a global provider of information technology, estimated that the Global Datasphere, referring to the total accumulated data in the "digital universe", would surge from 33 zettabytes in 2018 to 175 zettabytes by 2025 [269]. Visualising the scale of available data, 33 zettabytes equates to 33 trillion gigabytes, estimating approximately 4,125 gigabytes for each person on the planet. To further grasp the anticipated growth of 175 zettabytes by 2025, the IDC noted

that it would take someone approximately 1.8 billion years to download the entire global datasphere at an average download speed of 25 megabytes per second [269].

Steel production is one important manufacturing process that is undergoing this digital transformation. On one side, steelmaking involves industrial-size legacy components that generate vast amounts of dynamic and static data [271]. On the other hand, steelmaking processes also require an extensive amount of human knowledge to operate which needs to find its way into the digitisation processes.

These digital trends have enabled a shift toward smart factories, resulting in the production of Cyber-Physical Systems (CPS) [184]. While new manufacturers can invest in Industry 4.0-compliant machinery and CPS-ready equipment, replacing large-scale legacy systems in manufacturing enterprises remains a significant barrier. Therefore, an integral aspect of Industry 4.0 is upgrading the legacy systems without the need to replace physical components [304].

To reach this goal, system integration across three distinct dimensions of manufacturing has to be addressed [319], this includes:

- *Horizontal integration across the entire value creation network:* Horizontal integration involves the integration of data obtained through collaboration between individuals and their systems such as operators on the shop floor, suppliers, stakeholders, customers, and many others.
- *Vertical integration and networked manufacturing systems:* Vertical integration involves the integration of systems across various hierarchical levels within a factory, resulting in a large-scale unified network. The primary goal of this integration is to make data readily available across all levels of the network, e.g., data collected by sensors on the shop floor can be directly accessed by members operating at the enterprise level.
- *End-to-end integration across the entire product life-cycle:* Achieving end-to-end integration of the entire life cycle of a product is a necessary requirement within Industry 4.0. This involves capturing the historical and operational data of an asset, as well as its purpose. For example, a Programmable Logic Controller (PLC) would encompass its conception, design, production, utilisation, and termination phases, not simply the data collected by the PLC.

Attaining this level of integration is a complex task due to the diversity of languages and formats used by various systems, organisations, and individuals for communication. The data collected originates from a multitude of sources, resulting in a lack of uniformity. To address these challenges, data in its heterogeneity state must be uniformly represented and standardised, and any conflicts between representations must be resolved. This implies that entities, such as sensors, assets, machinery, etc., must be semantically described in a format that is comprehensible by both machines and humans, essentially conveying that machines should not only possess the capability to read and share information but also comprehend the underlying concepts in the data.

Thus, to address the Industry 4.0 requirements, reference architecture models have been developed on a global scale, each introducing new standards and methodologies. This includes: (1) the *Reference Architecture Model Industry 4.0* model (RAMI 4.0) in Europe [130], (2) the *Industrial Internet Reference Architecture* (IIRA) initiative from the US [189], (3) the *Industrial Value Chain Reference Architecture* (IVRA) led in Japan [226], and (4) the *China Communications Standards Association* (CCSA) and *China Intelligent Manufacturing System Architecture* (IMSA) led in China [66, 329].

RAMI 4.0 is one example of a reference architecture model, which introduces new concepts to overcome ongoing challenges. One example is the concept of an Asset Administration Shell (AAS), a fundamental concept within Industry 4.0 that digitises physical components within a manufacturing environment. When an AAS is installed onto a traditional asset, the asset is considered an Industry 4.0 component, offering enhanced capabilities within a larger, semantically interconnected network [305]. The requirements of AAS are segmented into layers. Each layer states the necessary changes and proposes international standards and technologies for each requirement. In total there are six layers; we primarily focus on the information and communication layers in this thesis.

The information layer focuses on the encapsulation and representation of industrial data, while the communication layer centers on enabling ubiquitous and uniform communication among physical devices. RAMI 4.0 adopts international standards to achieve the specifications of each layer, utilising OPC Unified Architecture (OPC UA) among other frameworks to address the communication tasks, and Automation Markup Language (AML) standards to cover the main foundation of the information layer. However, upon reviewing relevant literature, it is evident that although these standards excel at enabling data access and control, they lack complete semantic interoperability between physical components [281]. For instance, the semantics layer provided by OPC UA is defined in an implicit format inside specification documents, which rely on human comprehension and modification.

To achieve autonomous communication between components, enhanced semantic interoperability is required, prompting the need to explore methods that can capture data semantics to overcome these limitations. Hence, we propose knowledge-driven AI methods that focus on leveraging semantics to model and enrich knowledge within the domain of Industry 4.0. These technologies show promise in addressing Industry 4.0 challenges [237], and provide many advantages, including: (1) the ability to provide a shared, machine-understandable vocabulary for data integration and exchange between components [9], (2) enabling the capability to access and query data at a virtual level without physical data integration [335], and (3) simulating cognitive decision making tasks through logical inferences, rules, and reasoning [327].

In this thesis, we explore the use of semantic technologies, such as ontologies and knowledge graphs, in the domain of steelmaking. In addition, we investigate the use of various traditional machine learning models in the same context. These ML methods have demonstrated widespread success in diverse industrial domains, but face limitations in the context of Industry 4.0 [74], mainly due to the lack of context-aware information in dynamic production environments and limited incorporation of semantics, where meta-data is often

excluded from the model [338]. This challenge is particularly evident in steel production, where processes are heavily reliant on human knowledge. Thus, we investigate the development of hybrid models that combine semantic technologies and ML techniques. We propose new methods that leverage the intersection of the two AI methods, demonstrating how these technologies can be combined for predictive analytics and predictive maintenance purposes within the scope of Industry 4.0.

Finally, with the ongoing digitisation of steel production generating a vast amount of dynamic and static data, we explore possible methodologies for simulating real-world data using data streams. We utilise semantic technologies to continuously reason on the data on the fly, applying ML techniques and domain expert knowledge for the application of predictive maintenance.

1.2 Context of the Thesis and Motivation

This research is an Industrial Case (ICASE) project, which focuses on real-world industrial scenarios. Our research is conducted in collaboration with Tata Steel, a large-scale enterprise that is currently undergoing digital transformation. Tata Steel is utilising new industrial standards to convert traditional legacy systems into intelligent systems, aligning with the vision of Industry 4.0.

Looking at the broader context of Industry 4.0, one important aspect of this transition involves achieving semantic data integration between components. The integration goes beyond simple and traditional exchange of data among hardware components, requiring the ability for machines on the shop floor to comprehend the context behind the data. This enables regular components to make actionable decisions autonomously, i.e., without human intervention.

To address these challenges, especially in data modelling and exchange, semantic technologies have been proposed and have shown promising results for semantic interoperability [339]. These technologies offer methods for capturing data in a format interpretable by both humans and machines.

In this thesis, we address the ongoing challenges of Industry 4.0 and utilise semantic technologies to bridge the gap in data modelling and exchange. We focus on one particular aspect of steelmaking: the cold rolling of steel, with a specific emphasis on maintenance and scheduling. Our industrial partners have shown significant interest in this scope and direction.

Our research investigates many digitisation trends, specifically for the use cases within the ICASE project. Although our research is domain-specific to the ICASE project, much of the research can be generalised for other operations or enterprises. For instance, the ontology developed in Chapter 4 is constructed specifically for the maintenance of the cold rolling machines and processes at the Port Talbot steelworks but can be modified and generalised to fit other processes and enterprises.

1.3 Research Questions and Contributions

While exploring the literature on Asset Administration Shells (AAS), there were plentiful resources available. Meanwhile, when specifically investigating semantic-based AAS, the literature proved to be insufficient, and a clear understanding of the subject was lacking. Thus, more clarity on the subject was necessary, leading to our first research question: **(RQ1)** What are the foundational standards and ongoing challenges in Industry 4.0 that relate to semantics? To answer this question, we have compiled a survey paper on semantic-based AAS, examining the standards and ongoing challenges in Industry 4.0 relating to data semantics.

Additionally, large enterprises with physical sites such as Tata Steel, generate a vast amount of data, which are often segmented into different data silos. Different departments from the same business store data in different locations. To extract meaningful insights from the data, data integration becomes a prerequisite. Generally, this process involves the collaboration between external integration experts and internal system architects who are capable of explaining the data schema. While these data silos may not be directly linked, the data they store may contain semantic connections. For example, two data silos may share database tables containing similar columns serving as foreign keys, such as `Roll_ID`. Hence, our second research question is: **(RQ2)** How can the data be seamlessly integrated and understood with the help of semantics within smart manufacturing? To address this research question, we have developed the Steel Cold Rolling Ontology (SCRO). This approach showcases how semantic methodologies can be employed to achieve "virtual" data integration in the context of steelmaking. We demonstrate an example of how traditional data stored in an SQL database can be converted into a knowledge graph, utilising the semantic methods to query the data without the need to physically integrate the data silos.

Machine Learning (ML) has played a significant role in advancing predictive analytics and maintenance tasks. However, in the context of Industry 4.0, the presence of heterogeneous and unstructured data requires formal representations of knowledge to enhance data understanding. This consideration leads to our third research question: **(RQ3)** Is there a way to integrate ML with semantic methodologies in smart manufacturing? What are the existing AI methods available for combining these technologies? In response to this question, we have developed our own framework, which converts random forest classification into rule-based reasoning that can be applied to a knowledge graph. This approach incorporates meta-data as part of the model, which can be combined with external domain or expert knowledge to provide enhanced predictive assistance. We illustrate the utility of the framework through a use-case in cold rolling, where the framework is used for predictive analytic purposes.

Finally, considering the continuous nature of manufacturing processes and the importance of predictive analytic and maintenance tasks, we pose the following research question: **(RQ4)** Can we apply continuous reasoning on data streams, incorporating both machine learning classification and domain expert knowledge on the fly? To address this question, we have developed a framework for continuous reasoning to data streams, which involves

both machine learning classification and domain expert knowledge in real time. The proposed framework continuously simulates the generation of real-world data, and after specific timestamps, the data is collected and processed by a semantic reasoner using rule-based reasoning to obtain classifications. These classifications are generated on the fly by converting the random forest into semantic-based rules and applying rule-based reasoning to the data, without interrupting the ongoing process. These classifications are fed into the reasoner once more to apply the domain expert knowledge for a final output. We demonstrate the applicability of the framework in a use case for predictive maintenance purposes.

Table 1.1 provides a structured format that describes the contributions, highlighting the research question, focus, methodology, results, and publications of each chapter. This table extends multiple pages.

Table 1.1: An overview of our contributions.

	Chapter 3	**Chapter 4**	**Chapter 5**	**Chapter 6**
Research Questions	(RQ1) What are the standards and ongoing challenges in Industry 4.0 related to semantics?	(RQ2) How can data and semantics be integrated and understood seamlessly within smart manufacturing?	(RQ3) Is there a way to integrate ML with semantic methodologies in smart manufacturing? What are the existing AI methods available for combining these technologies?	(RQ4) Can we apply continuous reasoning to data streams, incorporating both machine learning classification and domain expert knowledge on the fly?
Focus	Understanding the foundations and definitions of the subject.	Knowledge acquisition, and using semantic methods to achieve data integration.	Experimenting with hybrid methodologies to assist with steelmaking decision-making tasks.	Experimenting with semantic tools for continuous streaming and reasoning.

Method-ology	Conduct a systematic literature review.	(1) Follow well-studied ontology development methodologies to create our own ontologies. (2) Investigate knowledge acquisition methods for interactions with domain experts.	Multi-case study and analysis of topics and tools available, including random forests and semantic languages (OWL, RDF).	A framework that combines machine learning with semantic querying and semantic reasoning.
Main Results	(1) Identified key terms and topics. (2) Found a gap in research areas.	(1) Developed the Steel Cold Rolling Ontology. (2) Demonstrates virtual data integration and querying capability of semantic technologies.	Developed a framework for combining knowledge-driven and data-driven AI for predictive analytics in steelmaking.	(1) An automated and continuous simulation of data. (2) Autonomous decision-making based on random forest and semantic methods.

Publications	Conference paper **Beden S.** & Cao Q. & Beckmann A., "Semantic Asset Administration Shells in Industry 4.0: A Survey," 2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS), Victoria, BC, Canada, 2021, pp. 31-38	Journal paper **Beden S.** & Cao, Q. & Beckmann, A. "SCRO: A Domain Ontology for Describing Steel Cold Rolling Processes towards Industry 4.0". Information 2021, 12, 304. Journal paper Qiushi C. & **Beden S.** & Beckmann A., "A core reference ontology for steelmaking process knowledge modelling and information management'', Computers in Industry, Volume 135, 2022 , 103574, ISSN 0166-3615	Conference paper **Beden S.** & Beckmann A., "Towards an Ontological Framework for Integrating Domain Expert Knowledge with Random Forest Classification" 2023 The 34th International Conference on Database and Expert Systems Applications (DEXA 2023), Laguna Hills, CA, USA, 2023, pp. 221-224 Journal paper **Beden S.** & Lakshmanan K. & Giannetti C. & Beckmann A., (2023). Steelmaking Predictive Analytics Based on Random Forest and Semantic Reasoning. Applied Sciences. 13. 12778.	

1.3.1 Publications

This section lists the publications that have been peer-reviewed and accepted since the start of the ICASE. In total, there are five publications: two international conferences and three journal publications.

1.3.1.1 Peer-reviewed International Conference Papers

- **S. Beden**, Q. Cao and A. Beckmann, "Semantic Asset Administration Shells in Industry 4.0: A Survey," 2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS), Victoria, BC, Canada, 2021, pp. 31-38, `doi:10.1109/ICPS49255.2021.9468266.`

  Abstract *The Asset Administration Shell (AAS) is a fundamental concept in the Reference Architecture Model for Industry 4.0 (RAMI 4.0), that provides a virtual and*

digital representation of all information and functions of a physical asset in a manufacturing environment. Recently, Semantic AASs have emerged that add knowledge representation formalisms to enhance the digital representation of physical assets. In this paper, we provide a comprehensive survey of the scientific contributions to Semantic AASs that model the Information and Communication Layer within RAMI 4.0, and summarise and demonstrate their structure, communication, functionalities, and use cases. We also highlight the challenges of future development of Semantic AASs.

- **S. Beden** and A. Beckmann, "Towards an Ontological Framework for Integrating Domain Expert Knowledge with Random Forest Classification" 2023 The 34th International Conference on Database and Expert Systems Applications (DEXA 2023), Laguna Hills, CA, USA, 2023, pp. 221-224, doi: 10.1109/ICSC56153.2023.00043.

  Abstract *This paper proposes an ontological framework that combines semantic-based methodologies and data-driven random forests (RF) to enable the integration of domain expert knowledge with machine-learning models. To achieve this, the RF classification process is firstly deconstructed and converted into semantic-based rules, which are combined with external rules constructed from the knowledge of domain experts. The combined rule set is applied to an ontological reasoner for inference, producing two classifications: (1) from simulating the selected RF voting strategy, (2) from the knowledge-driven rules, where the latter is prioritised. A case study in the steel manufacturing domain is presented that uses the proposed framework for real-world predictive maintenance purposes. Results are validated and compared to typical machine-learning approaches.*

1.3.1.2 Peer-reviewed Journal Papers

- **S. Beden**; Q. Cao; A. Beckmann "SCRO: A Domain Ontology for Describing Steel Cold Rolling Processes towards Industry 4.0". Information 2021, 12, 304. `https://doi.org/10.3390/info12080304`

  Abstract *This paper introduces the Steel Cold Rolling Ontology (SCRO) to model and capture domain knowledge of cold rolling processes and activities within a steel plant. A case study is set up that uses real-world cold rolling data sets to validate the performance and functionality of SCRO. This includes using the Ontop framework to deploy virtual knowledge graphs for data access, data integration, data querying, and condition-based maintenance purposes. SCRO is evaluated using OOPS!, the ontology pitfall detection system, and feedback from domain experts from Tata Steel.*

- Q. Cao, **S. Beden**., A. Beckmann., "A core reference ontology for steelmaking process knowledge modelling and information management", Computers in Industry, Volume 135, 2022, 103574, ISSN 0166-3615, `https://doi.org/10.1016/j.compind.2021.103574.`

Abstract *Following the trend of Industry 4.0, the business model of steel manufacturing is transforming from a historical inwardly focused supplier/customer relationship to one that embraces the wider end-to-end supply chain and improves productivity more holistically. However, the data and information required for supply chain planning and steelmaking process modelling are normally distributed over scattered sources across organisation boundaries and research communities. This leads to a major problem concerning semantic interoperability. To address this issue, this paper introduces a Common Reference Ontology for Steelmaking (CROS). CROS serves as a shared steelmaking resource and capability model that aims to facilitate knowledge modelling, knowledge sharing and information management. In contrast to most of the existing steelmaking ontologies which merely focus on conceptual modelling, our work pays special attention to the real-world implementation and utilisation aspects of CROS. The functionality and usefulness of CROS are evaluated and tested on a real-world condition-based monitoring and maintenance task for cold rolling mills at Tata Steel in the United Kingdom.*

- **S. Beden** & K. Lakshmanan & C. Giannetti & A. Beckmann (2023). Steelmaking Predictive Analytics Based on Random Forest and Semantic Reasoning. Applied Sciences. 13. 12778. 10.3390/app132312778.

  Abstract *This paper proposes a human-in-the-loop framework that integrates machine learning models with semantic technologies to aid decision-making in the domain of steelmaking. To achieve this, we convert a random forest (RF) into rules in a Semantic Web Rule Language (SWRL) format and represent real-world data as a knowledge graph in a Resource Description Framework (RDF) format, capturing the meta-data as part of the model. A rule engine is deployed that applies logical inference on the knowledge graph, resulting in a semantically enriched classification. This new classification is combined with external domain expert knowledge to provide improved, knowledge-guided assistance for the human-in-the-loop system. A case study in the steel manufacturing domain is introduced, where this application is used for real-world predictive analytic purposes.*

1.4 Structure of the Thesis

This section displays the structure of the thesis, which is partitioned into four parts:

- **Part I: Introduction and Background** containing two chapters:

  1. **Chapter 1: Introduction** Introduces the context of the thesis, research questions, and contributions.
  2. **Chapter 2: Background** Gives the background knowledge of steelmaking with a focus on cold rolling processes, Industry 4.0 with a focus on the Reference Architecture Model for Industry 4.0 (RAMI 4.0), semantic technologies, and machine learning.

- **Part II: Contributions** containing four chapters:

    1. **Chapter 3: Semantic-Web Asset Administration Shells Survey** This chapter presents a comprehensible literature review that employs semantic technologies for semantics modelling data and data exchange for the creation of asset administration shells. In the review, we focus on literature that focuses on or provides ontological implementations.
    2. **Chapter 4: An Ontology for Steel Cold Rolling** This chapter introduces an ontological framework we have developed that specifically captures and models domain knowledge of the cold rolling processes. A case study for data access, data integration, data querying, and condition-based maintenance purposes is demonstrated.
    3. **Chapter 5: Steelmaking Predictive Analytics Based on Random Forest and Semantic Reasoning** This chapter introduces a novel framework that combines machine learning classification with semantic web technologies for predictive analytic tasks. This framework demonstrates how random forest classification can be converted into semantic-based rules and recreated using rule-based reasoning, providing benefits such as the integration of domain expert rules for enhanced classifications. A case study is presented that assists plant operators with critical decision-making tasks.
    4. **Chapter 6: Towards a Framework for Continuous Streaming with Neruosymbolic AI** This chapter introduces a software prototype that simulates real-world data streams for predictive maintenance tasks. This framework generates data in Resource Description Language (RDF) format, a semantic-based language, which is continuously reasoned on, applying random forest classification and domain expert rules to the data stream in order to predict machine failure in the scenario of cold rolling.

- **Part III: Conclusions** this section contains one chapter:

    1. **Chapter 7: Conclusions, Reflections, and Future Work** This chapter concludes the thesis, giving a high-level summary of what has been achieved. We reflect on the research questions once more and describe possible future directions.

- **Part IV: Appendices** This final section contains Appendix A, which includes some historical articles, and Appendix B, which contains implementation code.

CHAPTER 2

Background

Contents

In this chapter, we introduce the main topics of steel manufacturing, Industry 4.0, semantic technologies, and touch on machine learning and neurosymbolic AI.

2.1 Brief Introduction and History of Steelmaking

Manufacturing is the process of transforming raw materials into finished products for consumer use [92]. Steelmaking is one example of a well-established manufacturing process. Steelmaking is the process of producing a robust and versatile alloy known as steel, which is primarily composed of iron and carbon among other chemicals. The production of steelmaking involves many physicochemical and refining processes such as melting, separation, chemical reaction, rolling, solidification, and many others [201]. Steel material can be produced with a variety of mechanical properties including strength, toughness, ductility, thickness, etc., to meet the specifications of a variety of products.

Numerous inventions and instances of innovations have significantly improved the rate of production, as well as the quality of steel. Figure 2.1 presents a timeline that highlights

some key advancements that brought fruition to steelmaking. This thesis covers several of these developments, including an example of how Wales has left a global impact in this domain.

These innovations have spanned the globe, expanding over an extended period, from the Middle Ages with the inventions of bloomery and forging, to modern-day practices like mass-automated steelmaking [113]. Bloomery involves heating rocks containing iron with charcoal in a bloomery—an old-fashioned furnace—resulting in chunks of impure iron, known as *iron bloom* [280]. These blooms are then hammered into the desired shape. Between 1983 and 2007, the Snowdonia National Park Study Centre at Plas Tan y Bwlch conducted nearly 100 experiments that explored the prospects of a bloomery, detailed in [222].

A pivotal advancement in the iron and steel industry was the invention of the early *Heat Furnace* or *Blast Furnace*, a tall, vertical shaft furnace [239]. This furnace utilised carbon, typically in the form of coke, to reduce iron from its original oxide ore form, resulting in a liquid known as *Pig iron* [239].

An innovation of the heating furnace was the *Hot Blast Furnace*, patented by James Beaumont Neilson in 1828 [29]. This variant marked a significant improvement over the traditional blast furnace. Neilson's innovation involved equipping a blast furnace with a hot blast stove, which preheated the air before placing it into the furnace. This process reduced fuel consumption and lowered the overall processing time. Initially, the furnace primarily used charcoal and bituminous coal as fuel for smelting iron. Bituminous coal is a soft and more readily available coal compared to burning wood for charcoal, but sometimes the coal contains impurities that affect the quality of the iron produced [109].

Around ten years later, Welsh inventor David Thomas made notable implementations and improvements to this process. He adapted the hot blast furnace to use anthracite coal, a harder, cleaner-burning coal [340]. Anthracite coal was abundant in mountainous regions [63], which was ideal for Thomas who worked in Ynyscedwyn near Swansea. It was in this context that Thomas and his colleague George Crane built the first anthracite-smelted hot blast furnace on February 7$^{\text{th}}$, 1837 [29]. Thomas soon after moved to Pennsylvania in 1840, where he took these inventions and catalysed the progress of ironmaking in America. By 1853, there were 121 anthracite blast furnaces in eastern Pennsylvania [63], earning him the title of *Father of the anthracite iron business*, which was presented on his obituary, published in *The Philadelphia Inquirer newsletter* on Wed, 21$^{\text{st}}$ June 1882 [306].

Looking at further advancements, one of the most notable innovations of steelmaking was the Bessemer Process, first introduced by Sir Henry Bessemer in his paper titled *Manufacture of Malleable Iron and Steel without Fuel*, which he presented to the British Association in Cheltenham on Wednesday, 13$^{\text{th}}$ August 1856 [103]. The Bessemer process, also referred to as the converter process, utilised a reactor to blow air through molten iron, effectively removing impurities from the material, and resulting in higher-quality steel. Notably, this was the first process to produce steel as a one-step process, eliminating the need to preheat the air before smelting [140]. Figure 2.2 displays the principal structure of the Bessemer converter on the left and a picture of Bessemer's steelworks in Sheffield in 1859

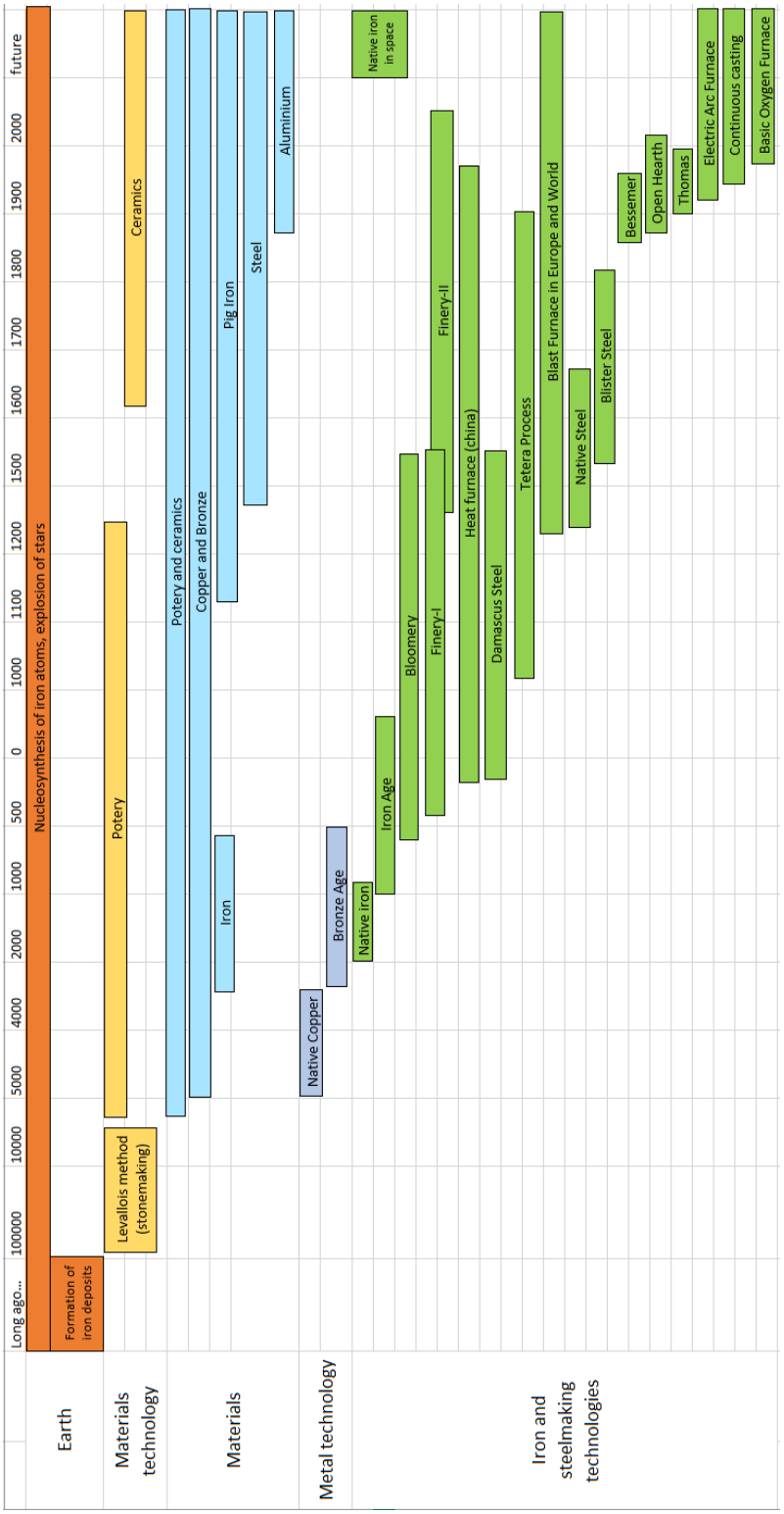


Figure 2.1: A timeline of materials and steelmaking technologies based on [37].

from his autobiography [35].

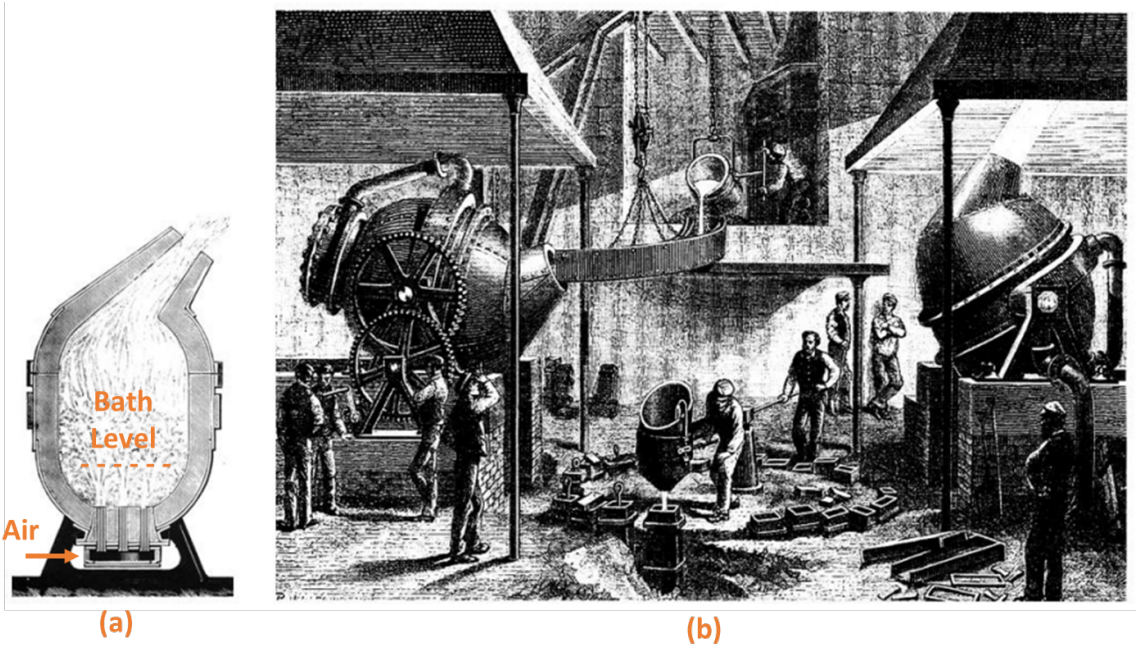


Figure 2.2: Two figures, where: (a) is the principal of the Bessemer converter, adapted from [140], and (b) an image of the Bessemer Steelworks at Sheffield after 1859 [35].

Subsequently, Bessemer published a more detailed description and formalisation of the process in his 1859 paper [34], highlighting its importance and significance, contributing to its widespread adoption. This method was inexpensive, which significantly reduced the costs of steel production, facilitating mass production of steel, and impacting substantial growth of the industries that heavily relied on steel for various applications.

Continuing the trajectory of steelmaking advancements, the 1860s saw the introduction of the Siemens-Martin process [294], also known as the Open Hearth Process. This method entailed melting a mixture of pig iron and scrap steel in a reverberatory furnace using air and fuel gas. This method utilised iron ore to oxidise carbon in liquid pig iron [98]. This process enabled more control of the chemical composition of the steel, offering more flexibility in terms of raw materials used. This method of steelmaking remained prominent for several decades, before being largely replaced by modern practices and technologies, including the electric arc furnace and basic oxygen furnace.

The Electric Arc Furnace (EAF), displayed in Figure 2.3, introduced near the end of the 19^{th} century, significantly advanced steelmaking by leveraging electricity and the melting of recyclable scrap [28]. The EAF utilised electrical power to achieve the necessary temperatures for melting, providing a feature that made attaining high-quality steel more accessible [310]. Standard EAF operations consist of a sequence of four sub-processes [229]: (1) charging the furnace using one to three scrap buckets for heating, (2) melting, where

the arc is ignited using electrical power- melting the scrap steel using electrical discharge between electrodes and utilising coke, lime, and dolomite to preserve slag formation [179], (3) superheating, wherein, after the scrap has been melted, the desired temperature is achieved, and alloying elements or additional scrap can be added to attain the desired steel composition, and (4) tapping, where, once the temperature is ideal, the furnace tilts into a tapping position, allowing molten steel to be tapped and cast into the desired form. This method added flexibility in terms of the raw material used, as well as facilitating the recycling of scrap metal. Moreover, achieving the desired composition of the steel became more easily manageable, expanding the production of wider ranges of steel grades.

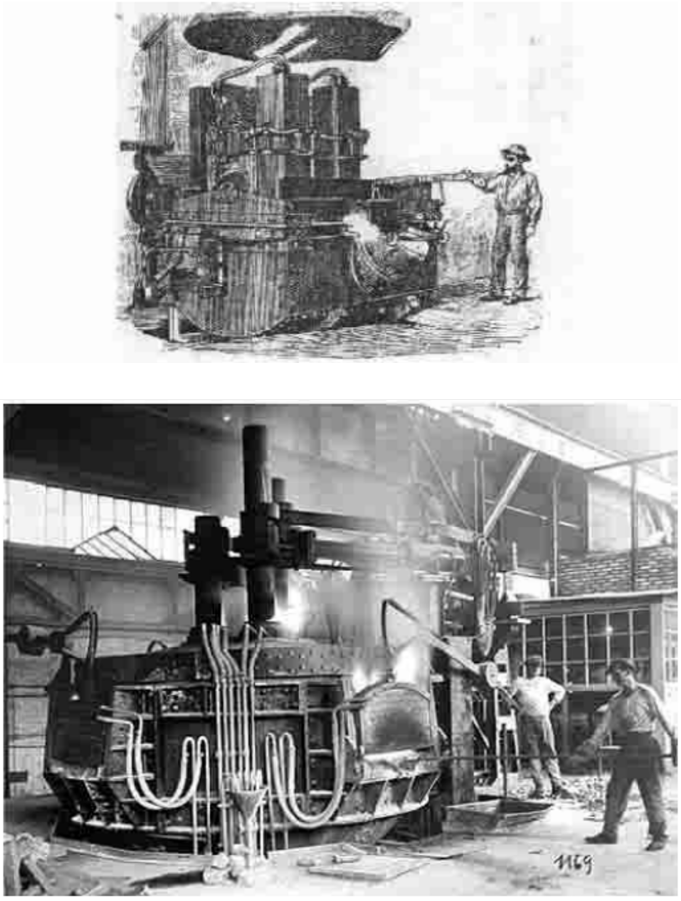


Figure 2.3: Two diagrams: (Top) One of the first EAF by Héroult. (Bottom) An industrial EAF in Dommeldange, Luxembourg [37].

The contributions and developments of the EAF are attributed to a few pioneers. First, Sir Humphry Davy, regarded as the father of dry electro-metallurgy [84], discovered the carbon arc and demonstrated an experiment in 1810 that used an electric arc to melt iron. Although he did not use it for steelmaking directly, he proved that such methodologies

were possible. One of the first real attempts to develop an electrothermic furnace was by M. Pinchon, patented on March 16th 1853 [84]. Pinchon experimented with electric arc welding, producing a process that differed from the practices mentioned above by mixing iron and other ores with carbon, which were shaken between two or more pairs of electrodes, melting and dropping the material into an externally heated receptacle below. Moreover, the first electric arc furnace for commercial steelmaking use was designed by the French pioneer, Paul Héroult around 1890 [215], enabling the industrial scale application of EAF for steelmaking [82]. The first production was in Syracuse, New York, in 1906 [94].

Finally, post-World War II witnessed the development of the Basic Oxygen Furnace (BOF) [209], leading to the replacement of many open hearth furnaces in the mid-20th century [231]. This process, often referred to as the oxygen converter process, operates by injecting oxygen into molten iron to obtain the desired steel composition [231]. The key distinction of the BOF, compared to EAF or open hearth process, is that no external heating is required. Instead, the molten iron and scrap metal are charged into a furnace, employing an oxygen lance to oxidise the iron and reduce its carbon content [83]. Figure 2.4 displays the schematic representation of BOF, highlighting the key inputs and outputs of the processes. The standard processes involve several steps [83]: (1) charging the blast furnace with molten iron from iron ore, (2) injecting an oxygen jet at high speed through a lance onto the surface of the molten iron bath, (3) refining by supplementing alloying elements to attain the desired steel grades, and (4) tapping, wherein the molten steel is cast into its desired form.

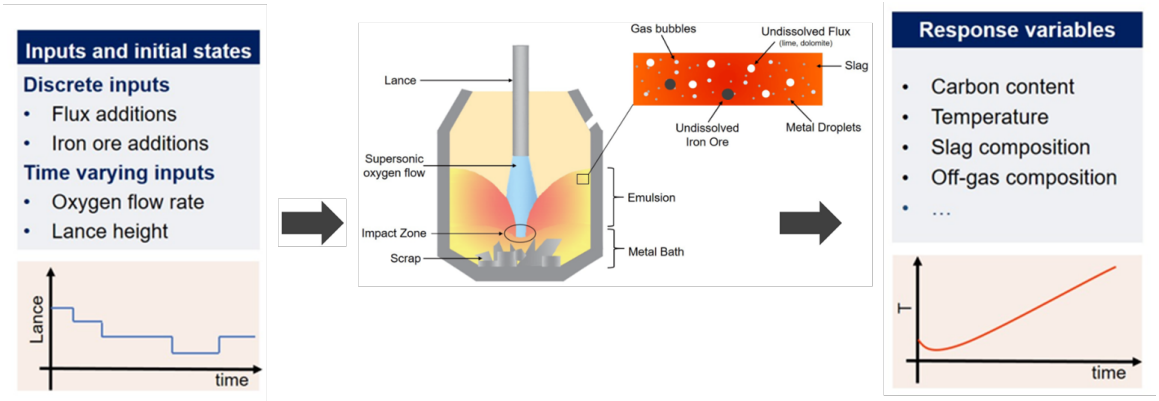


Figure 2.4: A schematic representation of the Basic Oxygen Furnace (BOF) [83].

BOF provides efficiency in the process of producing large quantities of steel at a relatively low cost. During the surge of its adoption, BOF began to replace many open hearth furnaces. For instance, in 1961, the capital cost of opening a new, optimal open hearth furnace was estimated at $35 per ton, whereas the capital cost of BOF was estimated $17 per ton [231]. Additionally, tapping time, i.e., the duration for extracting the molten steel from the surface, was also significantly reduced from eight hours in an open hearth to 45 minutes in the BOF. Figure 2.5 displays the rise and fall of different steel-making processes between

1954 and 1974.

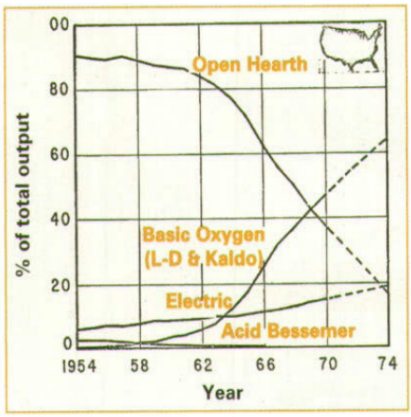


Figure 2.5: A graph displaying the rise and fall of the newer steelmaking processes between 1954-1974, from [302].

Furthermore, BOF methods offer precise control over the chemical composition of the steel produced, enabling the wide production range of steel grades with distinct properties for various applications. The method has matured in terms of maximising productivity, stability, and safety [274]. However, one limitation is that the process is limited in automation, relying heavily on the operational knowledge of the operators. Additionally, compared to EAF, BOF relies more on the quality and quantity of pig iron or similar raw material rather than scrap metal.

In summary, the history of steelmaking is rich, marked by numerous iterations of new inventions and innovations that have shaped modern-day steel production. The authors in [37] depict the progressions of development as a tree, where each branch represents an evolution of a particular technology until its promises are exhausted, prompting a shift to another branch.

At present, there are two main large-scale methods, Electric Arc Furnace (EAF) and Basic Oxygen Furnace (BOF) that stand at the forefront of the steel industry. According to the World Steel Association [332], in 2021, approximately 70% of global steel productions adopt the BOF in a blast furnace, resulting in the annual production of approximately 1.4 billion tonnes of crude steel while EAF methods account for 30% of global production, yielding around 560 million tonnes annually. The global steel industry utilises both EAF and BOF methods, each providing unique advantages and applications, contributing to the versatility of the steel manufacturing industry.

As mentioned, our research is an Industrial Case project in partnership with Tata Steel. Much of the steelmaking knowledge is obtained from engagement with experts in the Port Talbot plant. Figure 2.6 displays the layout of the plant when it was first designed by the Steel Company of Wales in 1947 [136]. In this thesis, the main research and contributions primarily focus on one specific process of steelmaking known as *cold rolling*, which we introduce in the next section.

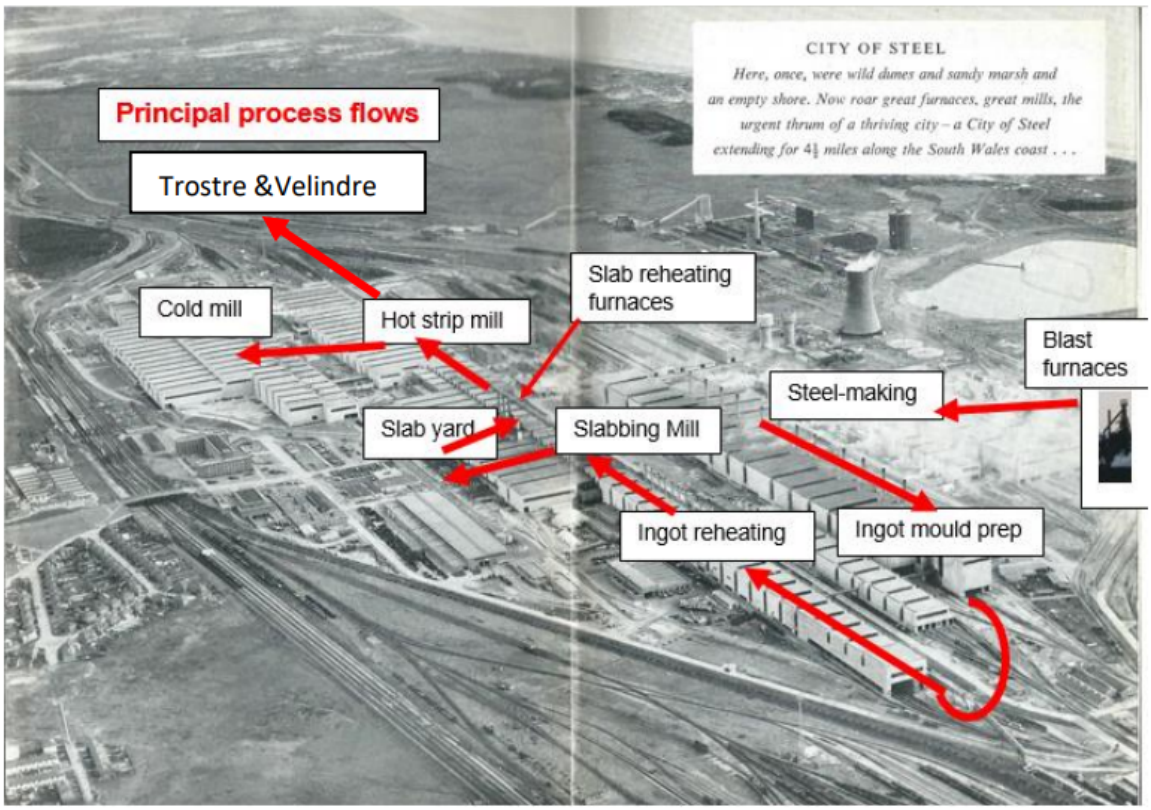


Figure 2.6: A diagram of the full Port Talbot power plant, designed in 1947 by the Steel Company of Wales [136].

2.2 Rolling of Steel

One important process in steelmaking is rolling. Rolling involves applying plastic deformation to metal through the use of rolls [283]. Plastic deformation refers to the permanent change in the shape of a material due to the influence of strong, applied forces [3]. In rolling, the steel material, in the form of a strip, is compressed between two rotating rolls, where high compressive stresses are applied to the metal strip, resulting in the desired shape. As the material passes between pairs of rotating rolls, its diameter is incrementally reduced.

Figure 2.7 displays a wide range of different types of mill stands. This figure illustrates several metal strips decreasing in size as they pass through the rolls. There are two types of rolls: *work rolls*, which are in direct contact with the material, and *backup rolls*, which apply compressive pressure on the work rolls. These types will be discussed in greater depth in the next section. These examples represent longitudinal rolling, the most common form of rolling, where the axis of rotation is parallel to the direction of motion [334]. Alternative methods include transverse rolling, where the rotation is perpendicular to the direction of motion, and skew rolling, where the axis of rotation is at an angle to the direction of the

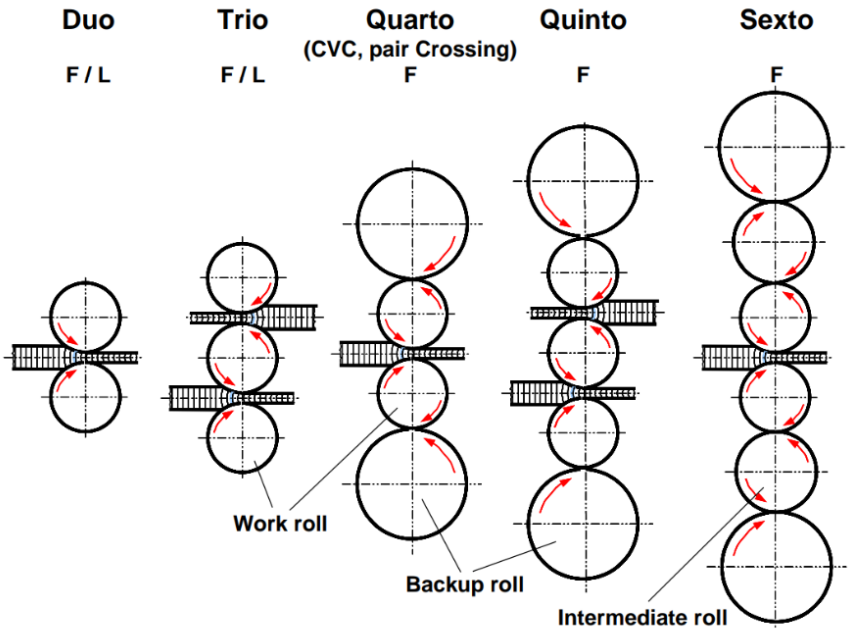


Figure 2.7: Numerous variations of mill stands with different load and stress distributions from [283].

motion. Transverse and skew methods are primarily used for hollow products such as tubes [334].

Rolling can be applied at two different temperatures, each resulting in steel with different sets of properties and advantages. In *hot rolling*, the process conducted at a high temperature, typically above the recrystallisation point of the steel– the temperature at which the grains in the metal structure are replaced by new stress-free grains [288]. On the other hand, *cold rolling* involves compressing the metal below its recrystallisation temperature, normally around room temperature. However, the work rolls are significantly impacted by strong forces and operating conditions due to the nature of plastic deformation [267]. Consequently, prolonged usage results in wear on the work rolls, requiring them to be refurbished. Neglecting the refurbishment of worn rolls can result in poor-quality steel, containing marks and dents. During the refurbishment, the surface of work rolls is marginally reduced to remove the worn surfaces, combating roll wear.

The resulting products of the rolling process are known as *coils*, which are very thin and ductile steel sheets that are subsequently coiled. Coils can have different properties, making them versatile for various applications in many industries such as manufacturing, automobiles, and many more. Hot rolled coils are typically used for structural components such as railroad tracks, whereas cold rolled coils are commonly used for applications that value precise and smooth surfaces such as automotive parts and electronic components [300].

The first rolling mill is not precisely documented, but Leonardo Da Vinci is credited

with an early drawing of the concept in the 1480s [270]. Figure 2.8 and Figure 2.9 display his early drawings of rolling, which notably highlights the need for larger diameter backup rolls to support the smaller rolls.

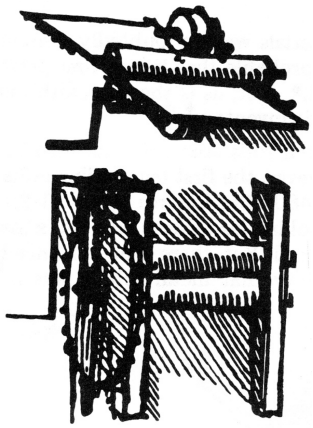


Figure 2.8: A sketch by Leonardo Da Vinci of a rolling Mill published in 1485 [205].

Figure 2.9: Da Vinci's design of a rolling mill from Science Museum, London (Science & Society Picture Library) [298].

During the 15$^{\text{th}}$ to the 17$^{\text{th}}$ century, the initial application of rolling was primarily for the production of gold lace and other soft metals [270]. It wasn't until the 17$^{\text{th}}$ century, with the invention of the Bessemer process, that rolling was extensively employed for mass steel production [77].

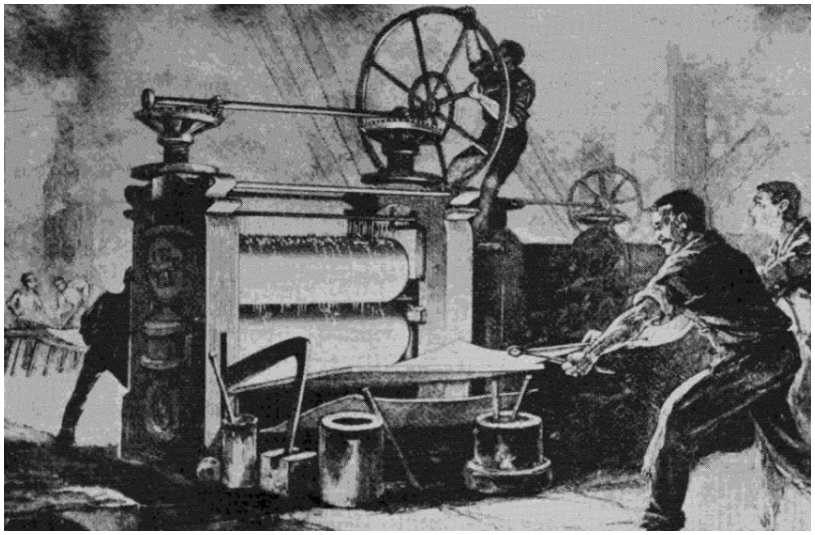


Figure 2.10: A sheet-metal mill during the Industrial Revolution, from the Metalworking World Magazine [205].

2.2.1 Use case of Cold Rolling: Cold Rolling at Tata Steel

In this section, we describe the cold rolling processes at the Port Talbot plant in South Wales. A large deal of the knowledge of the cold rolling processes presented in this section and thesis was obtained through live visits and demonstrations of the cold rolling operations at Tata Steel with experts.

In Port Talbot, home to one Tata Steel plant, cold rolling is divided into two distinct workstations. First, LINK is the name of the main operations of cold rolling, encompassing the physical processes of condensing the metal strip into sheets. This process is comprised of three sections: the pickle line, accumulators, and the rolling mill. Second, an extension that focuses on the refurbishment of the rolls.

The initial procedure of steelmaking produces undesirable oxidations on the metal. To address this issue, the metal firstly undergoes surface treatment on the pickle line. Here, the process of *pickling* cleanses the metal by using acid to eliminate impurities and oxidations, resulting in a smoother surface.

Subsequently, the pickled material is transported onto the accumulators. In this stage, the metal strips are stored and accumulated before entering the rolling mills. The accumulators enable the metal strips to move at varying speeds at different points of the full process. Thus, the accumulators play a crucial role in balancing the speed between the pickle line and the rolling mills, ensuring that the whole process is continuous and seamless from start to finish. For example, when the rolling mill is paused to swap work rolls, the pickle line can continue to run or vice-versa.

Finally, the material passes through the rolling mills, where its thickness is reduced. The rolling mill at Port Talbot compromises a total of five stands. However, the amount of thickness reduced is not uniform across each stand. Each stand is equipped with specialised machinery to achieve the optimal and precise desired thickness property. Stands 1–3 generally apply the most deformation to the steel strip, resulting in the highest reduction. Stands 4–5 are often employed to smooth or re-surface the steel strip, ensuring it meets the specifications of the customer's order.

In the rolling mill at Port Talbot, each stand is equipped with a set of components, including two work rolls, two backup rolls, and four chocks. The chocks securely hold the rolls in place and also incorporate bearings to facilitate rotation. The arrangement is such that each roll is paired with another of the same condition and diameter– whether it be two main rolls or two backup rolls. These pairs are expected to undergo their lifecycle together. Every roll is designated as either having a top or bottom status. In the event of severe damage to a roll, instead of discarding both rolls in the pair, the undamaged roll is reassigned to a new pair with the same diametric value. This approach minimises waste and optimises the use of functional components.

The backup rolls are much greater in size and are placed above and below the work rolls, applying a great amount of pressure on the work rolls. The work rolls come into direct contact with the steel material and are responsible for the deformation of the steel. Due to this direct involvement, they are more susceptible to wear, and thus, are expected to undergo the refurbishment process at TTL regularly. This process ensures that the worn

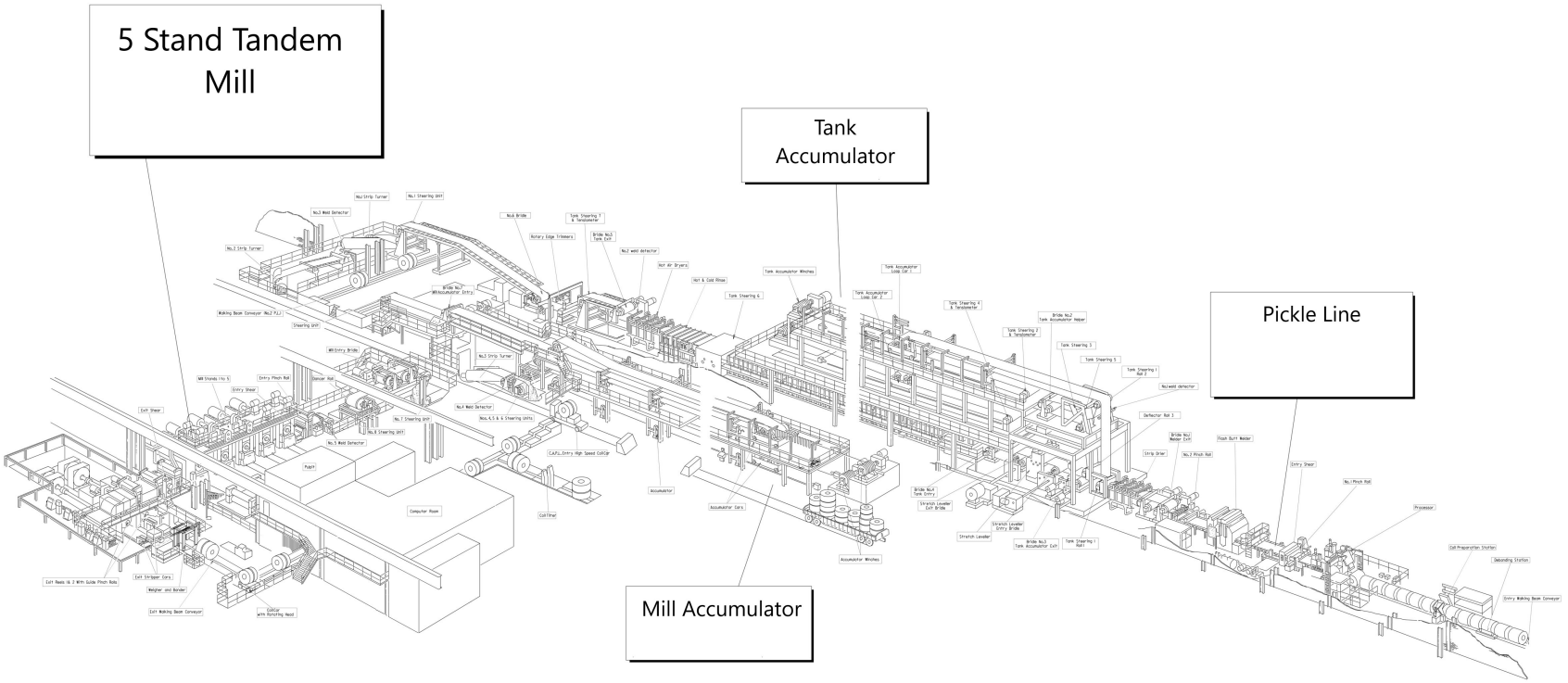


Figure 2.11: The big picture of the cold rolling processes at the Port Talbot plant provided by Tata Steel. Copyright © 2021 All Rights Reserved.

surface of the rolls is removed, maintaining the efficiency of the rolling mills and the quality of steel produced. Additionally, each roll is crafted with a slight camber at the center to provide support for the applied pressure during the rolling process.

Work rolls are originally purchased with a diameter of 600mm and repeat the refurbishment cycle until they reach a diameter of 512.8mm. At this point, they have reached the end of their operational life and are ready to be sold as scraps. Each refurbishment typically removes as little as a quarter of a millimetre, thus, each work roll is expected to be refurbished hundreds of times. Sometimes mill incidents occur which require removing greater chunks of diameter, greatly impacting the health of the rolls [266]. Backup rolls, on the other hand, are significantly larger and don't require refurbishment as regularly. Their primary role is to provide support to the work rolls, and their texture is deliberately kept slightly softer to prevent any potential damage to the work rolls.

2.3 Modern Steelmaking Architecture

With the advent of the Third Industrial Revolution (I3.0), marked by the introduction of computerisation and large-scale automation, the architectural framework of modern steelmaking adopts a hierarchical structure of systems based on the IEC-62264 standard for enterprise-control system integration [153]. This hierarchical representation is displayed in Figure **??**, where each layer signifies a distinct and separate system.

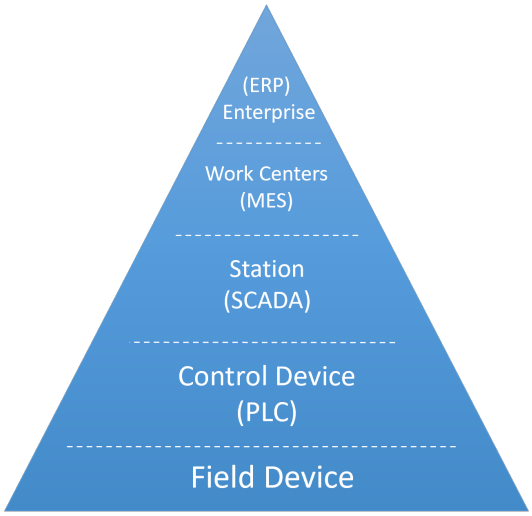


Figure 2.12: The Industry 3.0 stack based on the IEC-62264 standard [153].

Many large-scale enterprises in an organisation typically deploy an Enterprise Resource Planning (ERP) system. The purpose of an ERP system is to integrate business processes with their associated information and workflows [4]. When a new customer order is generated, it is incorporated into the ERP system, which subsequently delegates the tasks to the lower layers of the hierarchy [175]. The ERP system may include one or more work

centers, to which it dispatches the order. The work centers operate through Manufacturing Execution System (MES) systems [177], which have control over one or more Supervisory Control and Data Acquisition (SCADA) systems. Similarly, the SCADA system governs one or more Programmable Logic Controllers (PLCs), responsible for reading and writing to the field devices [75].

This hierarchical structure implies that for communication to occur between the top layer (enterprise system) and the bottom layer (field device), information must traverse through each intermediate layer [284]. In other words, establishing communication between the enterprise system and a field device requires interaction with every system within the hierarchy.

In addition, when adding a new asset or component to the plant, integration is required at every level of the hierarchy. Adding an asset to the ERP system will not automatically be included in the MES, SCADA or PLC systems. This is because each system has a unique namespace, and operates independently. Therefore, numerous separate integrations are required, with costs accumulating at each level. Integrators, known as experts specialising in data integration, often focus on a specific system. For instance, an ERP integrator may not have the necessary skills to integrate an MES or SCADA system, further increasing costs.

The advancements in digital technologies are propelling the steelmaking industry into a new industrial revolution. This vision revolves around creating a seamlessly integrated network where all systems operate under a unified namespace. In this interconnected landscape, systems are flexible, enabling direct information exchange between various components, such as PLCs sending data directly to the ERP system and vice versa. Instead of requiring integration efforts at each level, the integration process is streamlined to a one-time event when introducing a new component into the network.

As a result, this approach will not only significantly improve machine productivity, but also significantly reduce expenses, minimising the need for numerous integrations. This example of transformation is one of many benefits that illustrate a more efficient and interconnected manufacturing ecosystem.

2.4 Industry 4.0

Digital transformation in manufacturing is forging a shift in paradigm, introducing novel innovations, technologies, and concepts for organising industrial resources and processes. A handful of emerging and evolving technologies including Cyber-Physical Systems (CPS), the Internet of Things (IoT), Artificial Intelligence (AI), Industrial automation, and many others are interconnected under the umbrella term *Industry 4.0* (I4.0) [184].

I4.0 symbolises the fourth industrial revolution, linking to significant advancements in the industrial world illustrated in Figure 2.13. To briefly mention some history, the first industrial revolution, starting in the 1750s, marked the introduction of mechanisation and the adoption of steam and water power. This was the turning point of converting from human and animal labour to powered machinery, featuring innovations such as steam

engines, power looms, coke smelting, rolling processes for iron production, and many others [211]. The second industrial revolution, occurring between 1870 and 1914 (with some earlier mentions in the 1850s), was characterised by mass production, assembly lines, and the integration of electricity [212]. The third industrial revolution began around 1969, which unlocked mass automation, computerisation, and electronics. The steelmaking processes were substantially improved during this period.

In fact, the first computer in Wales was purchased by the Steel Company of Wales (SCOW) in 1960 at a price of £49,450 and was used to design the Abbey Steel Works [13]. This later became the same Port Talbot plant mentioned in our introduction, which is presently owned by Tata Steel. The device was known as the Ferranti Pegasus Mark I computer and was designed in the 1950s by former Elliott Brothers Ltd engineers, who sold around 40 Pegasus computers between 1956–1962 [285]. The first one in Wales was the 27$^{\text{th}}$ manufactured Ferranti Pegasus computer [311]. This machine was used exclusively by the Operational Research Department, which utilised the computer for production planning and control in the rolling mills [307]. The success of computerisation during the third industrial revolution in Wales led to astronomical investments in Steelmaking, for instance, the £1,400,000 investment by SCOW in 1967 when they purchased six computers [307]. I have included two articles that document this investment in Appendix A, from The Times and The Dragon, both published in May 1967.

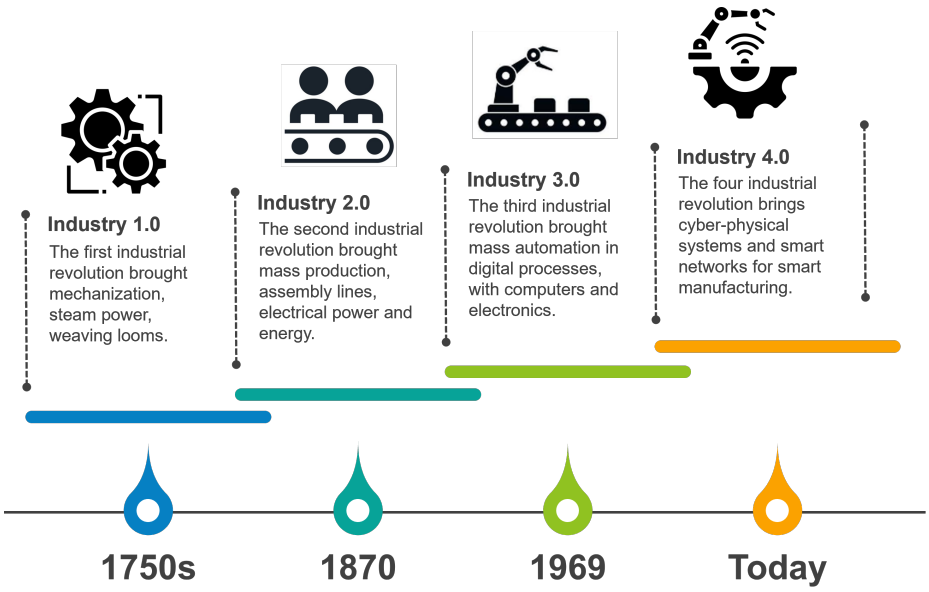


Figure 2.13: A quick summary of the four industrial revolutions.

Jumping back, the term *Industry 4.0* appeared in 2011 at the Hanover Fair in Germany by a working group from the Research Union Economy-Science of the German Ministry of Education and Research [71]. Despite its German origins, similar technology-driven initiatives within manufacturing were also emerging on a global scale. For instance, the

United States launched the *Advanced Manufacturing Partnership* was introduced in June 2011 [213], while the *European Factories of the Future Program* was initiated in 2013 [95]. Furthermore, numerous initiatives from consulting companies, technology providers, governmental bodies, and academia have introduced different labels to discuss the same shift in the manufacturing paradigm [71]. These labels include synonyms such as the *Fourth Industrial Revolution, Smart Manufacturing, Digital Transformation, Industrial Internet, Smart Factories*, and many others. In recent literature, Industry 4.0 has emerged as the key term that encompasses these labels [71]. While Industry 4.0 initially served as a proof of concept in the manufacturing domain, it has evolved with multiple implementations and applications, extending its reach into various disciplines.

The initial vision of Industry 4.0 in smart manufacturing began with designing components to use and share data without human input. Meanwhile, a study published by Acatech describes that the most crucial and difficult aspect of this vision requires achieving cooperation among the components in a plant [135]. Thus, Industry 4.0 seeks to leverage the widely interconnected digital trends to achieve "total connectivity" of data, which in turn, facilitates the cooperation and collaboration among technical objects. [148].

To achieve this level of digitisation, a crucial task involves the installation of sensors on hardware devices to capture all generated data as logs. Subsequently, the adoption of new technologies can utilise these logs, enabling devices not only to exchange information but also to make autonomous decisions, resulting in the transformation from regular shop-floor components into Industry 4.0 components (I4.0 components).

To achieve this goal, reference architecture models have been developed on a global scale.

2.4.1 Reference Architecture Models

A reference architecture model is a standardised and abstract framework for understanding relationships among entities within an environment, which can be applied to guide the development of systems, solutions, and application architectures [193]. The primary objective of the model is to define the overlying architecture of a system, by declaring consistent standards or specifications to support its necessary environment. For example, consider a reference architecture for a residential house. For a residential house, it is necessary to provide one or more bedrooms, bathrooms, kitchens, and living room spaces. The house must also be accessible through one or more doors, with enough security to detect and prevent unauthorised access. Additionally, the construction must be resilient against environmental threats, including fires, hurricanes, and earthquakes [151].

There is a great level of abstraction when creating a reference architecture model. In the example, the focus was on identifying the fundamental elements of all houses, and not one particular instance. Additionally, the architecture model does not focus on specific standards, technologies, or implementations. That is, it does not state that every house must be constructed using bricks, but instead provides a conceptual structure without specifying such detailed requirements or limitations.

Many countries have developed their own reference architecture models to realise Industry 4.0. This includes the Reference Architecture Model for Industry 4.0 (RAMI 4.0) [130], the European initiative led by the German Electronic and Digital Industry Association, ZVEI, in collaboration with the Association of German Engineers consortium, VDI. Another example is the Industrial Internet Reference Architecture (IIRA) model [189], the initiative led by the Industrial Internet Consortium based in the U.S. The IIRA focuses on encapsulating all *smart* applications, such as smart homes, smart cars, smart manufacturing, etc., whereas RAMI 4.0 solely focuses on smart manufacturing and Industry 4.0. Further models include the Industrial Value Chain Reference Architecture (IVRA) [226], representing the Japanese conceptual architecture for smart factories, or the China Communications Standards Association (CCSA) architectural model specifically for Internet of Things (IoT) related models [66], and China Intelligent Manufacturing System Architecture (IMSA) for smart manufacturing [329]. In this thesis, we primarily focus on the RAMI 4.0 model but briefly touch on the IIRA model in the next section.

2.4.1.1 Industrial Internet Consortium

Firstly, the Industrial Internet Consortium (IIC) based in the U.S., is the leading group that is investigating Internet of Things and Services (IoTS) [150]. This includes any form of service that can be improved through the use of IoT. Examples include smartphones, smart homes, smart mobility, etc. As Industry 4.0 falls under smart manufacturing, it is covered within their Industrial Internet Reference Architecture (IIRA) model.

IIRA contains the Industrial Internet Architecture Framework (IIAF) [150] which contains all the information identifying the fundamental architecture constructs. These constructs are divided into four main categories:

- Concerns: Concerns refer to any topic of interest related to the system.
- Stakeholders: Stakeholders are people or organisations interested in a concern.
- Viewpoint: Viewpoints are conventions framing the description and analysis of specific system concerns.
- Model Kinds: Model kinds aid the task of describing, analysing, and resolving concerns.

These categories are stored under "frame", as represented in Figure 2.14, which form detailed viewpoint-by-viewpoint analysis of potential concerns for adopting Industrial-scale Internet of Things (IIOT) systems, raised by their stakeholders. We use the business viewpoint as an example to describe the process of IIRA. In this viewpoint, stakeholders set out to identify new visions of how the adoption of a new IIoT system could benefit the organisation. The stakeholders establish the values and notable experiences this new vision could accomplish. Afterwards, the stakeholders develop the key objectives that will drive the implementation of the vision. From these objectives, the fundamental capabilities required for the system are derived.

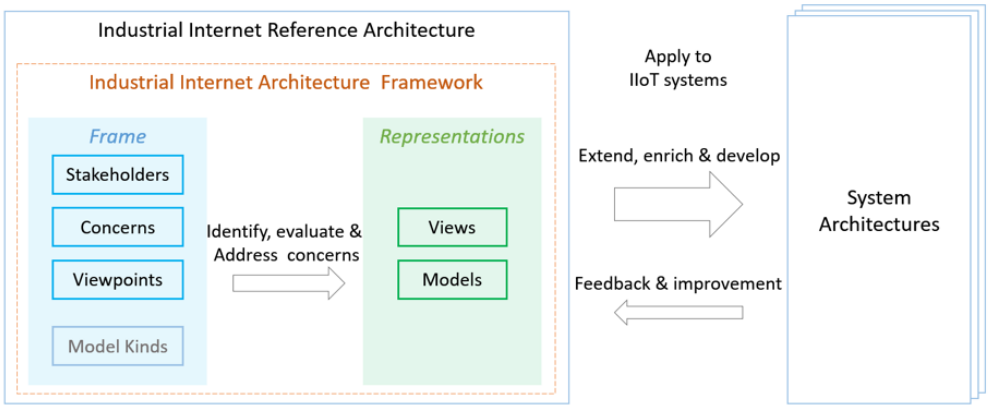


Figure 2.14: The IIC Industrial Internet Architecture Framework [150].

2.4.1.2 Reference Architecture Model for Industry 4.0

As discussed previously, the IIRA focuses on all things that are "smart" which includes smart manufacturing and smart factories. In contrast, Europe has a specific focus on Industry 4.0, leading to the creation of their own architectural framework, known as the RAMI 4.0 mode. RAMI 4.0 is displayed in Figure 2.15.

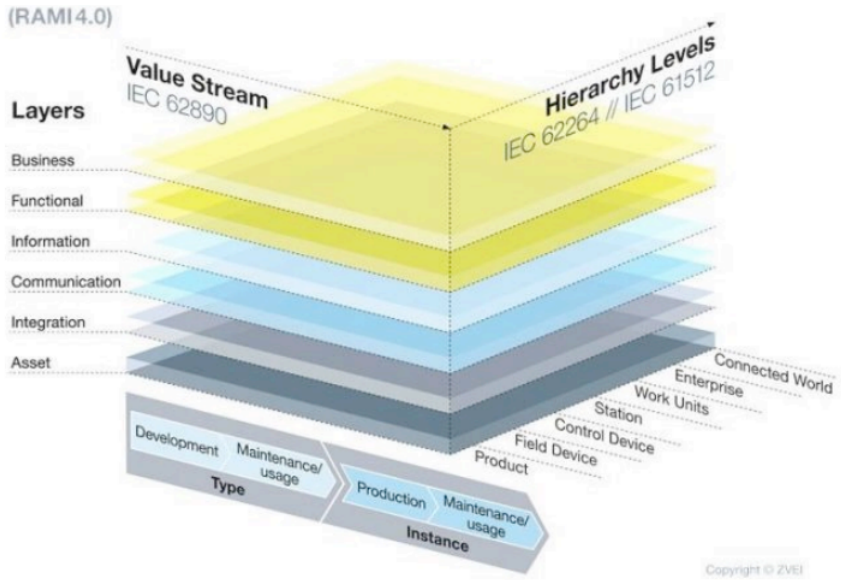


Figure 2.15: The Reference Architecture Model for Industry 4.0 (RAMI 4.0) model by ZVEI [318].

RAMI 4.0 is a 3D model framework, resembling the Smart Grid Architecture Model

(SGAM), and is designed to highlight the most important aspects of achieving Industry 4.0 [2]. SGAM, initially developed to facilitate new communication protocols in networks of renewable energy sources [345], serves as the basis for RAMI 4.0. This model uses a "divide and conquer" strategy to address the complex processes of Industry 4.0 by breaking them down into smaller and more manageable processes.

Like all 3D models, RAMI 4.0 contains three axes: the layers axis, the lifecycle and value stream axis, and the hierarchy levels axis. We will use examples from the cold rolling processes to describe some concepts.

Layers Axis Firstly, the layers axis, on the left-most axis of the model, represents the processes that describe the technical and business perspectives of a product [345]. The RAMI 4.0 model consists of six architectural layers: Asset, Integration, Communication, Information, Functional, and Business.

Layer	**Description**
Asset	The asset layer encompasses all tangible entities within a manufacturing environment that can be digitally connected. In the scenario of cold rolling, this may involve physical entities such as the rolling mills, rolls, coils, and other machinery, as well as intangible elements such as documents, plans, orders, and personnel involved.
Integration	The integration layer serves as a link between the physical assets from the asset layer and their digital representations. The objective of this layer is to compose a unified way of collecting data from all assets in the asset layer and digitally storing that information. This can be achieved with the use of sensors and actuators and other forms of connections such as QR codes. The collected digital data is then categorised within the subsequent layers.

Communication	The communication layer is responsible for establishing standardised communication between the integration and information layers, facilitating end-to-end data exchange paths and interconnectivity among components. Since various components are created by different companies and, thereby, use different communication protocols; this layer acts as a mediator. For instance, a Siemens machine may communicate differently than a machine from another manufacturer. The communication layer is tasked with developing a universal standard for communication between servers and devices in the field. Additionally, it defines how data should be packaged within its payload. There has been extensive discussions on the communication layer, as the widely used TCP/IP communication standard was found to be non-compliant with Industry 4.0 requirements. Consequently, alternative technologies have been researched to fill this gap. The ZVEI group has reached a consensus to adopt OPC Unified Architecture framework for this task, which we define later.
Information	The information layer holds the essential data and attributes of an asset, representing the digitalised information obtained from sensors in the integration layer. Taking an example of cold rolling, it would encompass the roll data, which includes the roll's unique identifier, position in the mill, initial diameter values, and other relevant information.
Functional	The functional layer formally describes the functions and services of an asset, outlining their purposes. This layer acts as a foundation for the horizontal integration of various services [99]. It is where rules and decision-making logic are generated and stored.
Business	The business layer is the enterprise-level layer, which deals with all the business models and overarching processes. The purpose of this layer is designed to uphold the integrity of functions in the value stream and finalise the rules that the system is required to follow [2].

Table 2.2: The layers axis of the RAMI 4.0 models.

Each layer encapsulates fundamental concepts, and introduces potential technologies to achieve each goal. In this thesis, we focus primarily on the semantic interoperability issues within the information and communication layers.

Life-cycle and Value Stream Axis This axis expresses the life-cycle and associated value stream of products, machinery, factory, and other assets according to the IEC 62890 standard (Industrial-process measurement, control and automation - Life-cycle-management for systems and components standards) [155]. The purpose of this axis is to record all

relevant data of a product– from its initial idea and design, until the end of its life. This is divided into two main sections: *type* and *instance*.

- *Type*: A type is always instantiated with an initial idea. It involves planning the idea and describing the asset in its most basic and first concept. This branches out to the development, maintenance, and usage phases of the idea. Examples may include low or high-fidelity prototypes, simulations, or instruction manuals.

- *Instance*: Instance describes the concrete assets and their operations, based on the idea formed in the *type* stage. This ranges from production to usage and maintenance of the asset, including product details, usage, and services, until it is scrapped.

An example of this process would be the development of a new roll for the cold rolling mill, representing the creation of a new type. The roll is designed to meet specific standards, and undergoes development, sampling, testing, and validation through prototyping and simulations. This process may involve several iterations, but once the design has been adequately validated, the roll type is released for sale and assigned its own product code within a market. Afterwards, series production will begin, where each roll is manufactured and given a unique serial number for identification and becomes an instance of the asset design developed in the type stage.

Hierarchy Levels Axis The final axis of the reference architecture model describes the hierarchy changes required for Industry 4.0. The core structure of this axis is based on the IEC-62264 (Enterprise-control system integration) [153] and IEC-61512 (batch control) [152] standards within a manufacturing environment. These standards cover various sectors of a factory, from the process industry to factory automation. The RAMI 4.0 model incorporates key terms from these standards, such as *Control Device, Station, Work Centers, and Enterprise*, and introduces additional layers, namely *Product and Field Device* at the bottom and *Connected World* at the top.

The Field Device represents the functional level of an intelligent field device, such as a smart sensor [2]. Additionally, the product itself is a crucial aspect of Industry 4.0 and remains integral to the process even after manufacturing and sale. Finally, the *Connected World* layer is introduced above the top hierarchical element of 'Enterprise' found in the previous standards. Industry 4.0 aspires to establish a higher level of connectivity compared to the factory level, emphasising collaboration not only within a single factory but also extending to other factories, suppliers, customers, and beyond.

2.4.2 The Industry 4.0 Component

Numerous manufacturers are currently developing new components for Industry 4.0 (I4.0 components) [331]. However, the majority of industrial manufacturing still operates under Industry 3.0, utilising legacy hardware and systems that do not comply with Industry 4.0 standards. The task of replacing these components is not only challenging but also

exceptionally expensive. Instead, Industry 4.0 aims to transform existing legacy components into compliant I4.0 components.

An I4.0 component functions as a model designed to encapsulate all data pertaining to an asset, encompassing properties, configuration parameters, and skills of said asset [100], essentially constituting a Cyber-Physical System (CPS). More specifically, an I4.0 component serves as a globally and uniquely identifiable participant with communication capabilities within an I4.0 system [49]. These components comprise two integral parts:

- *Asset*: Physical or immaterial objects existing in the real world e.g., legacy hardware or systems.
- *Administration shell*: Digital representation of the asset, encompassing digitised legacy hardware or systems and their associated data.

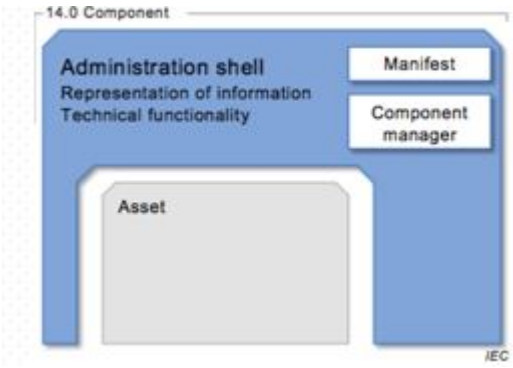


Figure 2.16: A representation of an I4.0 Component by the IEC [119].

Any asset that is equipped with an administration shell is described as an I4.0 component; this can range from a module within a machine to a whole production system [119]. Subsequently, communication occurs over an I4.0 network between administration shells, each of which contains formal and unified methods of storing and transferring information. The minimal level of communication necessary for an asset to be considered an I4.0 component is to provide passive communication capabilities. Passive in this context implies that the AAS can be a static document or package of files which simply provide all information related to the asset e.g., RFID or barcodes [250].

Based on the specifications in [49], I4.0 components require the following properties:

- *The component is clearly identifiable as an entity:* This implies the component requires a unique identifier in the network, ensuring all assets have their own non-duplicate ID.
- *Categorised as either type or instance:* The I4.0 component carries all information of an asset throughout its lifetime. The asset is either in its initial development stage (type) or a concrete asset that has its own operations (instance).

- *Capable of active or passive I4.0 communication:* The purpose of Industry 4.0 is the connection and interaction between devices. I4.0 components require secure I4-compliant communication, services and QoS. I4.0 components communicate on the basis of a Service-Oriented Architecture (SOA) which will include common I4.0-compliant semantics.

- *A representation of an asset by means of information:* AAS represents the data and dynamic behaviour of an asset. The information stored extends to characteristics related to the business model, functions, performance, and more [191], which are stored inside as a partial model within the administration shell.

The concept of an I4.0 component is an enabler to characterise the communication between virtual and cyber-physical objects and processes [195]. An I4.0 system is comprised of numerous I4.0 components.

2.4.2.1 Asset Administration Shell

The Asset Administration Shell (AAS), sometimes referred to as an Administration Shell, is the key enabler of transforming regular components into I4.0 components. It is the virtual and digital representation of the asset that contains all data shared and functions of an asset [328]. The data stored contains all the life-cycle data of the asset and provides the status of the asset, allowing the information to be shared over an I4.0 network. The AAS is able to provide controlled access to all information of the asset [252], and is developed to support interoperability between application managing manufacturing systems from hardware systems to software components [62].

Each embedded system will contain its own administration shell with active I4.0-compliant communication capabilities [49]. This model enables ubiquitous communication and common understanding between machine-to-machine (M2M), hardware and software components as well as humans-to-machine (H2M) from their accessible virtual interface [101].

Each AAS contained two compartments: a *Digital Factory Header* and a *Digital Factory Body*. The header contains the unique identification and information to identify the asset of the administration shell. In contrast, the body contains the information and functions of the asset. The data is represented and categorised as submodels containing hierarchical properties to specific parts of the information i.e., two submodels may lead to two different functions of the asset. These property descriptions are required to follow specific standards such as IEC 61360 (Standard data element types with associated classification scheme) [154].

The fundamental requirements of an AAS are presented in Table 2.3 [287]. As mentioned in the requirement, each AAS must contain a *manifest* and a *component manager*. A manifest contains the mandatory information about the asset, structured similarly to a table of contents, containing all the data of the AAS and component itself, e.g., its data and functions. The manifest also represents how the AAS is connected to the real asset through appropriately secure identification [49]. The component manager, on the other

Table 2.3: The fundamental requirements of an Asset Administration Shell.

ID	Requirement Details
1	The AAS must consist of a body and header, where the body contains all information of the asset and the header contains the meta-data and the utilisation of the asset.
2	The AAS must consist of a manifest and component manager.
3	The information in the AAS should be accessible via service-oriented architecture such as an API call.
4	The AAS must represent information about different application aspects of the asset.
5	The AAS must be structured in a format that is compatible with existing views (pursuant to MES ISO/IEC81346, Digital Factory BCFLP etc.)
6	The AAS must have a unique ID.
7	The asset represented by the AAS must have a unique ID.
8	The factory must be considered an asset, and therefore, have its own AAS with a unique ID.
9	The AAS types and instances must be identifiable.
10	The AAS should be able to include references to other AAS or I4.0 information (i.e., comprising of the submodels describing the asset).
11	The AAS must be possible to include additional properties, e. g. manufacturer-specific.
12	Each AAS must contain a reliable minimum number of properties and their definitions.
13	All properties and information captured in the AAS must be suitable for types and instances.
14	The AAS must provide a hierarchical and countable structure to store its properties.
15	The properties of one AAS must be referenceable by other properties in other AASs.
16	The data and functions of an AAS must be referenceable by properties of the AAS.
17	Security-based measurements e.g., availability, integrity, confidentiality, visibility, and authenticity must be made available to the properties of the AAS.

hand, provides external access to the data and functions stored in the other components on the I4.0 network.

2.4.3 Enabling Technologies for the Communication and Information Layers

To tackle some of the AAS requirements, a joint working group including OPC Foundation, ZVEI, and VDMA have investigated possible technologies and solutions for the implementation of AAS. Specifically, we mention the main standards for the communication and information layers of the RAMI 4.0 model.

OPC Unified Architecture The working group have chosen the OPC Unified Architecture (OPC UA) as the standard for machine-to-machine communication, [250] covering the communication layer of the RAMI 4.0 model. The OPC UA protocol follows the IEC-62541 standards [125], which are derived from the OPC UA's Core, Access, and Utility Specifications [133], as shown in Figure 2.17.

OPC UA is a platform-independent and connection-oriented communication protocol based on Service-Oriented Architecture (SOA), developed by the OPC Foundation [341]. The purpose of OPC UA is to standardise vendor-independent communications between machines and smart systems [90] and improve interoperability. Typically, these assets and systems follow different methods of communication such as SERCOS, ProfNET, or other ethernet-based fieldbus protocols, which need to be unified under one namespace.

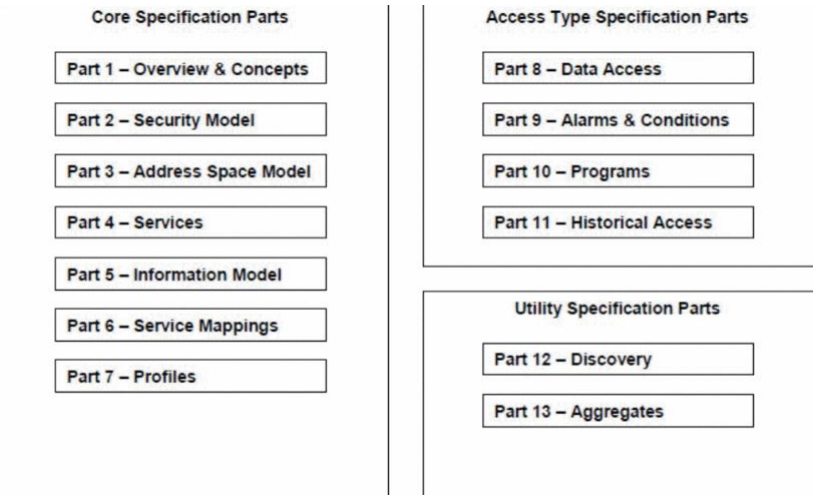


Figure 2.17: The OPC UA specifications from [230].

Similar to many communication protocols, OPC UA operates under a client-server session protocol and can be installed on small sensors and programmable machines such as PLCs, enabling communication over a single thread [199]. Additionally, supervisory systems such as SCADA, MES, and ERP fall within the scope of integration with OPC UA, covering the entire automation pyramid [199].

In 2019, OPC UA released a publish/subscribe (PUB/SUB) specification, allowing OPC UA servers to publish data, where clients can subscribe to specific parts of this data, independent of its origin. [257].

The main benefits of adopting the OPC UA standards include interoperability and machine-to-machine communication, enabling the transmission of low-level signals from field devices at a semantic level. Additionally, OPC UA provides improvements in performance, and stability, and incorporates built-in robust security [341]. These features contribute to its wide adoption in industrial systems. However, despite these improvements, the semantics covered by OPC UA are defined within specification documents, captured in an implicit format that is only interpretable by human understanding [281]. In other words, OPC UA does not have the semantic capabilities to satisfy large-scale information-sharing requirements fully [110], and thereby, only provides limited capabilities for enabling automated reasoning and knowledge inference within the OPC UA namespace [17]. Thus, we investigate other technologies to represent and process explicit semantics for the information and communication layers of RAMI 4.0.

AutomationML AutomationML (AML) is an XML-based data model and data exchange language designed specifically for modelling manufacturing systems [341], following the IEC 52424 standards [156]. AML is capable of adapting, extending, and combining existing standardised industrial data formats, such as CAEX, COLLADA, and PLCopen XML systems that model a wide range of data, including signals, geometry, kinematics, and sequential behaviours [87]. CAEX serves as the top-level core of AML, functioning as an object-oriented data format that can be utilised for various data formats. AML facilitates systematic management of data exchange workflows between multi-disciplinary engineering tools.

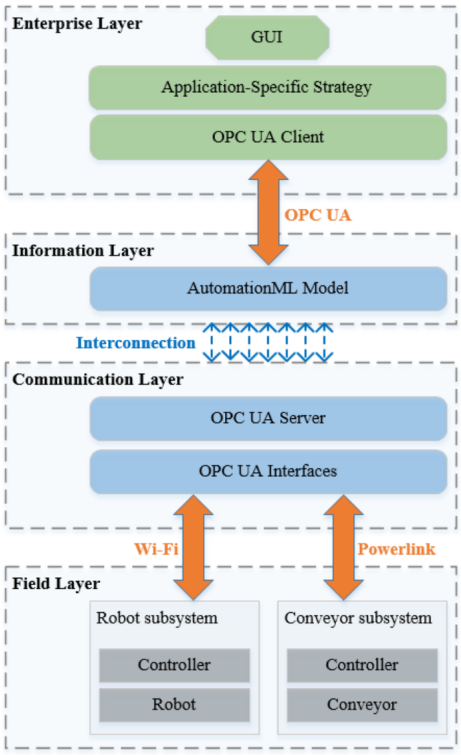


Figure 2.18: One implementation of a manufacturing system demonstrating the combination of OPC UA and AML [342].

Figure 2.18 is a snippet from [342] that illustrates one example of how OPC UA and AML can be interconnected within a manufacturing system. The authors in [342] condense the RAMI 4.0 model into four layers, where OPC UA was implemented for the communication layer, and how AML was interconnected to represent the information layer. This approach demonstrated how data from the field layer could be made accessible at the enterprise layer by combining these two standards. Specifically, communication with the field devices occurred over an OPC UA server via different communications, in this case, WiFi and Powerlink. This server was accessible through an OPC UA interface, interconnected with the information layer via a pre-set AML data model. This model was reachable by the

enterprise layer through a graphical user interface (GUI) or API call to the OPC UA server.

2.4.4 Summary

To address the challenges posed by Industry 4.0, several architectural frameworks have been developed. In this section, we introduced the European initiative, the RAMI 4.0 model, which breaks down complex challenges into smaller and more manageable tasks. The model consists of six hierarchical layers. The communication and information levels focus on maintaining a unified structure for capturing and autonomously transferring information between components within a smart manufacturing environment. Many international standards, such as OPC-UA for data exchange and AutomationML for data access and storage, have been adopted for this task.

However, in the broader context, while these standards are able to enable the exchange of information between devices on the shop floor, there exists limited semantic interoperability. Consequently, there is a need to explore other semantic methodologies as a possible solution to address the remaining challenges with semantic interoperability, which we aim to realise through this thesis.

2.5 Introduction to the Semantic-Web: Technologies and History

The World Wide Web (W3) was founded in 1989 by the World Wide Web Consortium (W3c) led by Tim Berners-Lee and was initially designed as a medium for storing and displaying documents for humans [31]. Meanwhile, the Semantic Web is an extension of the W3 in which information is given explicit meaning and is stored in a structured format that can be processed automatically by a machine [33]. This structure provides meaningful content on the web where knowledge-driven and sophisticated tasks can be executed autonomously and automatically by the computer, while not directly affecting the user.

Consider the following scenario, adapted from [33] to the context of the thesis: picture a garage owner looking to purchase some `steel parts`. They search the web for `steel manufacturers`, where a list of possible `providers` is displayed. The search shows providers within a `30-mile radius` of the user's `location`, each containing a `customer rating` ranging from `one to five stars`. The user clicks on the manufacturer with the highest rating, also the `nearest` to the customer's `home`. However, the specific components the user requires are `unavailable`. Thus, the user sets stricter preferences for location and product availability, which instantly displays available suppliers. The second selected supplier would take `longer` to deliver, which is `less important` to the user, while the `stock price` was `cheaper` which is `more important`. The emphasised keywords relate to terms whose semantics or meaning are not readily clear for a computerised system but can be defined through the Semantic Web. Therefore, the next time the same user conducts a similar search, the software agent can carry out similar sophisticated tasks without or with minimal human input.

The W3C has introduced a wide range of different languages and technologies, which have been innovated over many years to achieve this vision. In this section, we cover the fundamental technologies and languages for data semantics, including Knowledge Representation and Ontologies, which we apply in an Industry 4.0 setting.

2.5.1 Knowledge Representation

The first technology the W3C mentions is the advancement of Knowledge Representation (KR) [33].

Ian Horrocks, a key contributor within the KR community, defines the term "Representation" as the *'relationship between constructs in a language and entities in the world, namely the relationship between syntactic constructs in a formal language and objects in some other formal system that is supposed to be an analogue of (part of) the world.'* [143].

In other words, the notion of KR focuses on how *knowledge* can be represented symbolically and in a format that can be comprehended and manipulated by both humans and computers in an automated way. It is the idea of modelling *what a human knows* in a systematic and explicit format for logical reasoning [186].

Knowledge itself, in the context of Artificial Intelligence, can be defined using the Data, Information, Knowledge, and Wisdom (DIKW) hierarchy [291], displayed in Figure 2.19. In order to define and model knowledge, we must first define *data* and *information*. Firstly, data, in its raw form is an individual fact, which is considered to *know nothing* by itself [344]. Meanwhile, data can be collected, processed, and transformed into relevant information, providing useful answers to who, what, where, and when questions. In contrast, knowledge refers to the *know-how* and is capable of providing comprehensible insights and patterns obtained from processed information, enabling intelligent systems to understand, learn, and infer decision-making. The final layer, *wisdom* refers to evaluating and reflecting on decisions, leading to the *know why*.

The DIKW hierarchy can be applied to many domains, which include artificial intelligence and smart manufacturing. According to a study in [291], the first mention of such a hierarchical structure is from a poem by T.S Eliot in 1934 named *The Rock*, which quotes:

```
Where is the wisdom we have lost in knowledge?
Where is the knowledge we have lost in information?
```

In contrast, the concept of knowledge representation is not a recent development either. The practice of using a collection of symbols to represent information dates back to logicians such as Augustus De Morgan (1806–1871) and George Boole (1815–1864), who made significant contributions to propositional logic, or Gottlob Frege (1848–1925) and his introduction to First-Order Logic (FOL) in his book *Begriffsschrift* in 1879 [97]. FOL is an extension of propositional logic which is capable of expressing knowledge using mathematical logic with quantifies and variables [23].

Knowledge representation played a pivotal role in early AI systems, which utilised knowledge and reasoning methodologies to deduce and provide *inference* of knowledge (i.e., ob-

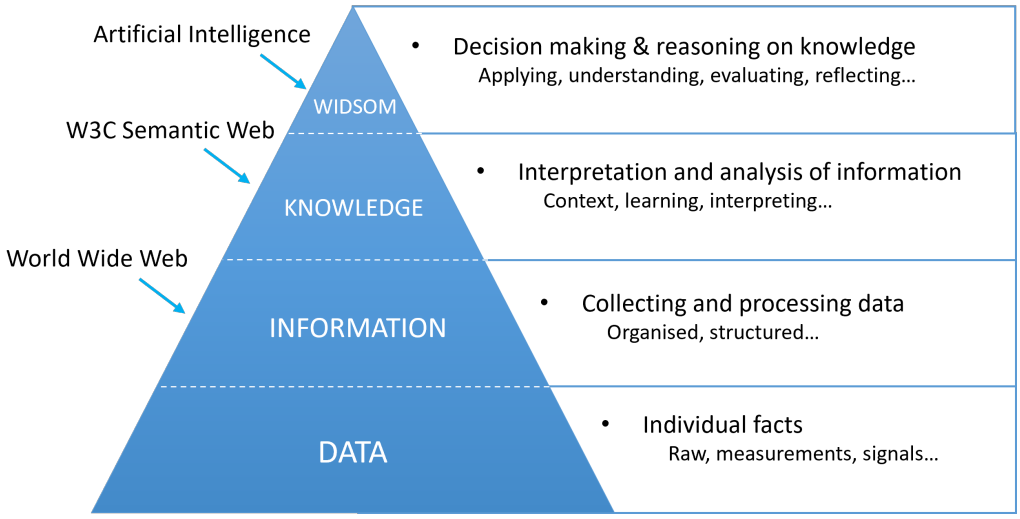


Figure 2.19: The DIKW Hierarchy in the context of the AI.

taining new expressions from old ones). One of the first few examples of this paradigm was led by Allen Newell and Herbert Simon at the RAND Corporation. In 1956, they created *The Logic Theory Machine*, which integrated mathematical logic as formal heuristics in order to automate human problem-solving [225]. Their outlook heavily focused on modelling a piece of problem-solving behaviour in terms of a generic program, rather than using computers as a "crude analogy to human behaviour" [223]. In 1959, they developed the *General Problem Solver (GPS)* whose purpose was to solve a range of different problems. However, their report claimed that GPS lacked several capabilities, but it highlighted fundamental challenges that needed to be achieved within the field of AI [224].

Meanwhile, other KR-based systems began to arise around the same period, that were programmed using the LISP processing language introduced in 1958, which provided features for symbolic reasoning [202].

In the following decades, *Expert Systems* began to arise, which are knowledge-based systems developed as applications for simple problem-solving tasks [159]. By definition, an expert system is a computer program that can: (a) reason with symbolic knowledge, (b) use methods that are heuristic and algorithmic, i.e., plausible and certain, (c) simulate and perform as well as specialists in the corresponding domain, (d) make understanding of its output, and (e) retain flexibility [51].

One example of an early expert system was the *Project MAC SYmbolic MAnipulator (MACSYMA)* system developed at Massachusetts Institute of Technology (MIT) between the years of 1960–1980 [198]. MACSYMA was a computer algebra system that focused on automating mathematical operations. This was achieved by capturing and modelling the concepts of mathematical operation as knowledge in the form of programming rules [108]. Each mathematical operation was stored as a module, constituting a centralised repository of knowledge. Users could add their knowledge and definitions into the system, provided

they followed the correct procedure.

The creators of MACSYMA stated that if such a system could be developed and maintained, it could be so impactful that it would succeed books; stating that *"books would still be used, but only for tutorial exposition"* [198]. However, MACSYMA did not live up to that expectation as it faced several challenges including learning, resource knowledge, and communication difficulties [108]. Users had to mechanically adapt their cognitive thinking process to accommodate the nature of the system. As an example in [85], picture a user attempting to define an even number. One user may say "n is even if and only if there exists an integer, m, such that $n = 2m$". Although this is true, a more explicit definition would be necessary, such as "if dividing n by two returns no remainder, then n is even". Often these translations were too challenging or time-consuming for most users.

Overall, these knowledge representation systems were often faced with great difficulty, requiring sufficiently precise notation for representing knowledge [221]. In these systems, knowledge has to be centralised, meaning that all people must share exactly the same definition of common concepts. Thus, the capabilities of such systems were constrained and limited to asking specific questions that the machine could reliably answer, and increasing the scope of such a system would be deemed to be unmanageable [33]. Berners-Lee compared the problem with Kurt Gödel's incompleteness theorem ([290]), which states that there will always be true statements that cannot be proven in a formal system [33]. These KR systems often avoided such conflicts by producing a unique yet limited set of rules, specific to the task at hand.

Contrastingly, the semantic web accepts that these unanswerable questions exist, and instead, accepts to decentralise the web to achieve versatility. To obtain this level of versatility, it is necessary to use a language that enables reasoning with sufficient expressivity to capture the scope of the web and be powerful enough to describe complex properties. This language needs to (a) express both data and rules for reasoning, and (b) be able to import any existing forms of knowledge-representations systems into the web. Thus, a wide range of different languages have been investigated and proposed to achieve these requirements.

2.5.2 The Expressivity of Modelling Languages

There is a wide range of different languages and models available to describe such concepts. Figure 2.20 displays a spectrum of categories of languages ranging from informal to formal, which are applicable for capturing knowledge [316]. These languages are split into three parts. On the left side, there are glossaries and dictionaries which are lightweight but are restricted in their ability to provide greater specification of the meaning of terms. The middle section contains technologies that are able to model meta-data using XML schema and other data models such as databases (DB) that are perfectly sufficient for the exchange of data between parties but lack the semantics and meaning of the data. Meanwhile, the final section refers to languages that are able to explicitly formalise logical theories and describe formal semantics.

Each Language has its own advantages and disadvantages, but generally, as you traverse through the languages mentioned, there is a trade-off between expressiveness and

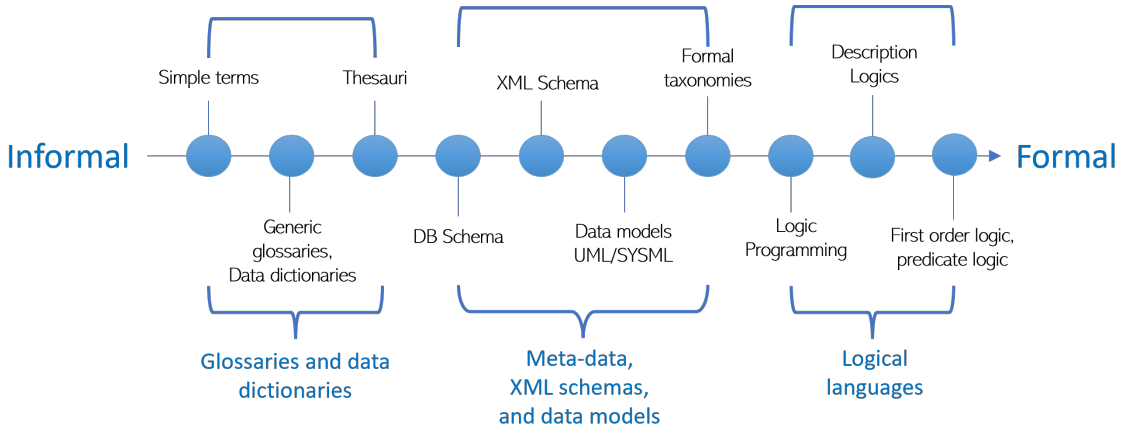


Figure 2.20: Different languages for informal and formal specification based on [316].

efficiency. The rightmost languages are superior in terms of expressivity, and are able to express more concepts symbolically, but subsequently, do not allow for sound or complete reasoning [128]. Soundness conveys that true statements are always actually true while completeness reasoning implies that the system can demonstrate and deduce statements to be true. Thus, using a more restricted yet decidable subset of first-order logic may be more suitable, e.g., Description Logics (DL) or Logic Programming (LP).

New languages that adopt these logical implementations have surfaced and are built upon one another, such as the *Knowledge Interchange Format* (KIF) language in 1992, *Options Configuration Modeling Language* (OCML) in 1993, *EXtensible Markup Language* (XML) in 1996, *Resource Description Framework* (RDF) and *Resource Description Framework Schema* (RDFS) in 1999, and *Web Ontology Language* (OWL) in 2004 [297]. In the following sections, we describe RDF and OWL in greater depth.

2.5.3 The Resource Description Framework

The Resource Description Framework (RDF) is a metadata modelling language produced by the World Wide Web Consortium (W3C) that expresses relationships between entities in a format that is both human-interpretable and machine-processable [79]. Data is presented as disjoint resources, each of which contains a unique Uniform Resource Identifier (URI) for identification. An example of a URI is *https://www.ietf.org/rfc/rfc3986.txt*, which is used as a locator, name, or both [32]. These resources join and form tuples consisting of a subject, predicate, and object, producing a directed graph. Figure 2.21 is a simple representation of a graph, displaying the subject and object as nodes, which are directly linked by an edge, representing the predicate that joins them.

To give a formal definition: let $G = (S, P, O)$ be a directed label graph as shown in Figure 2.21 where:

- S is a set of nodes that represent the subjects in the graph, which identify the resources

being described.

- P is a set of edges representing the predicates in the graph, which identify the relationships being asserted about the resources.
- O is a set of objects, of which are either literal values or other resources, identifying the information or characteristics of the resources of S.

Figure 2.21: A directed graph containing an RDF triple.

Figure 2.22 is an example of an RDF graph representing knowledge in the context of cold rolling. Each resource in the graph follows the notation of *prefix:element* where the prefix refers to the URI for identification. In this example, the resource `scro:roll_1234` is linked with two other resources: (1) the `1234^^xsd:integer` resource node via the `scro:hasRollId` resource edge, and (2) `scro:Work_Roll` resource node via the `rdf:type` resource edge. Meanwhile, the `scro:Work_Roll` resource is linked to the `scro:Roll` resource via the `rdfs:subClassOf` edge. By definition, the language is capable of deriving that the resource `scro:roll_1234` is a resource of which is a subclass of the `scro:Roll` resource.

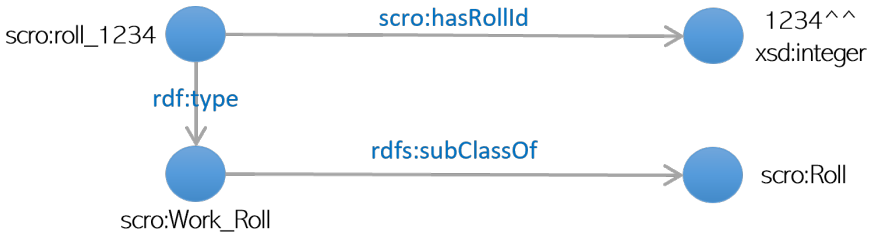


Figure 2.22: An example of an RDF graph of a work roll in the context of cold rolling.

RDF can be serialised in different formats, each of which contains its own syntax and benefits. Most common formats include RDF-based eXtensible Markup Language (RDF/XML), Turtle, or Web Ontology Language (OWL). Our contributions are mostly serialised in OWL, which provides the most functionality of the three, and presents the knowledge in a good human-readable format. We explain this language in greater depth in Section 2.5.5. However, OWL and RDF/XML contain XML-like structures with opening and closing tags which are lengthy. Therefore, the listings provided in this thesis are converted to Turtle syntax[1] to save space.

[1] `https://www.w3.org/TR/turtle/`

Meanwhile, Listing 2.1 is an example of the RDF graph in Figure 2.22 serialised in Turtle format. The top of the listing contains prefixes, which link to external RDF vocabularies. An RDF *Vocabulary* is a collection of Internationalised Resource Identifiers (IRIs) intended for use in RDF graphs [73]. IRIs are extensions of URI protocol, which expands the set of usable characters [88]. These have a common substring known as a *namespace prefix* e.g., `RDF`, `RDFS`, `SCRO` and a *namespace IRI* e.g., http://www.w3.org/1999/02/22-rdf-syntax-ns#.

Code Listing 2.1: Turtle serialisation of the RDF graph in Figure 2.7.

```
@PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
@PREFIX scro: <http://swansea.ac.uk/sadeer/scro#> .

scro:Roll_1234 rdf:type scro:Work_Roll .
scro:Roll_1234 scro:hasRollId 1234^^xsd:integer .
scro:Work_Roll rdfs:subClassOf scro:Roll .
```

The `RDF` vocabulary provides simple functionality to model the relationships between objects within a domain. Features such as `rdf:type` and `rdf:PlainLiteral`, among others are available, which we used previously in Figure 2.22. While RDF serves as a foundational data model, it does not follow a schema and lacks the inherent capability to define the meaning of data; it functions purely as a data model [52]. To embed semantics, i.e., meaning, into the data, explicit vocabularies are essential. The `RDF Schema` (RDFS) vocabulary is one example, which facilitates simple descriptions of classes and properties associated with RDF resources, including their ability to model hierarchical structures [204]. Both RDF and RDFS are standardised by the W3C. Meanwhile, the Web Ontology Language (OWL) extends RDFS, providing additional flexibility and capabilities to articulate the semantics of RDF statements. This requires the creation of an ontology. In the previous example, `SCRO` is an ontology that we have developed as a vocabulary for cold rolling processes. We define the concept of an ontology in Section 2.5.5.

2.5.4 Knowledge Graphs

A collection of RDF triples may be formed together to produce a knowledge graph. A knowledge graph, in essence, is a domain-specific database in a directed label graph structure, formed by nodes (representing entities) and edges (representing relationships). The concept of a directed label graph is not new, but the idea of capturing meta-data and relationships through edges provides capabilities to represent knowledge, which is not possible with traditional DB models like SQL.

The term Knowledge Graph was introduced by Google in 2012 when they first introduced the *Google Knowledge Graph* [346]. An article in 2019 stated that the Google knowledge graph contained over one billion entities with over 70 billion assertions [227]. Many large-scale enterprise companies such as Microsoft, Facebook, IBM, and Wikipedia have all adopted knowledge graphs as a medium to represent their data. As mentioned, knowledge graphs are domain-specific, and thereby, require some form of a knowledge base.

The knowledge base is required to provide definitions of concepts, and how those concepts are related to one another. To capture this knowledge, the concept of an *Ontology* has been developed, which we introduce in the next section. There are many applications developed for creating knowledge graphs, including Mastro [54], Morph [255], and Ontop [336]. Within this thesis, we utilise the Ontop framework to create our domain-specific knowledge graphs.

Querying Language for Knowledge Graphs

SPARQL Protocol and RDF Query Language (SPARQL) is the standard language for querying over RDF data [245]. Tim Berners-Lee, the director of W3C, states that "Trying to use the Semantic Web without SPARQL is like trying to use a relational database without SQL" [333]. SPARQL contains syntax that offers similar operations to SQL such as `select, where, optional, union, filter`, but is specifically designed for querying tuples over graph data.

Code Listing 2.2: An example containing a list of RDF triples.

```
1  @PREFIX rdf: <http://www.w3.org/1999/02/rdf#> .
2  @PREFIX scro: <http://swansea.ac.uk/sadeer/scro#> .
3
4  scro:Roll_0001 rdf:type scro:Work_Roll .
5  scro:Roll_0001 scro:hasRollID 0001^^xsd:integer .
6  scro:Roll_0001 scro:hasDiameter 500.0^^xsd:double .
7  scro:Roll_0001 scro:hasPartner scro:Roll_0002 .
8
9  scro:Roll_0002 rdf:type scro:Work_Roll .
10 scro:Roll_0002 scro:hasRollID 0002^^xsd:integer .
11 scro:Roll_0002 scro:hasPartner scro:Roll_0001 .
```

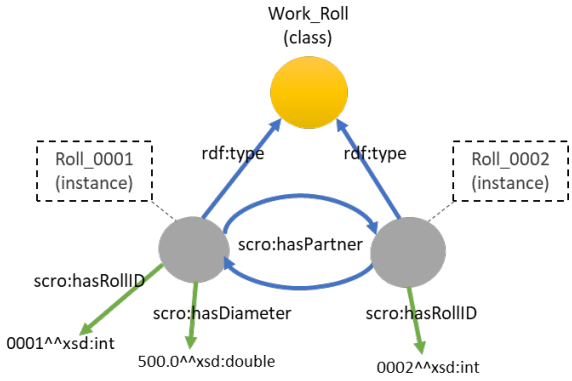


Figure 2.23: A list of RDF triples in Turtle syntax (left) and in graph format (right).

Listing 2.2 displays a list of RDF tuples that form the knowledge graph presented in Figure 2.23. In this particular example, there are two vocabularies, `rdf` representing the W3C standard RDF library and `scro` which is an ontology developed by us. The listing displays two instances being populated with RDF tuples. `Roll_0001` and `Roll_0002` are declared as instances of the `scro:Work_Roll` class via the `rdf:type` property on lines 4 and 9, respectively. `Roll_0001` contains three properties from lines 5–7 and `Roll_0002` contains two properties from lines 10–11. The difference between the two instances is that `Roll_0001` contains the `hasDiameter` property whereas `Roll_0002` does not.

Figure 2.24 demonstrates two examples of SPARQL queries based on the knowledge graph at hand. At the top of each query, different vocabularies are imported via prefixes. We import the RDF vocabulary in order to use the `rdf:type` property. Afterwards, SPARQL provides SQL-like clauses e.g., `SELECT` to select the variables we wish to retrieve from the

query. Variables in SPARQL are indicated using a question mark (e.g., `?roll`), while the end of each tuple is separated by a full stop.

In Listing 2.3, we are selecting the `?roll` variable as an output. In this case, any instance of type `scro:Work_Roll` is stored in the `?roll` variable. The result of this query will print out two results, `?roll=Roll_1` and `?roll=Roll_2`. In Listing 2.4, we are selecting the `?roll` and `?diameter` variables as outputs. In this case, the `?roll` object, must contain both a `hasRollID` property and a `hasDiameter` property. Optionally, this object may contain the `hasPartner` property. The result of this query will print out `?roll=Roll_0001` and `diameter=500.0^^xsd:double`.

Code Listing 2.3: A SPARQL query to return instances of a class.

```
PREFIX rdf: http://www.w3.org/1999/02/rdf# .
PREFIX scro: http://swansea.ac.uk/sadeer/scro#

SELECT ?roll
WHERE {
    ?roll rdf:type scro:Work_Roll .
    }
}


OUTPUT:
?roll=Roll_0001
?roll=Roll_0002
```

Code Listing 2.4: A SPARQL query to return rolls with diameters.

```
PREFIX rdf: http://www.w3.org/1999/02/rdf# .
PREFIX scro: http://swansea.ac.uk/sadeer/
     scro#

SELECT ?roll ?diameter
WHERE {
    ?roll scro:hasRollID ?id .
    ?roll scro:hasDiameter ?diameter .
    OPTIONAL {
       ?roll scro:hasPartner ?partner .
    }
}
OUTPUT:
?roll=Roll_0001, ?diameter=500^^xsd:double
```

Figure 2.24: Two examples of SPARQL queries.

2.5.5 Introduction to Ontologies

The origin of the term *Ontology* is from a branch of Philosophy known as Metaphysics and is used to define the study of existence [128]. The term dates back to ancient Greek philosophy. Plato and Aristotle defined the study of ontology as the science of *being* and used it as a term to determine whether something exists [128]. It attempts to seek the classification and explanation of the nature of *things* and their properties. The term is borrowed (sometimes referred to as a formal ontology) and is used in the context of AI for knowledge representation, exchange, and integration [321].

Before providing a definition of a formal ontology, take a minute to pause and observe your surroundings. Identify the physical and, perhaps, non-physical things around you. What are their characteristics, defining traits, and purpose? From a philosophical standpoint, creations around us are typically manifestations of thought. Things are designed and built for purpose. For example, consider the ontology of the desk you are working on—it has legs and a surface designed to fulfil its purpose of holding objects. Additionally, it contains personalised characteristics such as material, width, and height to fit in its environment.

The key message is that this desk is one instance of a table—other tables with different materials, patterns, lengths, widths, shapes, etc., exist, but all tables serve the same fundamental purpose. Formal ontologies are structured and systematic representations of these concepts or objects, which encapsulate the meaning, characteristics, and interrelationships between each concept.

The most commonly accepted definition of a formal ontology in the context of AI was given by Thomas R. Gruber [128], the co-founder of Siri Inc. In 1995, Gruber defined an ontology as an *explicit specification of a conceptualisation* [124], which was later expanded to *A formal, explicit specification of a shared conceptualization.* Defining each word in this sentence:

- *Formal:* The representation must be captured and stored in a format that is both readable by a machine while also interpretable by a human.
- *Explicit specification:* 'Specification' refers to the idea of structurally describing or identifying something in a precise manner. 'Explicit' implies the meaning of the thing must be clearly defined using formal logic in terms of its concepts, properties, and relations.
- *Shared:* The specification must be agreed upon by a larger consensus. An ontology should also be flexible and reusable across different knowledge-based systems.
- *Conceptualization:* An abstract, simplified view of the world containing the entities that are presumed to exist in some domain.

To summarise, an ontology is a knowledge model that defines a set of concepts of a particular domain, and all the relationships that link those concepts. The ontology must represent the knowledge in a structured format that is both human-understandable and machine-interpretable, typically expressed in a logic-based language. The knowledge is defined precisely, unambiguously, and explicitly, where the definition is agreed upon by a wider consensus.

2.5.5.1 The Components of an Ontology

There are four core components of any ontology [190], namely:

1. *Individuals:* Individuals refer to the instances or objects existing in a formal system, representing some part of the world being modelled. Returning to the previous example, the physical things surrounding us are individuals of particular entities. Different individuals of the same entity may have distinct characteristics, for instance, two people share common characteristics of being human but can have different names, ages, body parts, etc.

2. *Classes:* Classes represent the concepts or entities in the domain of interest. Each class serves as a schema or blueprint for encompassing all the shared characteristics of a concept. For example, a `Human` class could model the characteristics specific to a human e.g., name, age, and body parts. These classes may form hierarchical structures with other concepts. Formally, if concept C' is a subclass of concept C, then any instance of type C' will also be an instance of type C. For instance, the `Human` class can be declared as a subclass of the `Mammal` class, inheriting all defining features of the mammal class.

   Figure 2.25 is a visual representation of a group of classes in their hierarchical structure. Classes can represent not only physical objects but also include abstract ideas and concepts. In this example, we present the `Animal` class, along with subclasses `Mammal, Human, Dog, Frog, Bird,` and `Duck`.

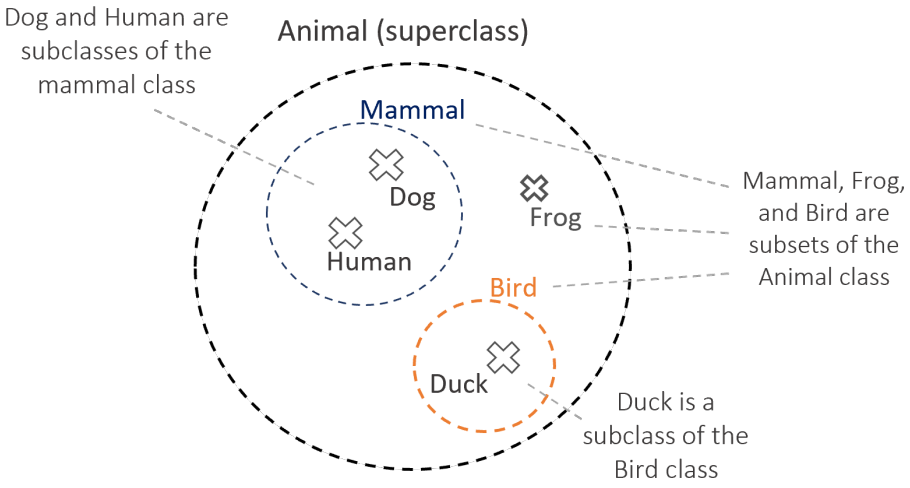


Figure 2.25: An example of classes and subclasses in an ontology.

3. *Properties:* Properties are binary relations representing the characteristics of classes. In graph format, properties function as edges, linking the characteristics to their classes. There are two types of properties: (1) *Data Property* and (2) *Object Property*. A data property is a relation representing a characteristic that links a class with a literal value, while an object property links two classes together. An example of each property is displayed in Example 2.1 and Example 2.2. The definition of *domain* and *range* are described after.

   Example 2.1 To represent the age of a human, the data property `hasAge` can be deployed and linked with a literal value:
   $Human\ (class)\ (domain) \rightarrow hasAge\ (data\ property) \rightarrow integer\ (literal)\ (range)$

   Example 2.2 To represent that humans can have dogs as pets, the object property

`hasPet` can be deployed to link the two classes together:
Human (*class*) (*domain*) → *hasPet* (*object property*) → *Dog* (*class*) (*range*)

`Domain` and `Range` are restrictions on properties that limit the property to only allow specific classes or data types. The class at the start of an RDF tuple is the domain, and the class or literal after the relation is the range. For instance, Example 2.1 defines the domain of the `hasAge` property as `Human`, specifying that only humans have this specific characteristic (age). In reality, all animals have ages, so a more accurate real-world representation would declare the domain for this property as the `Animal` class: *Animal* → *hasAge* → *integer*. Thus, all subclasses of the animal class can accurately use this property.

Additionally, the knowledge representation in Example 2.2 states that the `hasPet` relationship is strictly between a `Human` individual and a `Dog` individual. This implies that any other classes in the ontology that use the `hasPet` property will make the ontology inconsistent. In a real-world scenario, this is factually incorrect as humans can have other pets that are not dogs.

It is not necessary to specify a domain or range for a data property. For instance, having the *domain* of the `hasPet` property as `Human` and leaving the *range* as blank conveys that humans can have any pets of any class in the ontology, even other humans. A more accurate knowledge representation of the real world would require a richer ontology model. For instance, modifying the ontology to include a new class `Pet`, with all possible pet animals modelled as subclasses of this class. Thus, the final representation can be: *Human* → *hasPet* → *Pet*.

4. *Restrictions:* Restrictions are logical constraints that can be applied to classes and properties. For example, the `Human` ontology may restrict the `hasAge` data property to have a $1:1$ ratio, stating that a human may only have one age at a time. Other restrictions can declare constraints such as Male $\cap$ Female $= \emptyset$.

Figure 2.26 depicts a world of discourse containing three individuals: `Sam` and `Harriet`, who are instances of the `Human` class, and `Frank`, who is an instance of the `Dog` class. In this example, `Sam` has three data properties: `hasName, hasAge,` and `hasHobby`, which are literal values. Harriet has two data properties, `hasName` and `hasGender`, and one object property, `hasPet` Frank. Moreover, the Frank individual is instantiated as a type of sausage dog via the `isSpeciesOf` property, and is linked to Harriet via the `isPetOf` property.

2.5.5.2 Ontology: Web Ontology Language

The Web Ontology Language (OWL) is developed by the W3C as the primary language for formalising ontologies, and a means for giving explicit meaning to information on the web. OWL is the successor of *DAML_OIL* web ontology language [203]. OWL was specifically designed as a model that not only presents information to humans in a readable manner but also processes the content of the information, facilitating greater semantic interpretability

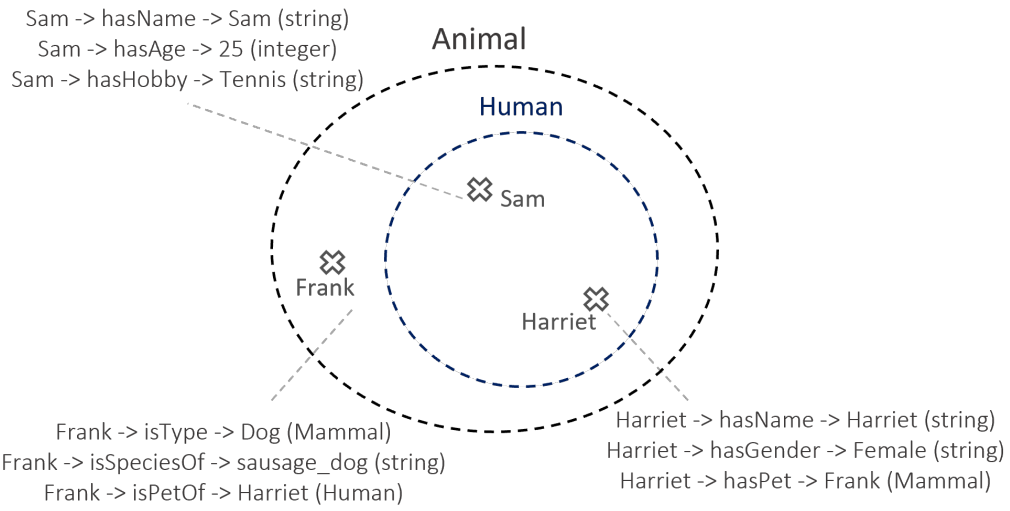


Figure 2.26: An example of individuals and their data properties in an ontology.

[204]. OWL enables users to create custom vocabulary for any domains of interest, e.g., human or cold rolling, while also supporting the use of existing languages such as XML, RDF, and RDF-Schema and their provided functionalities.

Different ontologies are developed for different tasks. Some are used simply to capture and display knowledge in a semantic way, while others may be used for virtual data integration, or ontological rules and reasoning. OWL provides three sub-languages, each of which provides different levels of flexibility and abstraction for versatile ontology purposes. Based on official W3C standards [204], these are:

- *OWL Lite*, which is less expressive than OWL DL and OWL Full, but supports simple classification hierarchy and basic constraints. OWL Lite is aimed primarily at users who seek quick migration paths for thesauri and other taxonomies.

- *OWL DL*, which stands for OWL Description Logics, is targeted towards users who prioritise expressiveness, computational completeness, and decidability. These are terms that originate from process logic [132], where (1) *expressiveness* implies the range and richness of statements that can be expressed within the ontology, (2) *completeness* referring to the validity of the ontology, whether every true statement can be derived or proven, and (3) *decidability* relating to the systematic determination of true of false statements in the ontology. OWL DL supports all OWL Lite features, as well as many other constructs within the formal foundation of OWL that do not affect completeness and decidability. For example, one class may be a subclass of more than one class, but a class cannot be an instance of another class.

- *OWL Full*, has no or very limited restrictions on the language. OWL Full is the most expressive but does not guarantee the completeness or decidability of a system in

return. Predefined definitions in vocabularies such as RDF or RDFS can be augmented and modified to fit new definitions. Although this can be a benefit, it also restricts some reasoning software for complete reasoning.

All ontologies are designed to serve a purpose. Therefore, each version of OWL is catered to fit different specifications. It is important for users to select the corresponding language that best fits the criteria and purpose of their ontology. Meanwhile, every ontology developed using OWL Lite can be considered a legal OWL DL ontology, and every ontology developed using OWL DL is considered a legal OWL Full ontology.

2.5.5.3 Ontology Design Criteria

As mentioned before, formal ontologies are designed. Each representation within the ontology required making design decisions. In our previous example, Figure 2.25, we have seven classes that represent different animals. We have chosen to represent each animal as its own corresponding class, each containing own properties and restrictions. An alternative and more simple route of implementing such knowledge would be by having one class, the `Animal` class, and simply having all other classes as individuals (instances) of the `Animal` superclass. This design and implementation is capable of representing similar knowledge but is less rich. With the chosen approach, we are able to uniquely represent each animal with its own properties and restriction.

Thus, ontologies should be designed to match a criterion. When Gruber defined what an ontology is, he also provided a preliminary set of design criteria that ontologists should consider when designing an ontology [124]. These include:

- *Clarity:* Definitions within an ontology should be communicated effectively with the intended meaning of defined terms while also documented with natural language. When possible, the definition should be stated in logical axioms, and should be independent of social or computational context.

- *Coherence:* the ontology should not contradict or be inconsistent with definitions, especially the definitions captured as logical axioms. There are semantic tools that are able to check whether an ontology is consistent to support coherence.

- *Extendibility:* The ontology is a consensus of shared knowledge. Thereby, it should be designed with the ability for different users to use and modify any terms. The creation of new terms should not directly impose any problems with existing definitions.

- *Minimal encoding bias:* Different knowledge-sharing agents may be developed using other technologies, thus, any encoding bias should be avoided when possible. This implies that conceptualisation should not depend on particular symbol-level encoding.

- *Minimal ontological commitment:* Ontologies should not be *overcommited* in order to support their ability to share knowledge. Commitment is based on consistent use of

vocabulary, and should be minimised, specifying and defining only the terms that are essential to the concept.

With these design criteria in mind, there have been a wide range of ontology development methodologies introduced.

2.5.5.4 Ontology Development Methodologies

The purpose of this section is to describe available methodologies for creating ontologies. Although it is not necessary to follow a specific methodology, many find it beneficial for ensuring a unified and standardised approach to knowledge representation, especially when multiple contributors are involved [279].

Ontologies have seen the introduction of new, structured, and rigorous methods over the years. In this section, we briefly present and survey existing methodologies for developing formal ontologies. A recent survey paper identified 28 different proposed methodologies for knowledge-based systems, which are applicable to ontology development [216].

We introduce three fundamental initiatives, dating between 1994 and 1998, which pioneered the development of early ontologies. These methods are *TOVE*, the *Enterprise Model*, and *METHONOLOGY*, which continue to influence ontology development today, with many recent approaches being modified iterations of these classical methods.

TOVE Methodology Toronto Virtual Enterprise (TOVE) is an earlier ontology development methodology first introduced in 1994 [127], which follows a six-phase formula for capturing knowledge for enterprise systems. These include:

- *Motivating Scenarios:* Create stories and scenarios based on realistic events to capture a set of possible problems that may occur in the domain of the enterprise.
- *Informal Competency Questions:* Using the scenarios, capture the requirements of the ontology as information questions that have concrete answers.
- *Terminology Specification:* Use first-order logic to declare all possible objects, attributes, and relations within the domain of the ontology.
- *Formal Competency Questions:* The declared requirements captured above must be formalised using formally defined terminology.
- *Axiom Specification:* Specify the definition of terms and restrictions of an ontology as Axioms in first-order logic. The axioms must be able to address the competency questions sufficiently.
- *Completeness Theorems:* An evaluation stage which, by defining conditions under solutions of competency questions, assesses the competency of the ontology.

The competency questions form the characterisation of the domain knowledge within an ontology. They are categorised as specific problems that require relevant solutions to be formed. The core advantages of the TOVE methodology are its evaluation-driven nature, and its capability to retain formalisations using first-order logic [168]. The evaluation criteria is provided in the form of completeness theorems, which are beneficial for maintaining and updating an ontology. For example, when extending the ontology with new knowledge, any new knowledge must be compliant with the validity of the completeness theorems. However, the drawbacks include a limited support for versioning, and the mapping of properties in an ontology [263].

The Enterprise Model Approach This methodology is a systematic process firstly introduced in 1995 by Michael Uschold [317], which was significantly revamped in a later publication [314]. The newer method consists of four main stages for constructing an ontology, including:

1. *Identify purpose:* This states the level of formality of the proposed ontology, as well as its intended use. The formality may range from: (1) Highly informal, where expressions are structured informally and captured loosely in natural language, (2) Structured informal, where expressions are restricted to follow a structured form of natural language, reduction ambiguity, (3) Semi-formal, where expressions are articulated in a formally defined language, or (4) Rigorously formal, where terms are meticulously defined with formal semantics, theorems, and proofs.

2. *Identify scope:* This identifies the specification of the ontology, clearly visualising the range of information the ontology must adopt. Unlike TOVE, this is more flexible and can be achieved using motivating scenarios or information competency question.

3. *Formalisation:* Develop the formal definitions and axioms of the specification intensified.

4. *Formal evaluation:* Evaluates the second and third stages of this list using formal knowledge-based system evaluation methods, and refines the formal definitions iteratively until the purpose of the ontology is met.

METHONTOLOGY This methodology was developed by the Ontological Engineering Group at the University of Madrid in 1998 [69]. This methodology focuses on modelling the whole life cycle of an ontology, from specification to maintenance. The methodology is activity-driven and follows seven activities to aid in the development of an ontology. These are:

- *Specification:* This first activity covers the scope of the proposed ontology. The activity focuses on understanding the purpose of the ontology, as well as its intended uses and end-users. This includes identifying the terms to be represented and their characteristics, typically formalised in a document in natural language.

- *Knowledge Acquisition:* This activity occurs in parallel with the specification, and searches for accurate and reliable sources of relevant knowledge. Examples include interviews or questionnaires with experts, as well as analysis of available texts.

- *Conceptualisation:* The terms identified in the specification are categorised into concepts, instances, properties, and relations using informal representation, which are then converted into a semi-formal specification using a set of intermediate representations based on tabular or graph notation.

- *Integration:* To build a uniform platform between ontologies, definitions should be incorporated and defined upon using agreed standards.

- *Implementation:* The implementation activity develops formal models that are represented using ontology languages such as RDF, RDF Schema, OWL, etc.

- *Evaluation:* The evaluation activity uses the validation and verification methods of traditional knowledge-based systems to validate and evaluate the ontology.

- *Documentation:* This final activity consists of collecting documents that are the results of all the other activities.

Similar to TOVE, the METHONTOLOGY method focuses on the maintenance of ontologies. The clear difference is that METHONTOLOGY comprehensively addresses the maintenance of all the stages in the life-cycle of an ontology, whereas TOVE focuses more on formal methods to address more secluded maintenance issues.

Meanwhile, newer methodologies such as NeON [303], typically designed for Waterfall-based software systems, or the *eXtreme design methodology* (XMD) [39] and *Simplified Agile Methodology for Ontology Development* (SAMOD) [247] methodologies used for agile-based software systems have been introduced which innovate the older methods introduced in this section.

Finally, despite the wide range of available development methodologies, ontologies are dynamic and flexible knowledge bases. Thus, it is perfectly acceptable for an ontology designer to create their ontology with their own standards. It is not necessary to follow any predefined methodologies, but highly recommended [279].

2.5.5.5 Ontology Development Tools

A handful of tools have been created to aid the development of formal ontologies. Similar to how software applications can be programmed using different programming languages such as Java, Python, etc., or databases can be developed using SQL, Neo4j, etc., ontologies may also be constructed using different languages. Different tools are available for different languages [297]. This section will primarily focus on tools available for developing ontologies that are programmed using the Web Ontology Language, OWL, described previously.

- *Ontolingua:* Ontolingua is a web-based ontology software developed at Stanford University. This tool was one of the original tools introduced in 1992 by Gruber, which supported the creation, distribution, and collaboration of an ontology [123]. Ontologies were made public via an ontology server which could be distributed globally, achieving the consensus requirement of an ontology.

- *Protégé:* Protégé [228] is one of the more popular available ontology editors [173], which was developed and released in 1999 by Stanford University. It is a free and open source integrated development environment (IDE) developed in Java, which supports most languages such as RDF, RDFS, OWL, etc. Protégé contains organised tabs, which offer the user different elements and functionality, e.g., Entities, Individuals, or Rules tab. Protégé also offers the ability to import any external ontologies into the specified ontology, and provides build-in reasoners that allow for the consistency of knowledge to be checked, as well as the inference of knowledge.

- *The Semantic Web Ontology Editor (Swoop):* Swoop was developed in 2004 at the University of Maryland [169], and is an open source project with global contributors. Swoop is built specifically for OWL-based ontologies, and focuses on providing an ergonomic experience for the user. This software provides functionality for creating and modifying ontologies, but centers on user experience aspects such as having an address bar, history buttons, navigation side bars, bookmarks, and hypertext navigation.

- *Anzo Platform:* The Anzo platform is a development tool released by Cambridge Semantics company in 2011, which focuses on knowledge graph and ontology development. Anzo is a privately licensed tool which contains an ontology editor with available rule engines, web services functionality, and relational database support [297].

- *Graphical Framework for OWL Ontologies (Graffoo):* Graffoo is an open source, graphical framework for developing ontologies introduced in 2013 [93]. The main advantage of using Graffoo is its ability to present classes, properties, and restrictions of an OWL ontology as simple graph diagrams.

For our research, due to its popularity and active userbase, we have chosen to adopt Protégé as the primary tool for developing our ontologies.

2.5.6 Semantic Rules and Reasoning

Semantic Reasoners or *Semantic Rule Engines* are software that provide a mechanism for inferring logical consequence from a set of asserted axioms using a restricted set of first-order formulas [327] [192]. Simply put, a rule engine enables the creation of logical rules, which can be applied to an ontology to obtain new knowledge from existing knowledge [326].

2.5.6.1 The Semantic Web Rule Language

The Semantic Web Rule Language (SWRL) is a rule language introduced by the W3C in 2004, and is acknowledged as the leading language for rules and reasoning by the W3C [144]. SWRL is based on a combination of the OWL Lite and OWL DL sub-languages mentioned previously, in conjunction with the Rule Markup Language (RuleML [41]). These logical rules appear in the form of IF-THEN statement containing an antecedent (body) and consequent (head), i.e., *antecedent* ⇒ *consequent*. These rules state that if the axioms in the antecedent are true, then the knowledge stored in the consequent of the rule can be inferred. All rules are expressed in term of OWL concepts, that is, classes, properties and individuals. Both the antecedent and consequent are formulated as conjunctions of atoms, for instance:

$$a_1 \wedge ... \wedge a_n$$

Variables are indicated by question marks (e.g., ?var). Below are a few simple examples of assertions obtained and adapted from [233]. Providing there is a *Person* ontology that contains the classes: `Person, Man, Adult` and properties `hasParent, hasBrother, hasUncle,` and `hasAge`, we can create the following semantic rules:

Example 2.3 If instance m is a man, then m is also a person:

$$Man(?m) \Rightarrow Person(?m)$$

Example 2.4 If a person has a parent, and that parent has a brother, then that person has an uncle:

$$hasParent(?x, ?y) \wedge hasBrother(?y, ?z) \Rightarrow hasUncle(?x, ?z)$$

SWRL also accepts rules that contain named individuals. Following the same example above, we can create the rule:

Example 2.5 If *John* has a parent, and that parent has a brother, then John has an uncle:

$$hasParent(John, ?y) \wedge hasBrother(?y, ?z) \Rightarrow hasUncle(John, ?z)$$

Additionally, SWRL supports the use of literal values, and provides built-in expressions and formulas for mathematical expressions; for example:

Example 2.6 If a person is 18 or older, they are considered an adult:

$$Person(?p) \wedge hasAge(?p, ?age) \wedge swrlb : greaterThanOrEqual(?age, 18) \Rightarrow Adult(?p)$$

Listing 2.5 displays how SWRL rules are stored in OWL, particularly Example 2.4. This code can be generated automatically from the ontology development tools mentioned previously. In this particular instance, the code was generated by using the *SWRLTab* tab

available in the Protégé IDE [234]. Each rule is declared as an annotation in XML-like structure, labelled as *rule1* in this example. The rule then contains a *Body* and *Head* with their corresponding values captured as atoms.

Code Listing 2.5: The OWL serialisation of Example 2.4.

```
<Annotation>
    <AnnotationProperty abbreviatedIRI="rdfs:label"/>
    <Literal>rule1</Literal>
</Annotation>
<Body>
    <ClassAtom>
        <Class IRI="#Person"/>
        <Variable IRI="#p"/>
    </ClassAtom>
    <ObjectPropertyAtom>
        <ObjectProperty IRI="#hasAge"/>
        <Variable IRI="#p"/>
        <Variable IRI="#age"/>
    </ObjectPropertyAtom>
    <BuiltInAtom IRI="http://www.w3.org/2003/11/swrlb#greaterThanOrEqual">
        <Variable IRI="#age"/>
        <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#integer">18</Literal>
    </BuiltInAtom>
</Body>
<Head>
    <ClassAtom>
        <Class IRI="#adult"/>
        <Variable IRI="#p"/>
    </ClassAtom>
</Head>
```

2.5.6.2 List of Semantic Reasoners or Rule Engines

There are a wide range of semantic reasoners, also known as semantic rule engines, available for inference tasks. There is extensive literature that review and compare rule languages and their rule-based inference engines [265], as well as their performances [188]. In this section, we summarise the most distinguished rule engines available for applying logical inference.

- *Drools:* Drools by JDog, is an open-source engine programmed in Java, and is a hybrid chaining engine, i.e., it can react to changes in data while also providing advanced querying capabilities [256]. Drools is the default rule engine included in the SWRL tab in the Protégé IDE, and therefore, is one of the more popular rule engines. Additionally, it is adopted as SWRL-API's default rule engine. Drools excels at executing smaller datasets but is not well optimised for larger data sets.

- *Jena Inference Engine:* Jena is an open-source Java rule engine, developed by the Apache Software Foundation [164] to assist in the development of Semantic Web applications. The rule reasoner supports a forward-chaining execution strategy, backward-

chaining strategy, or a combination of both [264]. Jena also provides simple built-in functions that supports users with developing their own functions andd rule creation, while also providing functionality to trace and explain proofs.

- *FuXi:* FuXi is a Python-based open-source expert system which includes a rule engine. The rule engine exploits the pattern matching *RETE* algorithm [96] for its ability to process RDF for pattern and object matching. One advantage of adopting FuXi is its ability to export inferred results as a graph, which can be applied to additional graph algorithms for optimised reasoning [265].

- *Jess:* Jess is a standalone rule engine also developed in Java [137]. Jess contains four components: (1) a rule base, (2) live memory, (3) inference engine, and (4) execution engine. However, the rules are required to either be Jess rule language or XML rather, where SWRL formatting is not compatible.

2.5.7 Benefits and Limitations of Ontologies and Knowledge Graphs

In this section, we discuss the advantages and limitations of formal ontologies and knowledge graphs. Some of the information presented below is derived from the 17th IEEE International Conference on Semantic Computing, during which, keynote speaker Eren Kurshan, discussed the applications of knowledge graphs.

The advantages include:

- *Clear representation of specialised knowledge and use of common vocabulary and grammar:* Organisations, people, or software systems may have different interpretations, definitions, or levels of understanding of the same subject matter [315]. There are varying viewpoints and assumptions in natural language, as well as jargon or mismatching definitions of the same subject matter. For example, the concept of a *customer* for one organisation could be a person who purchases an item, while for another company it may simply be a person who enters their store. Ontologies create and provide a domain-specific, shared, and agreed definition of concepts based on a wider consensus. Naturally, the knowledge is domain specific as different terms have different meanings in different contexts. For example, when discussing *toppings*, a pizza ontology may list foods such as cheese or pepperoni, whereas an ice cream ontology may list toppings relevant to ice cream such as whipped cream, chocolate, etc. (hopefully).

- *Knowledge is human-readable and machine-understandable:* The knowledge is captured in RDF format, which enables software and machines to interpret the structure and semantic of the data, while also being readable by a human.

- *Choosing the complexity of a system:* Ontologies are flexible knowledge base systems that can cover any domain of interest. There are examples of smaller ontologies for more basic knowledge representations such as the Pizza Ontology (PO), an introductory, tutorial example ontology in Protégé, or more complex subject matters such

as the Gene Ontology (GO) [67]. GO was developed twenty years ago, and is still actively adopted and managed in the biology community to represent common terms for genes, with over seven million annotations in the ontology [68].

- *Combination of ontologies:* Ontologies can be imported and merged together to share common definitions of concepts. A simple example for this is that we have imported the *Steel Cold Rolling Ontology (SCRO)* into the Core Reference of Steelmaking Ontology (SCRO), so the concepts and relationships in the former are readily available in the latter ontology. Additionally, there are global ontologies that are made publicly available to use, such as the Temporal Time Ontology (TTO). Therefore, when modelling a process that uses temporal time in the SCRO ontology, there is no need to create our own definition.

- *Semantic data access and data integration:* Companies in many domains generate vast amounts of data that are typically stored in data silos that are not typically inter-connected. Despite the data being semantically related, i.e., contain the same meta-data (e.g., `roll_id`) in different SQL tables (e.g., `Roll_Grinding` and `Roll_Trip`), a company would have to hire an expert to physically integrate the data. Meanwhile, an ontology would automatically link the semantically-related meta-data as the data would be in RDF format. Thus, when querying a specific `roll_id`, both the roll grinding and roll trip data can be virtually accessed through the knowledge graph [335].

- *Logical rules and reasoning:* Ontologies contain rule engines that permit inference tasks to obtain new knowledge from existing knowledge as explained previously.

- *Explainability:* Ontologies can provide embeddings within a knowledge graph, which can be leveraged to provide explanations in the context of machine learning.

The limitations of ontologies and knowledge graphs in the present day are as follows:

- *Temporal relationships and dynamic data:* Data captured within a knowledge graph may be dynamic whereas knowledge graphs are static. An update in information will require an update in the knowledge graph, e.g., Joe Biden is the current US president but this will inevitably change.

- *Noisy, unclear, and uncertain data:* When populating an ontology with individuals, the knowledge graph constructed struggles with capturing and filtering fuzzy relationships and data, i.e., the uncertainty and imprecision of information.

- *Implicit data:* Ontologies excel at capturing explicit knowledge but suffer capturing implicit knowledge. For example, it is not difficult to define that $Da\ Vinci \rightarrow painted \rightarrow Mona\ Lisa$. However, it is difficult to precisely describe the implicit knowledge behind such a concepts, e.g., does painting imply he painted every brush stroke?

- *Conditional relationships:* Within a knowledge graph, it is difficult to create conditional relationships, e.g., a person is considered hungry if they haven't eaten, or whether someone isn't eating because of an illness or diet.

- *Complex and probabilistic relationships:* Within a knowledge graph, it is easy to say $Smoking \rightarrow causes \rightarrow Cancer$, and $Cancer \rightarrow kills \rightarrow Person$.. Although it is true that smoking causes cancer, it is also not true that everyone who smokes has cancer, or everyone who has cancer smokes. Capturing these probabilistic relations are challenging.

- *Evolutionary relationships:* Data can evolve. For example, the stats of COVID were rapidly changing every week.

Presently, there is much research on overcoming these limitations. For instance, a recent paper titled *Saga: A platform for Continuous Construction and Serving of Knowledge at Scale* [149], published in 2022, demonstrated a novel methodology of acquiring ground facts about every day things and automatically update the knowledge graph, overcoming the evolutionary relations limitation.

2.5.8 Summary

To summarise, the Semantic-web started off as a vision, developed by the W3C, which extends the World Wide Web by adding a semantic layer of meaning to information on the web. With this vision, software agents and machines are able to understand and comprehend information, making it easier to automate processes to collect and utilise this information. The main contributions are the introduction of languages such as RDF and OWL, which enable standardised representation of data in graph format. These languages are able to capture meta-data of processes, such as the relationship between entities and concepts, with the introduction of ontologies. Ontologies provide a shared vocabulary of a particular domain, which can be expanded and combined with other ontologies to develop greater knowledge graphs for knowledge representation and reasoning mechanisms.

2.6 Data-driven Statistical AI

Many typical computer problems can be solved by procedural algorithms such as the common task of sorting an integer array. These algorithms operate on defined inputs, e.g., a list of integers, and follow a set of computerised instructions to result a new output. Meanwhile, certain tasks are substantially more challenging for procedural algorithms to compute, and require a computer to "learn" specialised knowledge to perform the task [6].

For instance, in image classification tasks, where the model aims to determine the contents of an image, it is necessary for the machine to comprehend the features and concepts within the image to make an accurate prediction. To illustrate this further, in facial recognition, even with inputs (e.g., images of faces) and expected output (Boolean value of true or

false), the computer must first grasp the features indicative of a face to make any decisions, such as eyes, nose, and mouth. Hence, it becomes necessary to precisely define the features characterising a face, and leverage statistical methods to replicate the cognitive capability of accurate facial recognition. This forms the basis for classification tasks in Machine Learning.

2.6.1 Machine Learning

Machine Learning (ML) refers to the development of algorithms that learn exclusively from data. While symbolic AI relies on expert knowledge and contextual information, ML accomplishes tasks by exploiting patterns within the data.

Drawing an analogy from an anthropological perspective, humans naturally refine their problem-solving skills over time. For example, Jean Piaget's theory of assimilation and accommodation states that humans learn through trial and error, devising new approaches to overcome challenges after experiencing failure [38]. In contrast, unlike traditional computer tasks that do not consider past outcomes to enhance their behaviour, ML confronts these limitations.

The idea of computers learning abstract concepts has been present since the 1950s, including a paper titled "Computing Machinery and Intelligence," in 1950 by Alan Turing. In this paper, he posed the question "Can machines think?" [312]. During this period, the question seemed preposterous, yet Turing was able to indirectly spark an interest by comparing the notion of machine thinking to a game, encouraging people to think differently and not jump to conclusion. In this paper, Turing introduced the Turing Test, an alternative version of the Imitation Game. The Imitation Game involves three participants, with one acting as an interrogator whose goal is to distinguish between the other two anonymous players. Player one succeeds if the interrogator correctly identifies them, while player two succeeds if the interrogator makes an incorrect identification. Turing suggested replacing one of the two players with an intelligent machine, challenging the interrogator to figure out which player is human through a series of questions.

In section seven of his 1950 paper, Turing transitioned from the task being a philosophical question into addressing possible approaches, suggesting three solutions to achieving machine intelligence: (1) AI by programming, (2) AI by machine learning, or (3) AI using logic and knowledge. This inspired and motivation interest in the concept of machine learning, and artificial intelligence as a whole [218].

One early example of machine learning is the development of MARK I, a machine created by the RAND Corporation in 1951, which simulated a neural network, with the goal of learning from experiences [176]. Another notable example is a machine developed by A. Samuel in 1952, which was able to improve its performance in the game of checkers, by increasing the number of games played [277]. While these early demonstrations of machine learning showed promise within the field, they often faced challenges due to technical limitations. However, it is only in recent times, with the growth of computational power, increased data availability, and evolution of new learning models, that machine learning

has made a substantial impact, and achieved breakthroughs in various scientific domains worldwide [15].

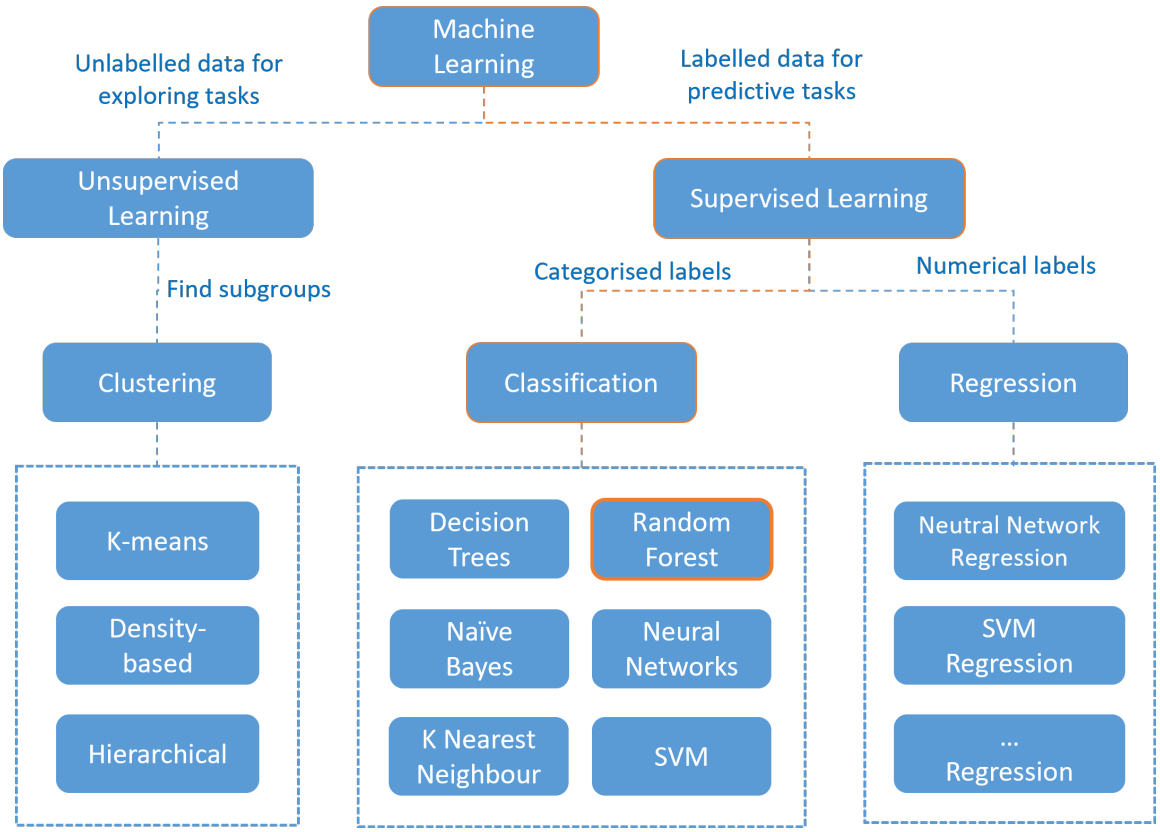


Figure 2.27: An overview of machine learning models based on [15].

In modern times, various machine learning models have been devised to address a wide range of different tasks, each offering different benefits and limitations. Figure 2.27 displays the hierarchical structure of some available machine learning methods.

These ML can be broadly categorised into two groups: (1) Supervised learning, where the model is provided with both input values and corresponding expected output, and (2) Unsupervised learning, where the model only has access to the input values [15]. Both categories involve the utilisation of external information, such as attribute values and metadata, as input for the algorithm. However, in supervised learning, access extends to output values, which represent specific labels for the class attribute. This signifies that the model already possesses knowledge about the structure of the data, and the objective is to assign new data to the correct classes.

In contrast, unsupervised learning models lack access to this labelled information, compelling them to discover inherent structures within the data by autonomously forming their own classes.

To conclude, machine learning models exploit patterns in data to accomplish tasks, which can be applied in maintenance use cases. This approach differs significantly from traditional symbolic AI, which focuses on constructing a knowledge base to capture the context behind processes. In ML models, contextual information is not captured, where the output is often difficult to interpret, raising the questions related to the trust and safeness of the models [258].

In the following sections, we briefly introduce a list of machine learning models. This includes Linear Regression, Bayesian Networks, Support Vector Machines, K-Nearest Neighbour, Neural Networks, Decision Trees, and Random Forests. In later chapters we illustrate the creation of random forests for predictive maintenance tasks.

2.6.1.1 Regression: Linear Regression

Regression analysis refers to statistical machine learning techniques for investigating and modelling relationships between entities [214]. Regression tasks can be applied across various fields and are used for forecasting, prediction tasks, as well determining relations between a dependent variable and a collection of independent variables or features [200].

One example of regression is Linear Regression. Linear regression is one of the first and more simple machine learning models, which employs a mathematical approach to perform predictive analysis. This is achieved by modelling the relationships between a pre-defined dependent variable and independent variables using a simple linear equation. The base equation with a single independent variable, known as the Simple Linear Regression, can be defined as $Y = \beta_0 + \beta_1 X + \varepsilon$, where Y is the dependant variable, and X is the independent variable, $\beta 0$ is the intercept, and $\beta_1 X$ is the coefficient that determines the slope of the line [1]. More complex versions such as Multivariate Linear Regression or Polynomial Regression expand the equation.

Linear regression was independently developed by Sir Francis Galton (a cousin of Charles Darwin) in the late 19^{th} century [301] while studying the relationships between the heights of parents and their offspring. However, Galton did not use the term *regression* – the term was later popularised by Karl Pearson, who used the term in a series of mathematical papers published between 1894 and 1896, mainly in "Contributions to the Mathematical Theory of Evolution" in 1894 [242].

The main advantages of applying linear regression lie in its interpretability and the clear representation of interpretations of coefficients [183]. It simplifies the understanding of relationships between the independent and dependent variables, allowing the relations be visualised on a plot graph. Additionally, linear regression is a straightforward model, which is advantageous when seeking simplicity in a model. It can be implemented as a baseline model to assess whether a more complex approach is needed. On the other hand, the limitations of linear regression include its lack of direct suitability for tasks with multiple classes. Additionally, linear regression assumes a linear relationship between the processed variables. In the occurrence where the relationship is non-linear, the model may underperform.

2.6.1.2 Bayesian Networks

Bayesian Networks (BN) are probabilistic graphical models designed to illustrate probabilistic relationships among variables in a directed acyclic graph (DAG) [295]. In this framework, the nodes of the graph correspond to propositional variables of interest, while the edges represent the direct dependencies between these variables [241].

The term "Bayesian Networks" was introduced by Judea Pearl in the 1980s [22], defining it as a formalism for reasoning under uncertainly. Pearl illustrates this concept by demonstrating how a BN can model relationships among medical diagnosis symptoms while encoding conditional independence assertions in the DAG [240]. In his example, the DAG contained variables such as "Flu", "Fever" and "Positive Test Result"as nodes, with directed edges indicating the probabilistic dependencies among these variables. In this scenario, the observation of "Fever" removes the need for a direct link between the other two nodes.

Notable advantages of this model include its inherent ability to interpret problems in terms of structural relationships between predictors, and ability to assess the probability of an event occurring in relation to a previously observed event [107]. BNs are preferred over other models for dealing with data characterised by uncertainty and probabilistic relationships. However, the key disadvantage is that the performance tends to decrease with an increase in data volume [295]. Additionally, the model is not compatible with dimensional data, posing a limitation in certain analytical contexts.

2.6.1.3 Support Vector Machines

A Support Vector Machine (SVM) is a binary supervised ML model utilised for both classification and regression tasks [206]. In the context of classification, SVM can be utilised to categorise data into two distinct groups, for example, determining whether an email is spam or not. This classification is achieved by identifying the optimal line or hyperplane that effectively separates the data into the different classes. SVM creates a margin i.e., a small gap, between the line or hyperplane and the nearest data points for each category. The data points closest to the line, known as support vectors, are leveraged to determine the optimal placement of separation boundary [45]. To handle more intricate shapes and non-linear relations, SVMs use Kernels– functions that transform data points into higher-dimensional spaces. This flexibility improves the capability of the model, not restricting the hyperplane to remain as straight lines.

SVMs were firstly introduced in a seminal paper titled "Support-Vector Networks" published in 1995 [70] through the combined contributions of Vladimir Vapnik and Corinna Cortes. SVMs are capable of handling complex functions and can employ different suitable kernel functions for different tasks. They can provide higher accuracy compared to other models, where the accuracy and performance are not affected with more features. However, SVMs are typically complex, and knowledge of different kernels is necessary. The performance is not optimal with larger data sets as the training time increases greatly [183, 295].

2.6.1.4 K-Nearest Neighbour

K-Nearest Neighbour (KNN) is a supervised machine learning method extensively applied in pattern recognition and classification tasks [217]. KNN operates as a straightforward classifier, which identifies the nearest neighbours to a given query, and utilises their characteristics to assign a class to the query itself [72]. For example, if something walks, looks, and quacks like a duck, then the likelihood is that it belongs to the duck category. [170]. K, in this context, refers to the quantity of nearby neighbours to the query. There are three distance metrics available, that can be applied to measure the similarity between data points. This includes the Euclidean, Manhattan, or Minkowski distances [296]. The determination of the class assignment is obtained by selecting the most prevalent class between all the neighbours– a process known as a decision rule.

The advantages of employing KNNs lie in its transparency and simplicity of implementation and validation [72]. As a non-parametric classification algorithm, KNNs are well-equipped for scenarios involving multi-modal classes and applications where objects may belong to more than one class. However, some disadvantages include the algorithm's runtime nature, making it prone to performance issues with larger training data sets. Also, the performance relies on selecting the optimal value of the parameter K, which often requires the need for computational expensive methods such as cross-validation to obtain such a value [295].

2.6.1.5 Neural Networks

A Neural Network (NN) or Artificial Neural Network is a mathematically intensive machine learning model designed to simulate the cognitive structure and functionalities of a biological brain [180]. The NN is composed of artificial neurons– simple mathematical models that perform functions such as multiplication, summation, or activation. All inputs in the NN model traverse these artificial neurons as layers. The first layer employs the multiplication neuron, assigning weightings to each input. Afterwards, the sum artificial neuron is applied in the next layer, aggregating all weighted inputs and bias. These new values then pass through an activation function, producing a final output.

The primary advantage of NNs lies in their capability to effectively classify complex scenarios when compared to other models in this category. The flexibility of the model allows the incorporation of numerous layers in the NN to cover deeper layers of data and relations. NNs can also employ non-linear mathematical equations to establish weightings between relations between input variables and capture complex relations [343]. Nevertheless, there is a trade off concerning interpretability, as NNs are inherently complex and function as black-box models. This complexity stems from the stacked layer architecture, where all the layers, except the initial input layer, remain concealed, presenting challenges for human interpretability.

2.6.1.6 Decision Trees

Decision Trees (DT) are a versatile and interpretable supervised machine learning technique that is applicable for both classification and regression tasks [261]. In classification, data is collected and segmented into distinct groups based on features, enabling instances to be categorised into different classes. The algorithm uses a divide-and-conquer approach, utilising recursive partitioning of data according to attribute values, resulting in a tree-like structure [220]. Each path within a tree consists of one or more nodes representing decisions based on specific conditions, leading to a leaf node containing a class label. Each path essentially represents a simple rule that assigns a unique categories to each instance.

DTs are advantageous due to their interpretable and transparent format of representing data in a tree-like structure. This format can be easily interpretable by humans while also being understandable for machines. However, they are prone to over fitting, which is the concept of capturing unwanted noise of the data.

2.6.1.7 Random Forests

Random Forests (RF) were introduced as a classifier to address the limitations of individual decision trees by Leo Breiman in 2001 [47]. The RF algorithm constructs an ensemble of decision trees, where each tree is trained using random subsets of the dataset. These decision trees are obtained through bootstrapping sampling, a method developed in 1982 [91]. As a result, each tree is trained on a diverse set of instances, making each decision tree unique.

To determine the final predicted category for each instance, the output from each individual tree is combined. This ensemble technique lowers the variance and enhances model accuracy. Afterwards, the predictions of each individual tree are combined by adopting either a "majority voting" strategy or a "soft voting" strategy to compute the final classification from the DTs. In the majority voting strategy, the final classification is computed by calculating the modal value of all DT predictions, while in the soft voting strategy, the final classification is derived by calculating the average value of all DT predictions.

Random forests offer several advantages over other models. This includes generally higher accuracy as the techniques applied are less prone to overfitting compared to individual decision trees [30]. In addition, the ensemble nature of RFs provide greater robustness and less sensitivity to noise in the data. They can be applied for both classification or regression tasks, making them versatile for different problems and datasets. Furthermore, RFs provide a wide range of techniques to improve model accuracy, including features such as bagging, node splitting, feature importance, and so forth. Taking feature importance as an example, this feature allows enables you to discover which features have the greatest weighting and impact on the output, indicating the contribution of each feature to the model's predictive capabilities. Finally, unlike some of the other models mentioned that suffer from increasing the quantity of features, RFs are not sensitive to scaling the input features.

In contrast, some of the limits of random forests are notable. First, despite being a collection of white-box decision trees, they are still considered of black-box nature. This is

because increasing the quantity of decision trees in the RF also increases the complexity, compromising the interpretability of the model.

2.6.2 Machine Learning in the Context of Industry 4.0

In the context of Industry 4.0 and smart manufacturing, machine learning techniques have been widely adopted, providing applications to optimise, control, troubleshoot, and enhance process operations and automation [74]. For instance, they can play an important role in predictive analytics and predictive maintenance purposes within cyber-physical systems.

However, their success in the context of Industry 4.0 has varied. A recent review has outlined challenges in this domain [74], highlighting issues such as the dynamic nature of operating environment and the necessity for context-aware information, including operational conditions in a production environment [338, 282]. These challenges are particularly notable in the steel manufacturing industry, where a substantial amount of both human knowledge and data-intensive processes intersect. Meanwhile, machine learning models lack the capabilities to capture and include domain knowledge in their data mining processes as the data is not context-aware.

Additionally, one major concern of machine learning models is their lack of interpretability due to their "Black Box" nature. Black box models refer to models that are either too complicated for any human to comprehend, or are proprietary [275]. Frequently, the results produced from such models cannot be traced or explained, regardless of their accuracy. Consequently, this raises concerns about the reliability of machine-generated outputs in the setting of smart manufacturing.

The trade-off between machine performance and interpretability becomes especially critical. Generally, an increase in complexity results in the decrease of interpretability. For instance, simple models such as linear regression are treated as white boxes as they provide transparency in their functioning, but have limited capabilities. Meanwhile, more complex models such as deep neural networks are considerably more complex and can achieve greater performance and accuracy, but are not interpretable.

To address this concern, novel tools have been introduced to assist in the interpretation of black box models, collectively falling under the term "Explainable AI" [337]. Examples of such tools include LIME and SHAPLEY [141]. In essence, these tools employ a second, post-hoc model to explain the workings of the initial black box model. While these models have demonstrated success in explaining black box models, ongoing discussions emphasise the importance of designing and utilising models that are inherently interpretable from the beginning, especially in domains such as criminal justice, healthcare, and computer vision [275]. These interpretable models refer to rule-based systems and knowledge-based systems where important decisions can be tracked, offering an alternative to black box models. Consequently, the development of hybrid models that combine semantic technologies and ML has been proposed to address these challenges.

2.6.3 Summary of ML

In summary, with the advancements of processing powers, data collection tools, etc., machine learning models have made significant advancements in artificial intelligence. ML models have demonstrated many prosperous capabilities and achievements in all sorts of domains and applications such as biology, healthcare, manufacturing etc. In the domain of manufacturing, they can be applied for predictive tasks which play a pivotal role on maintenance and analytic tasks during processes.

There are a diverse range of machine learning models that focus on either classification or regression tasks while some provide the ability to achieve both, i.e., predicting an output based on a set of inputs using either categorised or numerical labels. These ML models can be employed for different tasks. It is important to note that the performance of each ML model depends on the specific characteristics of the dataset and the specifications of the task at hand. Important factors such as computational resources, interpertability requirements, and the nature of the data play a significant part in selecting a ML model.

In this thesis, we have chosen to adopt random forests over other methodologies because of their advantages and few limitations. We explain our reasoning in greater depth in the relevant chapters.

2.7 The Idea of Combining Semantic Reasoning with ML Models

Machine learning and knowledge representation are two distinct approaches for artificial intelligence. Machine learning emphasises the identification of patterns in data for classification and regression tasks, while the knowledge representation methods are more focused on the acquisition and utilisation of knowledge for reasoning.

In recent years, notable progress has been made in machine learning and GPT-like models, but are often faced with non-factuality and non-reliability challenges. Sam Altman, the CEO of OpenAI, the organisation who developed ChatGPT, acknowledges the current limitations, stating that "*It's a mistake to be relying on it for anything important right now. It's a preview of progress; we have lots of work to do on robustness and truthfulness.*" [260].

Concurrently, discussions in AI emphasise the semantic limitations of ML, and the significance of adopting knowledge representation and reasoning mechanisms for specific tasks. Many contributing figures from both AI backgrounds, such as Benjamin Grosof, a major contributor to the SWRL rule language and the description logic programs [121], or Gary Marcus, a prominent figure in deep learning [196], argue that a hybrid approach combining both AI approaches will be necessary for achieving robust AI [197].

The core idea behind these hybrid systems involves extracting knowledge from the data and applying symbolic reasoning to produce interpretable results, overcoming the black-box behaviour of traditional machine learning methods [89, 138]. This has led to a growing movement to integrate these two methods, leveraging the strengths of each approach, ad-

dressing their respective limitations, and enhancing the overall capabilities of intelligent systems [178, 48].

In the domain of steelmaking and smart manufacturing as a whole, many processes are very knowledge-driven, typically requiring a human expert in the centre of operations. As mentioned, ML models are not able to capture this human knowledge and are restricted to only utilising patterns in the data. Thus, there is motivation to investigate this hybrid approach in the context of this thesis.

Sometimes, the combination of machine learning and semantic technologies is referred to as the third wave of AI, with knowledge representation and reasoning constituting the first wave, and ML representing the second wave [106]. However, the term *Hybrid-AI* can be ambiguous and interpreted in various ways, such as expressing the combination of two statistical ML methods. Consequently, a more specific term, *Neuro-symbolic AI*, has emerged to denote the combination of semantic technologies with machine learning [106].

IBM is just one example of an active and growing community in the research and development of neuro-symbol AI. They held the 1st IBM workshop with over 1,500 registrations in January 2022, which increased to over 6,000 registrations by the IBM summer school in August 2022. The second summer school, held in 2023, included talks from two Turing award winners: Leslie Valiant from Harvard University, recognised for his major contributions to the theory of computation and Probably Approximately Correct (PAC) learning [2], and Yoshua Bengio from Université de Montréal, acknowledged for his major contributions in artificial neural networks and deep learning (with over 730,000 citations) [3].

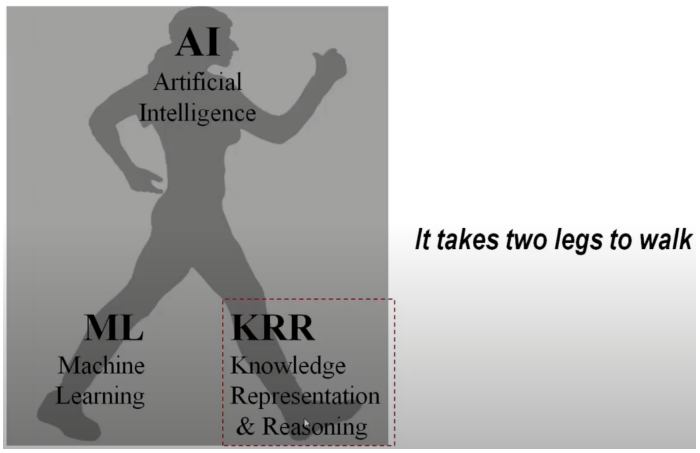


Figure 2.28: Neurosymbolic AI presented by Benjamin Grosof at 2nd IBM Neuro-Symbolic AI Summer School, 2023.

Figure 2.28 displays an illustration by Benjamin Grosof, which was captured during the 2nd IBM Neuro-Symbolic AI Summer School in 2023 [4], stating that neuro-symbolic AI is

[2] https://scholar.google.com/citations?hl=en&user=H509xdsAAAAJ
[3] https://scholar.google.com/citations?user=kukA0LcAAAAJ
[4] https://neurosymbolic.github.io/nsss2023/

necessary to advance the currently state of AI.

However, this research area is still in its infancy. There are many overheads associated with incorporating semantics into machine learning, particularly in the context of predictive maintenance and real-time processing. For instance, traversing semantic structures during inference adds latency, making real-time responses more computationally demanding and challenging [129]. Also, semantic models need to scale in order to handle large data sets, which requires more hand-crafted domain knowledge, increased complexity, and higher computational costs [129].

Despite these overheads, semantic models provide valuable contextual information to data and improve interpretability, which is necessary for interoperability within the Industry 4.0 setting.

Because of the growing interest and potential of neuro-symbolic AI, this thesis explores the integration of random forest classification, a machine learning technique, with semantic technologies. We focus on developing a hybrid method that will assist steel operators with critical decision-making tasks. This research direction aims to bring the strengths of both AI approaches, presenting a solution with several novel contributions.

Part II

Contributions

CHAPTER 3

Semantic-Web Asset Administration Shells Survey

Contents

The Asset Administration Shell (AAS) is a fundamental concept in the Reference Architecture Model for Industry 4.0 (RAMI 4.0), that provides a virtual and digital representation of all information and functions of a physical asset in a manufacturing environment. Recently, Semantic AASs have emerged that add knowledge representation formalisms such as RDF or OWL, to enhance the digital representation of physical assets. In this chapter, we provide a comprehensive survey of the scientific contributions to Semantic AASs that model the information and communication layers within RAMI 4.0 and summarise their structure, communication, functionalities, and use cases. We also highlight the challenges of future development of Semantic AASs.

This chapter was a manuscript that was initially published in 2020 in [27]. We have modified the manuscript to include up-to-date findings.

3.1 Introduction

Digital transformation in manufacturing is enabling a shift in paradigm towards smart manufacturing, which makes use of new technologies and concepts. The advancements in Cyber-Physical Systems (CPS), Industrial Internet of Things (IIoT), Cloud Computing, Artificial Intelligence (AI), and other key enabling technologies, have led to a vision that machines, processes, and products are connected via intelligent networks that utilise information and communication technologies. This vision is called the fourth industrial revolution, commonly referred to as *Industry 4.0* [145]. To realise this paradigm shift, there are many initiatives developing architecture models to support Industry 4.0. The Reference Architecture Model for Industry 4.0 (RAMI 4.0), developed by Plattform Industrie 4.0 and other associations including ZVEI and VDMA [2], includes the fundamental concept of an *Asset Administration Shell (AAS)*, serving as the virtual and digital representation of all information and functions of a physical asset in a manufacturing environment. Within an AAS, big industrial data is stored in a strict format through one or more sub-models to enable real-time condition monitoring of machines and machine tools.

As the manufacturing domain is highly data and knowledge-intensive, uniform knowledge representation of physical resources and seamless integration of heterogeneous data for analytic tasks are needed as the basis to automate decision-making processes for production systems. To address these challenges, knowledge-driven manufacturing methods have received significant attention in recent years. Normally, these methods involve knowledge models during decision making processes in order to provide rich data semantics throughout the manufacturing activities [65]. This trend has enabled the adoption of *Semantic Asset Administration Shells*. A Semantic AAS uses knowledge representation formalisms such as Resource Description Framework (RDF) and Web Ontology Language (OWL) to create digital representations of physical assets [118]. Semantic AASs enable interoperable communication among machines, and compatibility with diverse digital frameworks and architecture.

In this chapter, we provide a comprehensive survey on semantic-based AASs. The aim of the survey is to provide a vision and outlook on the critical and cutting-edge technologies used within Semantic AASs.

The rest of the chapter is structured as follows. First, Section 3.2 gives a brief introduction to RAMI 4.0 and Industry 4.0 components. In Section 3.3, we introduce the survey methodologies used in this chapter. Afterwards, in Section 3.4, we present an in-depth construct of semantic-based AAS, by introducing their structure, communication technologies, characteristics, and functionalities. Section 3.5 demonstrates some typical application use cases of semantic AASs. Finally, Section 3.6 concludes the chapter and outlines open challenges within this research field.

3.2 Background

This section recaps the RAMI 4.0 model, providing a brief description of relevant topics and concepts coined within Industry 4.0.

3.2.1 Reference Architecture Model for Industry 4.0

To recap from the background section, the Reference Architecture Model for Industry 4.0 (RAMI 4.0) is a 3D model frame that highlights the most important and key aspects of achieving Industry 4.0 [2]. The model uses a divide-and-conquer approach to tackle the complex technical to business processes and architecture of Industry 4.0.

Firstly, the *Layers* axis, represents the six processes that describe the technical and business perspectives of a product [345]. The asset layer represents the *things* that are in the physical world whereas the communication, information, functional, and business layers represent the asset's form in the digital world. The integration layer is responsible for bridging these two worlds together by reading and storing all digital information of assets, typically through sensors.

The second axis is the *Life-cycle and Value Stream* axis. This axis expresses the lifecycle and associated value stream of products, machinery, factories, and other assets according to the IEC 62890 standard. The purpose of this axis is to record all data of an asset, from its first idea and initial design, all the way until the asset is scrapped.

The final axis, the *Hierarchy Levels* axis, describes the necessary advancements in hierarchy required for achieving Industry 4.0. The core structure of this axis is based on the IEC-62264 (Enterprise-control system integration) and IEC-61512 (Batch control) standards within a manufacturing environment. These two standards cover many sectors of a factory, from process industry to factory automation. The RAMI 4.0 model adds two additional layers:

- *Product:* The product itself is a key aspect of Industry 4.0 and is still part of the model even after it has been manufactured and sold.
- *Connected World:* Connected world portrays cloud storage and collaboration beyond one factory, between other factories, suppliers, customers, and so forth, to achieve greater connectivity than at the factory level.

3.2.2 Industry 4.0 Component

An Industry 4.0 component (I4.0 component) serves as a model, the purpose of which is to hold all data of assets that can describe properties, configuration parameters, and functions of other assets [102].

The British Standards Institution (BSI) group, known for developing standards in Manufacturing and Engineering, defines an I4.0 component as a *globally and uniquely identifiable participant that is capable of communication within an I4.0 system* [2, 50]. This can range

from a module within a machine to a whole production system [118]. In general, an I4.0 component is composed of an *Asset* and an *Asset Administration Shell* as described below:

The **Asset** layer, as previously mentioned, represents all real, physical things within a manufacturing environment that can be connected digitally. Taking cold rolling mills as an example, this will include physical equipment or products such as the rolling mills, the rolls, the coils, and other machinery. This also includes immaterial items such as documents, plans, orders, and even the shift workers involved.

An **Asset Administration Shell (AAS)** serves as a virtual and digital representation of all information and functions of an asset, and acts as an administration interface accessible within an Industry 4.0 network [111]. Communication between assets occurs between their AAS, whereby the AAS provides and moderates controlled access to the data [251]. Additionally, an AAS contains an administration interface where its data may be accessed through external application services, e.g., by calling an Application Programming Interface (API) from an Enterprise Resource Planning (ERP) system. The data itself is stored in a strict format through one or more sub-models e.g., a *security* sub-model would store all the security-related information. The sub-model approach of storing data is expandable and flexible, and storage location is of free choice [322]. This means that assets from different vendors may contain different sub-models where the AAS simply acts as the data manager of the asset [322]. The data set should contain the full lifecycle of the asset, from its early stages of being planned and manufactured until it is scrapped.

The stored data is processed by data-driven methods for understanding manufacturing processes, thus enabling real-time condition monitoring and maintenance of the physical assets. The AAS is the key enabler of transforming legacy industrial components into I4.0 components.

As knowledge-based frameworks, **Semantic AASs** equip traditional AASs with knowledge representation formalisms to allow semantic metadata modelling. Resource Description Framework (RDF) and Web Ontology Language (OWL) are prime examples used for standardising metadata and semantics [78]. Benefiting from the explicit data semantics specified in semantic models, these Semantic AASs enable interoperable machine-to-machine data exchange and communications [118].

3.3 Survey Methodology

3.3.1 Research Questions

This survey chapter will investigate research literature related to Semantic AAS as an I4.0 component. There is much literature on Industry 4.0 and I4.0 components, but the understanding of Semantic AASs is often overlooked. This review aims to provide a clearer and more concrete understanding of this topic.

The work of this survey chapter is guided by answering the following research questions:

- What are the key characteristics of Semantic AASs?

- What sub-domains of manufacturing have been studied in the literature?
- What are the typical application cases?

In section 3.4, we address the first two research questions by reviewing the existing literature on Semantic AASs. In section 3.5, we address the third research question by summarising the application cases of Semantic AASs.

3.3.2 Inclusion & Exclusion Criteria

In this chapter, we focus on the Semantic AASs that model the information and communication layers within RAMI 4.0. These two layers are essential for the digital representation and real-time data analytics of physical assets. Although there are Semantic AASs that address the challenges in other layers, the amount of papers found are almost negligible compared to the two layers mentioned above. Because of this, research papers that focus on other layers are excluded from this study.

During this study, research papers, technical reports, and white papers are considered valuable resources. However, duplicates among these resources are excluded. This survey primarily focuses on English papers only.

3.3.3 Corpus & Web Search Engine

To derive greater understanding of Semantic AASs, an analysis of all relevant publications was carried out. This topic contains strong blends of scientific literature and engineering principle. Because of this, this survey used different bibliographic databases, including Institute of Electrical and Electronics Engineers (IEEE) Xplore for the engineering side, and ACM for the scientific literature side. Finally, Scopus was used which provided a combination of literature found in both IEEE and ACM, as well as other valid sources.

3.3.4 Initial Survey Results: 2017-2020

At the initial stage, the main keywords searched were *Asset Administration Shell*. Scopus provided a result of 61 unique hits from 2017 to 2020; 70.4% of which were conference papers, 26.2% Articles and 3.4% categorised as others. The vast majority of these publications were from IEEE and only a handful from ACM. Very few results were from other bibliographic databases, such as Science Direct and Web of Science. As a result, this chapter includes all relevant results but will primarily focus on the results found from IEEE as well as ACM. These keywords resulted in 37 papers on IEEE Xplore from the years 2017 to 2020. This small number conveys that this field of research is new and upcoming. From 2017 to 2020, the average annual number of papers are nine. In 2017, four papers mentioned "Asset Administration Shell", followed by eight papers in 2018. This value jumped to 16 papers in 2019 and dipped to seven from January to October 2020. Furthermore, there are no papers prior to 2017 which further emphasises that this field is in early stages of research.

In comparison, ACM only contained six papers from 2017 to 2020. This conveys that most research is happening from an engineering point of view.

When expanding the search to include *Semantic AASs, RDF-based AASs, knowledge-based AAS*, we found under ten relevant papers from the above digital libraries. Thus, we extended our search with Google Scholar and other available sources, discovering a total of seventeen relevant papers in total.

3.3.5 Extended Survey Results: 2021-2024

Following the same approach as the initial search, we searched for *Asset Administration Shell* in Scopus, which resulted in 271 documents from 2021 to 2024. 180 papers are published as conference papers, whereas 71 are articles, and 20 are labeled as 'other'. The increase in papers emphasises the growing interest of the topic. Once more, we refined these papers to include semantic methods using the same search: *("semantic" OR "RDF" OR "knowledge-based") AND ("asset administration shells" OR "AAS")*. This search query returned a total of 47 papers from 2021 to 2024. After applying the exclusion criteria and reviewing the abstracts of each paper, we refined the search to prioritise manuscripts with application implementation, expanding the scope of the survey to cover 25 relevant papers.

3.4 Semantic Asset Administration Shells: The State of the Art

This section gives a detailed description of the Semantic AASs that appear within the papers mentioned in the previous section. The papers are categorised according to which layer the Semantic AASs serve within the RAMI 4.0 architecture. In this chapter, we focus on two layers: the Information Layer and the Communication Layer. Most of the reviewed literature contribute to these two layers.

3.4.1 The Information Layer: RDF-based AASs

Normally, Semantic AASs use formal knowledge modelling languages such as RDF and OWL to create digital representations of physical assets. These type of AASs enable industrial devices to communicate and understand each other, for the goal of semantic interoperability.

The first RDF-based data model for AAS was introduced by Grangel-González et al. in 2016 [118]. This chapter proposed to add a semantic layer to AAS, stating the advantages of adopting an RDF-based approach. The developed AAS benefited from the traits of RDF schema as it provided decentralised and extensible global *Identification*, unified data *Integration*, and *Coherence* among new taxonomies, vocabularies, and ontologies. Because of the uniform information representation, existing standards and asset data could easily be integrated and referenced. To access relevant asset information, SPARQL was used as a query language to retrieve data captured by the Semantic AAS. The authors demonstrated the Semantic AAS using a motor controller as a use case. This Semantic AAS was extended

by the same authors later in [117]. Here, they proposed three improvements: 1) A significant extension of the RAMI 4.0 vocabulary for describing sensor data, units of measurement, and product information; 2) An RDF-based vocabulary which incorporated the international standard IEC 62264 that aligned with the RAMI 4.0 vocabulary; 3) A real-world use case on black carbon monitoring in industry. The prominent characteristic of their work is the translation of IEC 62264 standard-based RAMI 4.0 model to an RDF-based uniform vocabulary. This translation provided a common description of I4.0 components using a unified knowledge representation language.

Similarly, Bader et al. introduced Semantic AASs in more detail [14]. In their paper, the authors filled the gap between industrial reference frameworks and semantic description of the physical world. To achieve this goal, the authors mapped the latest AAS data models into RDF format, and used Shapes Constraint Language (SHACL) shapes to enable schema validation. Firstly, when mapping to RDF, they stated that the AAS object is the root of every AAS. Thereby, it could also be the entry for traversing the Semantic AAS graph. To cover the semantics of an asset, the authors created *rdf:type, rdfs:label, rdfs:comment* to cover the class assertions, name, description, and kind attributes of the asset. Secondly, a fundamental requirement of an AAS is to have a unique identifier—the authors declared that it is possible to use the RDF's URI as a possible identifier. Otherwise, custom formats such as International Registration Data Identifiers (IRDI) are allowed. Submodels and SubmodelElements of AAS were realised and modelled using Operations, ReferenceElements, Files, binary objects and Properties classes. *rdf:property* was developed to align the Property class with RDF's graph model. To execute this mapping, the authors used an open-source tool named RMLMapper. They also illustrated how the transformation to the semantic data model was able to decrease the amount of required storage space.

Tantik and Anderl aligned the Plattform Industrie 4.0 AAS guidelines with the World Wide Web Consortium (W3C) specifications in [305]. W3C provides the Object Memory Model (OMM) which contains block-based digital object memories (DOMe) to present highly standardised meta-information. Combining OMM with an integrated component data model produces a new data model: the Component Data Model (CDM). The authors used CDM to present data of nested I4.0 components. Using a robot arm as a use case, they presented an integrated data model that used a central remote maintenance platform (CRMP) as a form of communication between AASs.

Hua and Hein investigated AutomationML from a semantic point of view in [146]. They argued there was a lack of semantic support for automated machine processing, and thereby, used OWL to transform AutomationML data to a formal and declarative semantic representation. They combined Inductive Logic Programming (ILP) techniques with automated reasoning to obtain meaning of system unit classes. To demonstrate this, they used a machine learning example using OCEL and CELOE algorithms to demonstrate concept learning in AutomationML using DL-Learner.

Thuluva et al. dive into the Web of Things (WoT) and semantic web technologies to address cross-domain interoperability problems in Automation Systems (AS) [308]. The WoT standard provides an interface named *Thing Description (TD)* that was developed to

describe an object, as well as its meta-data and interaction. This paper employed SWT in WoT and used TD as the basis to model a WoT-enabled AAS.

Rongen et al. [273] highlight the benefits of employing AAS models in the context of digital twins, and how an RDF-based approach can enhance semantic expressiveness and advanced querying within the digital twin framework. The authors propose two meta-model methods for integrating RDF and AAS: (1) the first method involving the generation of an AAS template based on RDF-based data models, ensuring robust semantic interoperability, and (2) a method involving the direct mapping of RDF classes and relationships to sub-models within an AAS, utilising the domain and range of RDF to serve as search indexes. The paper introduces the MAS4AI project as a use case, which focuses on the development of agent-based digital twins for modular production environments. The methodology leverages RDF and is implemented in collaboration with pilot lines, showcasing the proposed methods in real-world scenarios [293].

3.4.2 The Information Layer: OWL Ontology-based AASs

As another type of semantic models, OWL-based ontologies also play an important role for describing manufacturing entities within a Semantic AAS. In computer science, an ontology is considered as "an explicit specification of a conceptualisation for a domain of interest" [122]. Within this definition, specification refers to an act of describing or identifying something precisely. This requires the concepts and relationships in ontologies to be clearly defined by using formal logic. Since ontologies are developed based on formal logic foundations, they have been pervasively used in industry to ensure the semantic interoperability among different systems and users. In the manufacturing domain, ontologies play a key role in many distributed intelligent systems as they provide a shared, machine-understandable vocabulary for information exchange among dispersed agents [59, 115, 104, 58]. Large ontologies are designed in a modular structure to enhance their re-usability, extendability, and easy maintenance.

To model the concepts and relations within ontologies, the W3C developed a formal ontology language named OWL. OWL is a component of the Semantic Web, that explicitly represents the meaning of terms in vocabularies and the relationships between those terms. The representation of terms and their interrelationships form an ontology. In the following section, we review the existing OWL ontologies and their rule-based extensions that are relevant to Semantic AASs.

The first group of studied ontologies were developed to model *product-related* concepts for manufacturing. Vegetti et al. [320] proposed a PRoduct ONTOlogy (PRONTO) for the domain of Complex Product modelling. With primary focus on product structure, this ontology considered different abstraction levels of product concepts such as *Family* and *Variant*. It also extended the conventional product structure representations (BOMs) with considering composition and decomposition structures of products within a variety of manufacturing environments. However, PRONTO was not capable of referring to the existing international standards related to the modeling of product structure, processes, and features. To address this weakness, Panetto et al. [236] developed ONTO-PDM, which is an

ontological model considered as a facilitator for interoperating all application software that shared information during the physical product lifecycle. The distinctive merit of this ontology was its incorporation of standardisation initiatives, such as International Organisation for Standardisation (ISO) and International Electrotechnical Commission (IEC) standards. This nature of ONTO-PDM allowed the management of heterogeneous information scattered within organisations, by formalising the knowledge related to product technical data [236]. The MASON ontology [185], developed by Lemaignan et al., is another prominent ontology for manufacturing. This OWL ontology was developed to draft a common semantic net in manufacturing. It conceptualised three core concepts: *Entities*, *Operations*, and *Resources*. In more detail, the product information was specified by three sub-concepts: *Geometric Entities*, *Raw Material*, and *Cost Entities*. The representation of product-related concepts were linked to the description of manufacturing processes and resources under the proposed semantic net in manufacturing.

The second group of reviewed ontologies focused on *process-related* concepts for manufacturing. A manufacturing process is a sequence of activities through which the raw materials are assembled, integrated, and transferred into a final product. To model manufacturing processes, Grüninger et al. [126] proposed the Process Specification Language (PSL) Ontology, which is a semantic model to facilitate correct and complete exchange of process information among manufacturing systems. Within the PSL ontology, the authors formalised the concept of *process* in the form of first-order logic theories. This formalisation has been widely used in domain applications such as process modeling, process monitoring, process planning, simulation, project and workflow management [126]. Another process-related OWL ontology is the one developed by Cao et al. [56]. In their work, a domain ontology was developed to formalise essential concepts and relations about condition monitoring. The ontology was structured into three sub-modules, namely the *Manufacturing* module, the *Context* module, and the *Condition Monitoring* module. This ontology was used within a Cyber Physical System to enable real-time predictive maintenance. A case study on a conditional maintenance task of bearings in rotating machinery was performed to evaluate the proposed ontology. This ontology was extended by the same authors in their recent work [57]. The extended new ontology was named *Manufacturing Predictive Maintenance Ontology (MPMO)*. MPMO was used together with Semantic Web Rule Language (SWRL) rules to enable ontology reasoning, for detecting and predicting possible anomalies that may happen within manufacturing processes. The effectiveness and usefulness of MPMO was tested on a real-world data set collected from a semi-conductor manufacturing process.

The third group of ontologies addressed *resource-related* concepts for manufacturing. In general, the concept *Resources* is defined as physical objects that can execute a range of operations during a manufacturing process. Borgo and Leitão [44] formalised the concept of *Resources* by developing a core ontology for manufacturing scheduling and control environments. In their work, a *Resource* was defined as "an entity that can execute a certain range of jobs, when it is available, as long as its capacity is not exceeded" [44]. The core ontology was implemented as part of a multi-agent manufacturing control system by using

the Java Agent Development Framework (JADE) framework. The author concluded that an established foundational ontology plays an important role in handling heterogeneous data generated by manufacturing control applications, especially those built upon distributed approaches such as multi-agent systems. The notion of *Resources* was also studied in the MASON ontology [185], where *Resources* are further classified into four sub-notions: 1) *Machine-tools* (e.g., turning machines, drilling machines, milling machines); 2) *Tools* (e.g., forging die and punch, turning tool, founding pattern and mould); 3) *Human Resource* (e.g., procedure expert, handling operator, programming operator); and 4) *Geographical Resources* (e.g., plants, workshops). The modelling of manufacturing resources was of vital importance for estimating the total cost for manufacturing activities. [147] et al. also developed the Manufacturing Resource Capability Ontology (MaRCO) that describes the capabilities of manufacturing resources, and leverages semantic querying using SPARQL, to retrieve the needed resources in an AAS-based plant model. The authors demonstrate ontology-based capability checking in AAS to improve the semantic interoperability of AAS-based digital twins. The authors implement a use case of a robotic cell, demonstrating how their ontology can be combined with the "Papyrus4Manufacturing" tools to effectively show the capability of checking protocols in a manufacturing use case.

3.4.3 The Communication Layer: Semantically-enhanced OPC-UA

To tackle the Communication layer of RAMI 4.0, a joint working group including OPC Foundation, ZVEI and VDMA have chosen the OPC Unified Architecture (OPC-UA) as the standard for machine to machine communication [250]. OPC-UA is an enhanced, platform-independent, connection-oriented communication protocol based on service-oriented architecture (SOA) developed by the OPC Foundation [341]. The OPC-UA protocol follows the IEC 62541 standards [125] which are derived from the OPC-UA's Core, Access and Utility specifications [134].

Within RAMI 4.0, the Communication Layer provides components for the communication between machines, devices, production lines, and products. To enable this goal, data communication standards are used to offer required communication between sensors, actuators, and smart devices. In Industry 4.0, OPC-UA has emerged as a widely used standard for data exchange and communication in smart factories [131]. Recently, semantic technologies are considered as a solution to annotating data and provide unambiguous data semantics. The provided data semantics are machine-readable information that allows AASs to perform required actions intelligently without human intervention.

When tackling the communication aspects, Grangel-González et al. used RDF, and STO ontology to describe I4.0 communication standards in [116]. They provided building blocks for the implementation of knowledge graphs for Smart Factory Standards to enable mapping and semantic integration. They used existing ontologies such as *MUTO* (for tagging), *FOAF* (to represent agents & linking documents), and *DCTERMS* (for document meta-data and *RAMI* (for vocabulary). Afterwards, they described the main STO classes. Some examples included *sto:Standard, sto:SDO, sto:Domain, sto:isPartOf, sto:relatedTo e.t.c.*. They used VoCol as an integrated environment to view and explore these ontology classes. Using

these classes, the authors provided a semantic description of the OPC Foundation as well as OPC-UA. They also provided a use case for searching standards and their metadata.

Katti et al. [172] focused on the concept of the Semantic Web Services (SWS), well known for allowing machines to connect without human intervention. The authors integrated Semantic Markup for Web Services (OWL-S) concepts into the OPC-UA specification, expanding to the last layer of the automation pyramid, the shop floor. To achieve this, they integrated the edge component *GeSCo*, which also tackled the connectivity and network latency challenges in Cloud MES manufacturing systems. GeSCo communicated with the manufacturing resources through the OPC-UA protocol. This approach added flexibility in generating orchestration plans, to overcome any unpredicted events in production. Their work was further developed in [171] where they introduced *Semantic Annotations (SAWSDL)* concepts for more added benefit. They named this approach *Semantically Annotated OPC-UA (SA-OPC-UA)*. They stated that the underlying enabling technology for SAWSDL was WSDL. WSDL contains extension attributes that enable semantic annotations to describe the syntax of the web services and their operations. They stated that there was no equivalent facility in OPC-UA that could achieve the same goal.

Weiss et al. [330] present a method for representing an OPC UA server as a generic enterprise knowledge graph, resulting in an 'enhanced OPC UA digital twin'. The author highlights the similarities between OPC UA's 'node-and-references' data model and semantic languages such as RDF. Thus, to address the semantic interoperability challenges inherent to OPC UA, the authors propose modelling the OPC UA information models as knowledge graphs. The authors develop a software tool capable of representing an OPC UA server into RDF. However, the tool is designed to support RDF representations only, excluding RDFS and OWL languages. This restriction aims to preserve original semantics and enable users to query the graph without prior knowledge of ontologies. The authors demonstrate the ability to query the AAS digital twin object using SPARQL.

Wang et al. [324] demonstrate one example of incorporating ontologies with OPC UA to achieve greater semantic interoperability in the domain of Water Conservancy Equipment (WCE). WCE faces inherent challenges in achieving semantic interoperability in smart water conservancy due to the vast quantity of generated information, structural heterogeneity, and complex relationships within the data. To address these challenges, the authors develop the Smart Water Conservancy Ontology, specifically designed for representing WCE. This ontology focuses on numerical representation, data semantics, and capturing the relationship within the data. Afterwards, a network information model is constructed to capture the structural aspects of water conservancy, resulting in a tree model that is capable of realising semantic expressions. The authors implement a use case that deploys an OPC UA server to read and collect water pumps and sluices data, proving the feasibility of the WCE ontology model.

3.4.4 The Communication Layer: the Semantic Web of Things

In Industry 4.0, the Internet of Things (IoT) aims to create a network of physical objects that are embedded with smart sensors, actuators, and software. To enable real-time data

analytics, AASs are required to be equipped with low-latency data exchange and communication capabilities. IoT is a promising solution for this task. However, IoT suffers from a lack of interoperability which leads to its weakness in data management in pervasive and heterogeneous environments [286, 60].

One of the earliest contributions to the Semantic Web of Things was proposed by Pfisterer et al. in [248]. In their work, a service infrastructure named SPITFIRE was developed as an architecture of the Semantic Web of Things. SPITFIRE provides uniform vocabularies to integrate descriptions of sensors and physical objects with the linked open data (LOD) cloud, by which the analysis of data is accomplished on the web. It also provides a comprehensive representation and integrated abstractions for physical objects, their high-level states, and how they were linked to sensors. By this, users were able to access the current status of real-world entities. This vision was achieved by embedding a semantic search engine in the architecture which integrated different static and dynamic data sources in a seamless way [248].

Another framework for the Semantic Web of Things was developed by Ruta et al. in [276]. To associate semantic annotations to real-world objects, locations, and events, the authors outlined a novel general framework for the Semantic Web of Things. The framework *ubiquitous Knowledge Base (u-KB)* is based on an evolution of the classical Knowledge Base model [276]. To enable data annotation, domain ontologies were used as conceptual models for a particular domain of interest. The data annotation allows semantic-based dynamic resource dissemination and discovery within a mobile communication network.

In more recent work, Jara et al. analysed the IoT convergence issue by presenting the different architecture levels of the Semantic Web of Things [163]. The focus was addressed on the trends for capillary networks and for cellular networks with standards such as IPSO, ZigBee, OMA, and the oneM2M initiative [163]. The work paved the way for developing a semantic layer for the IoT by giving a comprehensive analysis of each technology. The analysed technologies are mainly the common internet protocols, including IPv6, Hypertext Transfer Protocol (HTTP), Constrained Application Protocol (CoAP), and the Internet Engineering Task Force (IETF) protocols.

The reviewed articles in this section mainly contributed to the information and communication layers of RAMI 4.0. Table 3.1 summarises our survey results regarding the domains, information and communication models, and achieved goals of the reviewed papers. This table highlights the significance of adopting RDF, OWL, and ontologies for the information model requirements, as well as how semantic technologies can be leveraged for the communication layer. This solidifies our research direction towards building steelmaking-related ontologies for improved data integration and semantic interoperability without our chosen domain.

Ref	Domain	Information Model	Communication Model	Achieved Goals
[118]	Automation control	RDF, RDFS, OWL.	OPC-UA	Present the initial concept of Semantic I4.0 Component.
[117]	Industry sensors	RDF, D2RQ, SPARQL	Semantic Sensor Network (SSN)	Translate the IEC 62264 standard into RDF.
[14]	Automation control	RDF, SHACL	RMLMapper	Map the latest AAS data models into RDF.
[305]	Production robotics	Object Memory Model (OMM)	Central remote maintenance platform	Align AAS guidelines with W3C specifications.
[146]	Engineering systems	OWL	AutomationML	Transform AutomationML data to OWL.
[320]	Complex product modelling	Ontology	-	Model product information in different abstraction levels.
[236]	Complex product modelling	Ontology	-	Incorporate international standards in product modelling.
[185]	Smart manufacturing	Ontology	-	Draft a common semantic net for manufacturing.
[126]	Process modelling	PSL	-	Use PSL to enable the exchange of process information.
[56], [57]	Smart manufacturing	Ontology, logic rules	-	Enable ontology-based predictive maintenance.
[44]	Smart manufacturing	Ontology, JADE framework	-	Handle heterogeneous manufacturing data.
[116]	Smart manufacturing	RDF, Standards Ontology (STO)	OPC-UA, AutomationML	Facilitate the structuring, selection, and integration of standards.
[172], [171]	Smart manufacturing	OWL-S, SAWSDL	OPC-UA, GeSCo	Add flexibility in generating production plans.
[248]	Semantic sensor web	SPITFIRE	Linked Open Data (LOD) cloud	Link semantic sensor models to the cloud.

[276]	Semantic web of things	Ontology	ubiquitous Knowledge Base (u-KB)	Link semantic web to a mobile communication network.
[163]	Semantic web of things	Ontology	IPSO, ZigBee, OMA, oneM2M	Pave a way for developing a semantic layer for the IoT.
[273]	Digital twins	RDF	-	Discussed and integrated RDF and AAS meta-models for semantic interoperability.
[147]	Digital twins	RDF, Ontology	-	Implement semantic interoperability using an ontology, transforming AAS-based models into ontology instances, where semantic reasoning can be applied to retrieve the necessary resources.
[330]	Digital twins	RDF	RDF, OPC-UA, knowledge graph	Represent an OPC UA server as a comprehensive enterprise knowledge graph, resulting in an "enhanced OPC UA digital twin."
[324]	Semantic interoperability	RDF, Ontology	OPC-UA, XML	Combine the semantic interoperability benefits of OPC UA and ontologies

Table 3.1: The domains, information and communication models, and achieved goals of the reviewed papers.

3.5 Semantic Asset Administration Shells: Application Cases

In this section, we abstract from technologies and focus on application cases of Semantic AASs.

Grangel et al. demonstrated the semantic implementation of a *Servo Motor Controller* used for automation within an Engineering and Manufacturing domain in [118]. The same authors also semantically described industry sensors in legacy systems in [117]. Within these legacy systems, the authors used the industrial data set *AirProbe* from an SQL dump. This dump contained sensor information such as geospatial locations and measurements of black carbon concentrations, temperature and humidity [117]. From this data, they were

able to query active sensors within a given time interval and display them as geographical coordinates on an interactive map.

Similarly, Pfisterer et al. also used sensors as a use case [248]. However, their application focused on connecting sensors to the internet and the web, as well as searching for specific semantic entities from the sensors. Their application valued and attempted to derive real-world entities and their high-level status (e.g., meeting room that is occupied) over simply displaying the sensor and its raw data (sensor 536 with motion detection at time T). To realise this, appropriate technologies or mechanisms that establish an explicit mapping between sensors and real-life entities were looked into.

Katti et al. [171] made use of a manufacturing production scenario of a shop floor. Their example included a production routing consisting of three unique operations: welding, colour spraying, and quality checking. Firstly, these three tasks published their meta-data to the appropriate server that could send and retrieve routine details from a Cloud Manufacturing Execution System (MES), then back to the server. The server then assigned tasks and acknowledgements to each operation starting with welding.

Bader et al. [14] developed three examples of mapping Asset Administration Shells into Semantic Asset Administration Shells. The first AAS represented a Raspberry Pi (Pi 3B+) that contained three sub-models that stored technical characteristics data, documentation materials (product sheet and usage manual) and asset descriptions, constructed by a total of 52 SubmodelElements. In contrast, the second AAS represented an electronic Automation Controller for automation facilities, constructed by three sub-models with more than 100 SubmodelElements. Finally, the third AAS represented a Multi-Protocol controller; this AAS contained eight submodels and over 150 SubmodelElements. When evaluating the mappings, they discovered that some expressions were not able to be transported from AAS to Semantic AAS as RDF struggled to sufficiently present some constructs e.g., the Property class in[14]. This also occurred as many input entities contained redundant information in their example.

Hua et al. [146] used *DL-learner* framework to adopt the OCEL and CELOE learning algorithms for concept learning in AutomationML. The authors were primarily interested in the accuracy and efficiency of these algorithms and thereby developed three scenarios that follow typical system unit class structures, each of which has unique complexities. The data set (AutomationML data) contained 222 classes, 63 properties and 61 individuals. Their tests derived that CELOE is much slower than OCEL, and not determined by the complexity of the target concept. This was verified when their most simple test case in OCEL was the most computationally intensive.

Thuluva et al. [308] used an embedded micro reasoner to demonstrate the FESTO process on an automation workstation. The micro reasoner ran on resource-constrained devices with Unix/Linux OS and consisted of two main parts: a micro event processing engine and a datalog reasoner. The former managed event rules that were accessible over a RESTful API and were directly deployed on the automation system devices. The latter provided datalog reasoning, and was embedded in the edge device. The edge device itself was embedded on an automation system and acted as a gateway between the automation

system and the cloud. The author's demo included a binary float sensor that included liquid level values on a tank to ensure the tank did not overflow.

Jacoby etl al. [160] integrate AAS with the Apache StreamPipes application to establish a digital twin of steel production. Their use case involves creating a virtual representation of the steel production process, which can provide accurate recommendations of optimal timing of steel ladle repairs. The AAS metamodel was formulated using various serialisation formats, including RDF, XML, and AutomationML to capture the schematics of the data, while OPC UA and Apache StreamPipes were set up to configure the digital twin environment.

Table 3.2 summarises our survey results including the purpose for each application, as well as the physical objects used for each use case.

Ref	Physical Object	Application Purpose
[118]	Festo AG Motor Controller	Semantically describe an I4.0 component and some of its basic relations.
[117]	Industry Sensors in Legacy Systems	Display sensors on a global interactive map based on time stamps.
[14]	Raspberry Pi, Electronic & Multi-protocol Controllers	Map AASs into Semantic AASs.
[305]	Robot Arm	Simulate the adaptation and remote maintenance of a production robot.
[146]	KUKA Robot	Adopt OCEL and CELOE learning algorithms for concept learning.
[308]	Embedded Micro Reasoner	Detect overflow of water in a tank using sensors.
[171]	Shop Floor Production Machines	Welding, color spraying and quality testing processes on shop floor.
[248]	Sensors	Connect, search and display sensors to the web & internet.
[273]	Robotic arm	Provides additional functionality required by the pilot lines specifications in the MAS4AI project.
[147]	Robotic cell	Automate the entire capability checking process on the shop floor.
[324]	Water conservancy equipment	Simulates typical WCE information space based on water pumps and sluices.
[160]	Steel production ladles	Simulates steel production as a digital twin, generating recommendations in a steelmaking use case.

Table 3.2: List of application cases, with physical object and purpose.

3.6 Conclusion and Open Challenges

The Plattform Industrie group developed a reference architecture model named RAMI 4.0 to tackle the ongoing questions of Industry 4.0. This model introduces a new key technology known as the AAS that is the key enabler for providing a unified way of storing and communicating information between components.

From our survey, we have discovered that there is plenty of ongoing research looking at the RAMI 4.0 model but only some contain implementations of AASs. It is clear that there is a lack of formal standardisation when implementing AAS as many papers use different standards and technologies for their AAS implementations. Furthermore, a key fundamental of Industry 4.0 that needs to be addressed is the semantic interoperability between machines.

Many of the AAS implementations mentioned in this survey display ways of storing and communicating data between industrial devices but these devices lack to demonstrate the ability to *understand* the data itself. Thus, the direction towards Semantic-based AAS with the use of RDF, OWL and other semantic approaches has been investigated to achieve greater interoperability. In this chapter, we have demonstrated existing research contributions regarding Semantic AASs, under the framework of RAMI 4.0. The main characteristics, functionalities, frameworks, and use cases have been covered in this survey.

The survey results expose three main open challenges. The first challenge is the lack of standardised information models for Semantic AASs. Most of the reviewed papers proposed fragmented frameworks or models with a focus on a specific sub-domain of manufacturing, while only a few of them addressed the standardisation issue at a high level. For future works, standardised information models need to be developed with referring to formal ontologies and international standards. The second issue is the need for statistical data processing capabilities for Semantic AASs. As semantic technologies mainly use logic formalisms to describe a certain domain of interest, they have an inherent weakness in processing numeric data. To address this issue, statistical methods such as machine learning and big data technologies are required to equip Semantic AASs with strong mathematical data processing functionalities. The third open challenge is the lack of human behaviour modelling within Semantic AASs. Although fundamental machine operations can be executed autonomously without human intervention, many decisions for complex tasks need the involvement of humans. However, only a few of the existing research works have considered human behaviour within a decision-making cycle. In the future, data from social perspectives need to be considered to improve the decision-making of manufacturing systems.

CHAPTER 4

An Ontology for Steel Cold Rolling

Contents

This chapter introduces the Steel Cold Rolling Ontology (SCRO), which models the domain knowledge of the cold rolling processes and activities within a steel plant. A case study is set up that uses real-world cold rolling data sets to validate the performance and functionality of SCRO. This includes using the Ontop framework to deploy virtual knowledge graphs for data access, data integration, data querying, and condition-based maintenance purposes. SCRO is evaluated using OOPS!, the ontology pitfall detection system, as well as feedback from domain experts from Tata Steel.

4.1 Introduction

Presently, large-scale industrial machines follow rigid automation protocols, resulting in the generation of a vast amount of data. This data is often stored in a format that is not understandable by a machine, and stored in data silos that are not often interconnected yet contain data that is semantically related [335].

A fundamental task to enable Industry 4.0 is to enrich data with semantics to make the data interoperable and machine-understandable. In this chapter, we propose applying ontologies for this task in the domain of cold rolling.

As mentioned previously, cold rolling is one of many steel-making processes. Rolling, in general, processes the greatest tonnage of metals than any other metal working technique [271].

Due to the strong forces involved, the work rolls are affected by roll wear, impacting the service life of the rolls and the quality of the product produced [283]. To avoid this, the rolls are refurbished regularly, where the diameter of each roll is marginally reduced to remove the worn surface. One long-term aim of our research is to utilise semantic technologies for improved data interoperability and combine these methods with traditional machine learning models to optimise the life of the work rolls, improving the total tonnage and yield. In addition, accidents and anomalies that occur, such as overloading, spalling, and incorrect grinding operation [266], can be avoided preemptively once achieving better semantic interoperability.

The goal of this chapter is to develop an ontology that focuses on modelling the cold rolling processes that occur during steelmaking. This chapter introduces the Steel Cold Rolling Ontology (SCRO) that acts as a knowledge base for cold rolling processes within a steel manufacturing plant. This includes the relevant systems, facilities, hardware, software, and inventory of a cold rolling mill. To validate and evaluate the usefulness and accuracy of SCRO, we perform a case study that aligns the ontology with real-world data sets of a cold rolling mill provided by Tata Steel[1]. In this case study, we employ the Virtual Knowledge Graphs (VKG) framework to enable the creation of domain-specific knowledge graphs, which we use for data access and querying.

The remainder of the chapter is structured as follows. In Section 4.2, we provide a literature review that focuses on two key topics: ontologies for Industry 4.0, and ontologies for the steel industry. We also introduce our selected design methodology of ontology development. In Section 4.3, we describe SCRO in detail, including its classes and main concepts. This section also highlights the usefulness of the ontology on an application that uses real-world data. In Section 4.4, we discuss the validation of SCRO to ensure that the knowledge is accurate. Finally, we reflect over our work and end with a conclusion and future work in Section 4.5.

4.2 Literature Review

The W3C have developed a formal ontology language named The Web Ontology Language (OWL)[2] to model concepts and relations within ontologies. OWL is a component of SemanticWeb that allows for explicit representations of the meaning of terms in vocabularies and the relationships between those terms. These representations and their interrelations form an ontology. In the following subsections, we review relevant existing OWL ontologies and their rule-based extensions.

[1] `https://www.tatasteeleurope.com/ts/`
[2] `https://www.w3.org/TR/owl2-overview/`

4.2.1 Ontologies for the Steel Industry

In the steel industry, ontologies are used as an effective and intelligent knowledge management tool for conceptual modelling and information integration. Leveraging the strong modelling and reasoning capabilities of ontologies, process knowledge regarding steelmaking is structured and inferred to facilitate decision making.

Developed as a core component of a Big Data Knowledge Management System (BDAKMS), the ontology introduced in [18] is used to model domain knowledge of steelmaking and enhance the usability and interoperability of BDAKMS. The developed ontology is further used together with SWRL [144] rules to infer knowledge regarding the demand of raw materials. In [325], a shared global supply chain ontology is designed to manage the heterogeneous internal and external decision knowledge of steel companies. Similar to the previous literature, semantic rules are also used to perform ontology reasoning. The goal of ontology reasoning is to facilitate the decision making of business strategies of steel companies. In this way, senior managers can use the ontology to retrieve useful implicit decision knowledge such as pricing strategies, partner selection strategies, and product development strategies.

Ontologies are also used for planning and scheduling of steel production. In [86], an ontological approach is proposed for the goal of optimal planning and scheduling. Within the proposed approach, a set of ontologies are integrated to form an ontological framework. A core meta-ontology and different domain specific ontologies for primary steelmaking are integrated with ANSI/ISA-S95 standard to construct the main body of the framework. Another ontology is introduced in [313] to help with the conceptual design of steel structures. During the ontology design phase, required knowledge elements are identified using intelligent agents. The proposed ontology is reused in other projects such as Agent-Based Collaborative Design of Light Industrial Buildings (ADLIB) and Automated Agent Learning (AAL).

4.2.2 Ontology Development Methodology

Over the years, several methodologies have been introduced to support the development and engineering of ontologies. [168] provides a comprehensive survey of different methodologies available for ontology engineering, including: the Toronoto Virtual Enterprise (TOVE), Methontology, and ONtologic Integration Of Naive Sources (ONIONS) as examples. Each methodology follows unique engineering principles and have different benefits and drawbacks.

For the design and development of SCRO, we have concluded to use the *Ontology Designing Patterns* (ODPs) [105] methodology. We have chosen this methodology for its recognition in modelling solution that support re-usability of good design practices and experiences to solve ontology design problems [105]. There are different types of ODPs that cover different problems such as *structural, correspondence, content, reasoning, presentation, and lexico-syntactic*. More specifically, when developing SCRO, we chose to the use *eXtreme Design methodology (XD)* [254] which is an extension of the *Content* ODPs. This methodology is inspired by *eXtreme Programming (XP)*, which is an *agile* methodology in

software engineering. In XP, the client is involved in the development of the product by providing feedback in cyclical iterations [25]. This was necessary when developing SCRO as most of the domain knowledge obtained was provided by involvement and feedback with experts from Tata Steel. Other XD principles include: collaboration and integration, task-oriented design, and test-driven design which are explained in [254].

We conclude that this design approach offered numerous evident advantages for developing ontologies, including: a faster ontology design process, more flexible design choices, improved interoperability, and ontology quality [40].

4.3 SCRO: Steel Cold Rolling Ontology

Most of the domain knowledge mentioned in this section was obtained from a case study with domain experts from Tata Steel at the Port Talbot plant. One report is mentioned in Section 2.2.1 (Use case of Cold Rolling: Cold Rolling at Tata Steel).

SCRO models the fundamental structure and operations of the rolling processes in the case study, which a focus on maintenance. There are other focus points that we have excluded from the coding, such as economics and production rate, which fall out of scope for this chapter. This level of granularity can be added as future direction.

Additionally, Although SCRO is initially designed for the processes and machines at Tata Steel; the ontology could be modified and reused by other steel manufacturers for knowledge modelling of other cold rolling processes.

In this section, we describe SCRO in detail, beginning with the encoding and classes.

4.3.1 Coding

SCRO was developed using the free, open-source ontology editor and framework called Protégé [219]. We used the latest version to date, Protégé 5.5.0, that offers a unique interfaces for creating and maintaining ontologies for intelligent systems. Protégé supports the commonly used ontology language, OWL, which enables us to model concepts, as well as their relations and attributes through classes, object properties, and data properties [12]. Figure 4.1 displays the structure and the architecture of SCRO, whereas Figure 4.2 displays the classes, object properties, and data properties.

4.3.2 Reusing Existing Ontologies

An extensive amount of data within the domain of steel manufacturing is generated and read through sensors. Generally, these sensors run on timestamp data to record the continuous flow of dynamic data. Therefore, we have imported the *Time* ontology created by W3C that supports the use of timestamp data [139]. These are excluded from Figure 4.2 but play an important role in SCRO.

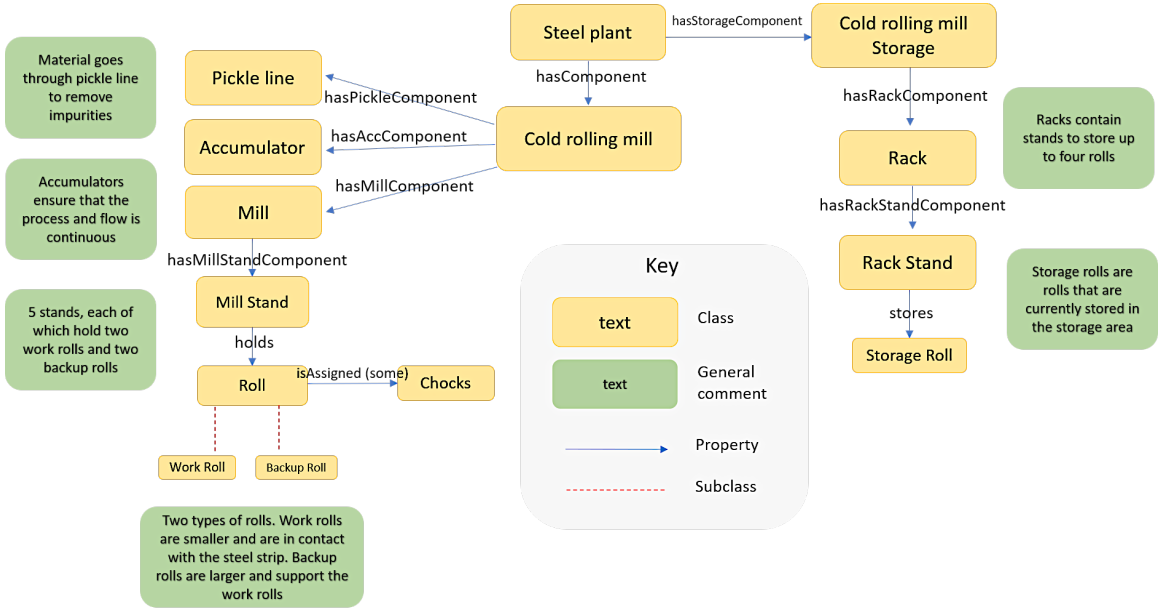


Figure 4.1: The structure of SCRO.

4.3.3 Classes

There are many processes and components on the shop floor that are fundamental for cold rolling, depicted in Figure 2.11 in Section 2.2.1. We create classes for each one respectively. The cold rolling mill processes are divided into three sub-processes: the pickle line, accumulators, and the mill.

Firstly, the process of steelmaking creates undesirable oxidations on the material. To counter this, the material, `entry coil`, undergoes surface treatment on the pickle line. The process of pickling cleanses the entry coil by using acid to eliminate impurities and oxidations, providing a smoother surface. The class ***:Pickle_Line*** denotes this process whereas the superclass ***Pickle_Line_Component*** contains the necessary pickle line components on the shop floor as subclasses; these components are defined in Table 4.1.

Both the pickling and mill processes are continuous and run at different speeds. Often one of these processes is required to stop while the other is still in operation. For example, when introducing a new coil into the pickling process, the pickle line is paused to weld/stitch the new coil while the mill process is still running at a constant speed. An *Accumulator* between these two processes is able to facilitate such activities through movable rolls that are able to control the amount of material in that intermediate section, ensuring the whole cold rolling process to be continuous from beginning to end. The class ***:Accumulator*** denotes this process.

Finally, the material is passed through the mills where its thickness is reduced. The class ***:Mill*** denotes this process whereas the superclass :***Mill Component*** contains the necessary mill components on the shop floor as subclasses; these components are also defined

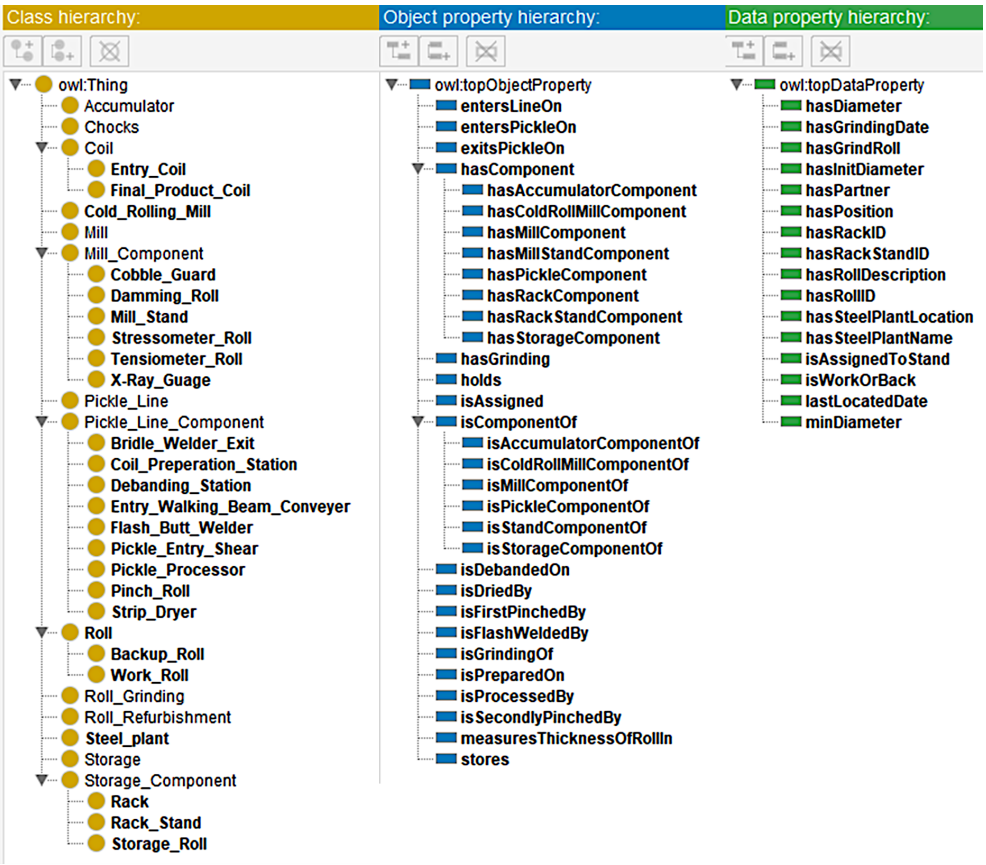


Figure 4.2: The classes, object properties, and data properties of SCRO.

in Table 4.1.

The rolls are fundamental components of the cold rolling process. The rolls are the physical entities that rotate to reduce the thickness of the steel trip. These are denoted by the superclass ***:Roll*** and its two nodes ***:Work Roll*** and ***:Backup Roll***. These rolls are assigned some chocks which allow for rotation within a mill; these chocks are denoted as ***:Chocks*** in the ontology. In addition, we have included ***:Storage Roll*** which are rolls that are out of the mill and are in the storage area. This storage area is denoted by the class ***:Storage***, and the superclass ***:Storage Component*** contains the components of the storage as subclasses.

Finally, the ontology contains other classes such as ***:Steel Plant***, ***:Cold_Rolling_Mill***, ***:Roll Refurbishment*** and ***Roll Grinding*** which are briefly described in Table 4.1. Figure 4.3 displays the hierarchy of all the classes, generated by the Protégé tool. All the class relations in SCRO follow a *is-a* relationship. We have decided on this approach for its simplicity as it provides provides enough granularity for our research goals. There are alternative methods for capturing the domain knowledge using other relationship types, such as, *has-a* or *part-of* compositions, which can always be implemented as future work

to enrich the ontology.

SCRO Classes	Description
Accumulator	Manage the speed of the rolling processes to ensure flow is continuous
Chocks	Attached to rolls and contain bearings that allow rotation
Coil	**Superclass of the material and final product**
Entry_Coil	Denotes the steel strip that enters the cold rolling mill
Final_Product_Coil	The final product sold to customers
Cold_Rolling_Mill	Denotes the shop floor of the cold rolling mill
Mill	Process of the cold rolling mill for reducing steel thickness
Mill_Component	**Superclass of all Mill component**
Cobble_Guard	Component that reduces the chance of producing cobbles
Damming_Roll	Component that restrains the outward flow of coolants
Mill_Stand	Stand that fits two work rolls and two backup rolls
Stressometer_Roll	Measures the flatness of the steel strip
Tensiometer_Roll	Measures the tension of the steel strip
X-Ray_Guage	Measures the thickness of the steel strip
Pickle_Line	Process where the entry coil undergoes surface pickling
Pickle_Line_Component	**Superclass of all Pickle component**
Bridle_Welder_Exit	Equipment that the strip uses to exit the pickle line
Coil_Preparation_Station	Station where the entry coils are entered
Debanding_Station	Station where the entry coils are debanded
Entry_Walking_Beam_Conveyor	Conveyor where entry coils are first placed
Flash_Butt_Welder	Machine that presses and welds the ends of the workpiece
Pickle_Entry_Shear	Machine that cuts rolls to desired size
Pickle_Processor	Machine that processes the coil, minimising coil breakage
Pinch_Roll	Machine that holds and moves the strip
Strip_Dryer	Removes excess water from the strip to prevent rusting
Roll	**Superclass of the rolls at a cold rolling mill**
Backup_Roll	Larger roll that support a work roll during milling
Work_Roll	Smaller roll that rotates to reduce the thickness of steel
Roll_Grinding	Contain previous grinding data of rolls
Roll_Refurbishment	Process where rolls are sent to be refurbished
Steel_Plant	Denotes the whole steel plant
Storage	Inventory where assets (e.g unused rolls) are stored
Storage_Component	**Superclass of the Storage component**
Rack	Contain stands for rolls to be stored
Rack_Stand	Store one storage roll
Storage_Roll	A roll that is not currently being used and stored away

Table 4.1: A list of the classes in SCRO, alongside their description.

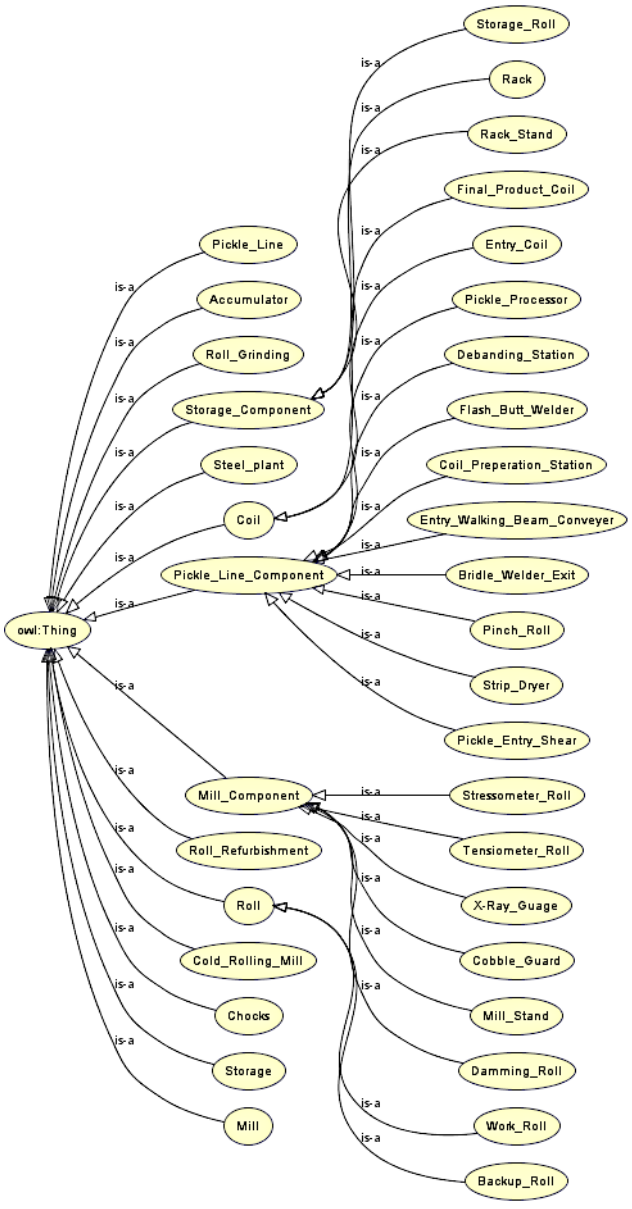


Figure 4.3: The hierarchy of all the classes in SCRO, generated by OWLViz plugin in Protégé.

4.3.4 Object and Data Properties

To semantically describe the properties of the cold rolling processes, it is important that we specify the domain and ranges of each property. The domain and range are constraints that limit the subjects and objects that are allowed for specific relationships. The domain refers to the set of classes in which the property can be applied to as a subject, whereas the range defines the set of classes in which the property can be applied to as an object, e.g., *domain* → *property* → *range*. These properties are presented in Table 4.3.4:

Object Property	**Domain**	**Range**
entersLineOn	Entry_Coil	Entry_Walking_Beam
entersPickleOn	Entry_Coil	Pickle_Entry_Shear
exitsPickleOn	Entry_Coil	Bridle_Welder_Exit
hasComponent	**undefined**	**undefined**
hasAccumaltorComponent	Cold_Rolling_Mill	Accumulator
hasColdRollMillComponent	Steel_plant	Cold_Rolling_Mill
hasMillComponent	Cold_Rolling_Mill	Mill
hasMillStandComponent	Mill	Mill_Stand
hasPickleComponent	Cold_Rolling_Mill	Pickle_Line
hasRackComponent	Storage	Rack
hasRackStandComponent	Rack	Rack_Stand
hasStorageComponent	Steel_Plant	Storage
hasGrinding	Roll	Roll_Grinding
holds	Mill_Stand	Storage_Roll
isAssigned	Roll	Chocks
isComponentOf	**undefined**	**undefined**
isDebandedOn	Entry_Coil	Debanding_station
isDriedBy	Entry_Coil	Strip_Dryer
isFrstPinchedBy	Entry_Coil	Pinch_Roll
isFlashWeldedBy	Entry_Coil	Flash_Butt_Welder
isPreparedOn	Entry_Coil	Coil_Preparation_Station
isProcessedBy	Entry_Coil	Pickle_Processor
MeasuresThicknessOfRollIn	X-Ray_Guage	Mill_Stand
stores	Rack_Stand	Storage_Roll

Table 4.2: A list of the object properties in SCRO with their corresponding domains and ranges.

Similarly, Table 4.3.4 displays the data proprieties in the ontology:

Data Property	**Object**	**Datatype**
hasDiameter	Roll	xsd:double
hasGrindingDate	Time instant	xsd:date

Continued on next page

Data Property	Object	Datatype
hasGrindRoll	Roll_Grinding	xsd:integer
hasInitDiameter	Roll	xsd:double
hasPartner	Roll	xsd:integer
hasPosition	Roll	xsd:string
hasRackID	Rack	xsd:integer
hasStackStandID	Rack_Stand	xsd:integer
hasRollDescription	Storage_Roll	xsd:String
hasRollID	Roll	xsd:integer
hasSteelPlantLocation	Steel_Plant	xsd:String
hasSteelPlantName	Steel_Plant	xsd:String
isAssignedToStand	Roll	xsd:integer
isWorkOrBack	Roll	xsd:string
lastLocatedDate	Time instant	xsd:dateTime
minDiameter	Roll	xsd:double

Table 4.3: A list of the data properties in SCRO with their corresponding object and datatype.

4.3.5 Dataset

We test and evaluate SCRO through a real-world industrial application. Within this industrial application, a collection of real-world data sets have been provided by Tata Steel. These data sets come specifically from the five-stand Tandem Cold Rolling Mill at their Port Talbot plant.

Firstly, static data related to the rolls, roll storage, and roll refurbishment have been collected. These data sets are stored in a database where the values of these rolls are always updated manually from someone at the plant. This data is considerable in quantity and located in different tables within their database. For our research, we focused on three specific tables: the `Roll`, `Roll Grinding` and`Roll Storage` tables. These tables contain many fields that we have chosen not to include in SCRO, following the design criteria to not overflow the ontology. Instead, we limit our interest to core fields such as *RollID* and *diameter*, abstracting from less relevant data. The domain experts from Tata Steel agreed with this approach. Table 4.4 describes the these fields, along with their data types and descriptions.

Secondly, the data sets also contain dynamic data from the cold rolling mills that are read by sensors and stored in a database. These sensors record the condition of rolls in short intervals, thus, creating huge amounts of industrial data. The data includes the chemical composition, temperature, and pressure of the rolls, among others.

Table and fields	Data type	Description
Rolls	**Table**	**Contains static data relevant to the Rolls**
Roll_ID	Integer	Unique identifier of the roll. Primary Key
Diameter	Double	Stores the value of the diameter of the roll.
Position	String	`Top` or `Bottom` to denote position of roll in mill
Partner_ID	Integer	Unique identifier of the roll's partner
Work_Backup	String	Identifier to specify whether a roll is a work or backup roll
Last_Loc_Date_Time	Date	Timestamp of the date when the roll was last located
Last_Stand_ID	Integer	The last stand this roll was placed in
Roll_Grinding	**Table**	**Table that stores the previous grindings of rolls**
Roll_ID	Integer	Non-unique identifier to specify which roll
Diameter	Double	Stores the value of the diameter of the roll
Grind_date	Date	Timestamp of the date when that roll was ground
Stand_ID	Integer	The last stand this roll was placed in
Roll_Storage	**Table**	**Table of rolls that are currently not in use**
Rack_Location	Integer	Non-unique identifier of the location of the racks
Single_Rack_ID	Integer	Unique identifier of the rack
Roll_ID	Integer	Unique identifier of the roll that is stored on a rack
Status_description	String	The status of the roll, i.e., it's a new roll or damaged roll
Actual_Diameter	Double	Stores the value of the diameter of the roll

Table 4.4: The different available data, including their types and descriptions.

Note: these tables are not interconnected but contain fields that are semantically related. For example, `Roll_ID` appears in all three tables. To effectively use the data, data integration is required. However, it can be costly to join, clean, and homogenise the data. To avoid this, as mentioned previously, Virtual Knowledge Graphs (VKG) exploit data virtualisation to create domain-specific knowledge graphs [335], which we employ for data integration and data access. This is achieved by creating graphs on top of relational databases where the data is not physically moved to another database and instead kept and viewed at a virtual level [336].

Virtualisation is achieved by creating an ontology, and linking the data sources to the ontology via `Mappings`. We provide examples of this process in the next subsection. These mappings enable the ability to query data at a virtual level without paying the cost of integration. Numerous applications have been developed to support the VKG approach. Some examples include Mastro [54], Morph [255], and Ontop [336]. For our approach, we have adopted the Ontop framework.

4.3.6 Ontop Framework

The Ontop Framework[3] is an open-source VKG (previously known as Ontology-based Data Access) framework developed by the Free University of Bozen-Bolzano. We have chosen Ontop over the other VKG approaches as Ontop supports all the W3C languages and recommendations including RDF, OWL, SPARQL, R2RML, and SWRL [53]. Additionally, it supports widely used standards including: (1) Ontologies: Ontop supports *OWL 2 QL* ontology language which runs on description logics; (2) Mappings: Ontop supports its own *Ontop Mapping Language* as well as the W3C recommendation *R2RML mapping language*; (3) Data Source: Ontop supports the major commercial and free structured databases such as *MySQL, H2,* and *PostgreSQL*; (4) Querying: Ontop supports the latest version of the *SPARQL* querying language, which includes many features such as aggregation and negation [16].

4.3.7 Mappings

Mappings are created to link ontology classes and properties with data from the relational data sources to produce RDF triples. R2RML is the standard mapping language used in the semantic web [272]. For our mappings, as mentioned above, we used the Ontop mapping language which is fully interoperable with R2RML [53].

Mapping engineering is considered a difficult and time-consuming activity that requires strong knowledge of not only the domain of interest, but also the rigid structure of databases and their schemas. Presently, there are several contributions working towards this direction to automate the process. There are two main approaches to mapping engineering. The first is using Mapping Bootstrappers (MB) which automatically generate a mapping for a data source [335]. These mappings follow a set of rules based on the W3C Direct Mapping specification to generate RDF graphs [289]. Ontop bootstrapper and BootOX [165] are two examples of existing MBs. A benchmark suite named Relational-to-Ontology Data Integration (RODI) [249] has been developed to evaluate and compare MBs. Using an MB has both benefits and drawbacks. The key benefit is that it is fast and automatic; whereas the biggest drawback is that it lacks flexibility when having numerous data sources as the generated vocabulary becomes restricted to data-source specific data. The second approach is to use mapping editors to manually write mappings. For our approach, we manually wrote our mappings using a text editor that is available in the Protégé IDE.

Figure 4.4 shows a mapping between the `Work_Roll` class in SCRO and the `Rolls` table in the SQL database. The bottom half of the figure illustrates the source, in the form of an SQL query that allows us to specify and filter the data we want to map. Like with all SQL queries, we use the `SELECT` clause to select the necessary fields from the database, followed by the `FROM` clause to select the table name. Finally, we use the `WHERE` clause to refine the query. As seen in Figure 4.4, we are interested in the *roll_id, position, diameter, partner_id, work_backup, last_loc_date_time, and last_stand_id* values from the `rolls` table where the *work_backup* field is *'W'* which denotes work rolls. We use the `AND` clause to further refine

[3] `https://ontop-vkg.org//`

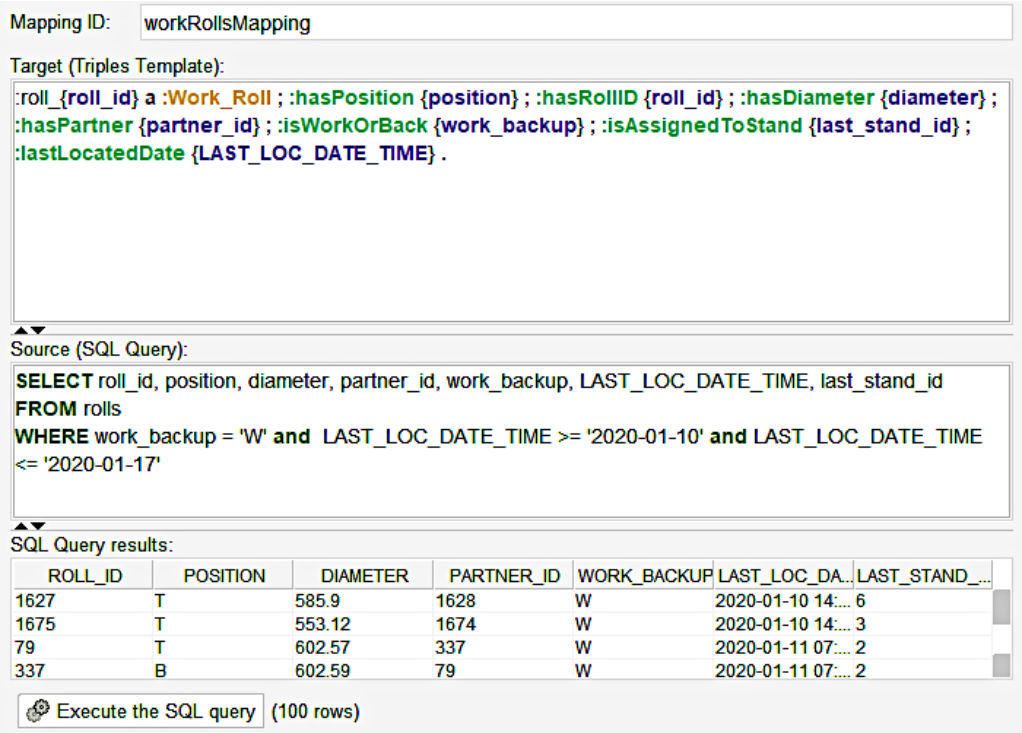


Figure 4.4: An ontop mapping for work rolls.

the query to restrict the `last_loc_date_time` timestamp value to a seven day period. We can then click the "Execute the SQL query" provided by the Ontop Mappings plugin in Protégé to print and verify the results of the query. To conclude, the SQL query returns all work rolls that were last located between the 10^{th}-17^{th} of January 2020.

Secondly, we create a mapping `target` which maps the selected fields from the database onto the classes in SCRO. The target section is written using Turtle-like syntax[4]. The first part ***:roll_{roll_id}*** is a variable name of the individual, and the subject of the RDF triples being generated. Here, we used the primary key `roll_id` from the SQL query to create a unique IRI for each individual roll. For example, the roll with *roll_id* of 500 in the database will be named *roll_500*. The second part **a :Work_Roll** specifies that this individual and RDF triple will be an instance of the *Work_Roll class*, followed by a semi-colon. Note, by using a semi-colon instead of a fullstop, Ontop is able to map numerous fields from the SQL query to the data properties in the ontology without having to specify the initial subject and class each time. The syntax for these mappings are shown in Figure 4.4. For example, **:hasPosition {position}** implies *:hasPosition* is a data property from the ontology where the value of this property is mapped to the {*position*} field from the SQL source.

Similarly, we have a comparable mapping for the backup rolls. The key difference is the **:roll_{roll_id} a :Work_roll** becomes **:roll_{roll_id} a :Backup_roll** and the

[4] `https://www.w3.org/TR/turtle/`

work_backup field in the SQL *WHERE* clause is set to equal *'B'*.

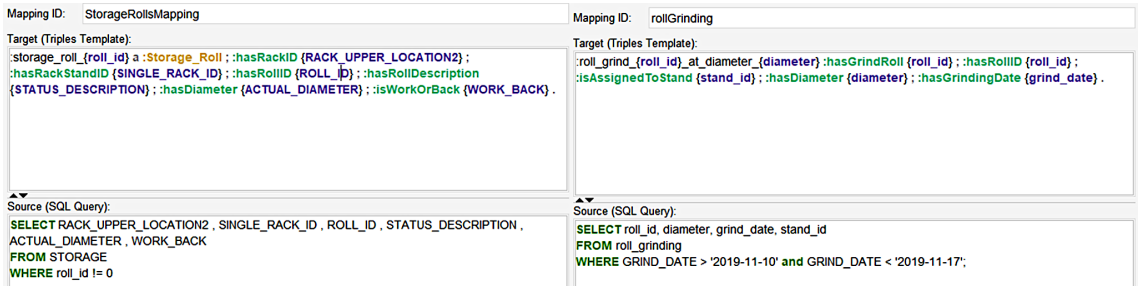


Figure 4.5: Ontop mappings for roll grindings and storage rolls.

Figure 4.5 depicts two other mappings. The mapping on the left manages and links SCRO with the *roll_storage* data set, containing the data properties: *hasRackID, hasRackStandID, hasRollID, hasRollDescription, hasDiameter* and *isWorkOrBack*. The selected data values with *roll_ID* with zero is to be ignored as these are invalid data entries in the data. Meanwhile, the mapping on the right manages historical grinding values of rolls from the *roll_grinding* data set, containing the data properties *hasGrindRoll, hasRollID, isAssignedToStand, hasDiameter,* and *hasGrindingDate*.

Once these queries have been created, Protégé provides the ability to materialise these mappings to generate RDF tuples for each data entry, populating a knowledge graph. Once this data is in this format, we are able to query the data.

4.3.8 Querying on Knowledge Graphs

We use SPARQL[5] to query the data for condition-based maintenance of rolls and information retrieval purposes. SPARQL is a well known querying language within the semantic web. The difference between SPARQL and SQL is that SQL queries on structured databases, whereas SPARQL queries on RDF triples [272]. As described above, the RDF triples are generated by the Ontop mappings that are depicted in Figure 4.4 and Figure 4.5, which enable us to query the data with SPARQL.

There are applications being developed to aid the assistance of SPARQL query formulation. An example includes the OptiqueVQS tool [299], which provides an interactive interface that generates components to build SPARQL queries. However, the Protégé software we used to create the SCRO ontology also provides a built-in text editor to create SPARQL queries, which we opted to use for easier convenience. The queries we demonstrate below did not require the assistance for external software. However, when writing more complex queries in the future, end users may opt to use OptiqueVQS or other tools. Below are some queries that we developed to query the knowledge graph.

Listing 4.1 is a query that outputs the diameter values that have three or more rolls that share that diameter. Rolls in operation are always paired with other rolls that have

[5] `https://www.w3.org/TR/sparql11-overview/`

the same diameter value, thus, each diameter should appear twice in the rolls data set. In contrast, rolls from the storage data set have yet to be paired. By limiting our search to only return diameter values that appear three or more times, this type of query can be used to discover rolls that have matching diameter values to other rolls from either data set. This is important because if a roll gets damaged, we can use this query to see if other rolls with the same diameter in both the storage data set and roll data set exist, and quickly replace the damaged roll for maintenance purposes.

Code Listing 4.1: Diameter values which appear in more than two rolls.

```
PREFIX : http://www.semanticweb.org/sadee/ontologies/2021/1/SCRO#
PREFIX time: http://www.w3.org/2006/time#

SELECT ?diameter
WHERE {
    ?roll :hasDiameter ?diameter .
    MINUS {
    ?roll :hasGrindRoll ?grind .
    }
}
group by ?diameter
having (count(?diameter) > 2)
```

To construct this query, it is a requirement to specify the `prefixes` of the ontologies we wish to use. As shown in the first two lines of Listing 4.1, and for most of our queries, we have declared two prefixes: an empty prefix to denote SCRO and a `time` prefix to denote the time ontology that we have imported.

The main body of a SPARQL query is structured similarly to an SQL query. We start the query with the `Select` clause to select the fields we are interested in. In SQL, this would be one or more fields from a specific table. In SPARQL, we simply enter a variable name that will hold our results. Note that all variables begin with a question mark. As shown in Listing 4.1, we have chosen to select a variable called `?diameter` to denote the result of the SPARQL query will be related to the diametric value of the rolls. Then, we use the `WHERE` clause to condition our results. In our query, we specify that we are interested in the RDF triples whose subject contain the property `:hasDiameter`, where the `:hasDiameter` property can be any value. This subject is then stored in the `?roll` variable, and the actual `:hasDiameter` property values are stored in the `?diameter` variable. The `Minus` clause removes the subjects that also contain the `:hasGrindRoll` property as we are not interested in the historical rolls grindings data that previously contained this diameter. We then use "`Group by`" which creates columns for the fields we have selected. Generally, these will always be the same variables in the `Select` clause. In this example, we are only printing out the diameter variable.

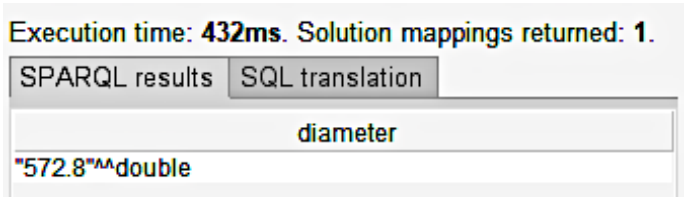


Figure 4.6: The SPARQL result from previous listing.

Figure 4.6 displays the results of this SPARQL query. The results show that `572.8` is the only diameter value that has three or more rolls that were last located between the 10–17th of January 2020. We create another query to print out these rolls in Listing 4.2.

Listing 4.2 is a query written to display all the rolls that have the specific diameter of `572.8`. Similarly, we first select the ontologies we wish to use by declaring their prefixes. These are identical to our previous query. This time, however, our `Select` and `Group By` clauses contain the variables `?roll, ?rollid, ?partner,` and `?diam` which will be the columns containing our results. Once more, we use the `Where` clause to filter our results.

We created the variable `?roll` to store all the subjects that contain both the `:hasRollID` and `:hasDiameter` properties. The value of these properties are not specified and thereby can be any value. Each of these ?roll subjects may contain the optional property `:hasPartner`, but must not contain the `:hasGrindRoll` property.

Code Listing 4.2: All rolls that have the diameter of 572.8.

```
PREFIX time: http://www.w3.org/2006/time#
PREFIX : http://www.semanticweb.org/sadee/ontologies/2021/1/SCRO#

SELECT ?roll ?rollid ?partner ?diam
WHERE {
    ?roll :hasRollID ?rollid .
    ?roll :hasDiameter ?diam .
    OPTIONAL {
     ?roll :hasPartner ?partner .
    }
    MINUS {
     ?roll :hasGrindRoll ?grind .
    }
    FILTER (?diam = "572.8"^^xsd:double)
}
GROUP BY ?roll ?rollid ?partner ?diam
```

Then, we filtered the `?diam` value to only return rolls that contained the diameter value of `572.8` which was the result from the first SPARQL query in Listing 4.1. Figure 4.7 displays the query result. Here we can see that `roll_1678` and `roll_1679` are partners that contain the diametric value of `572.8`. We can also see that there is a roll in storage with ID of `4631` that has the same diametric value and has no assigned partner. This type of query can be used to identify replacement rolls in case a roll gets damaged or needs replacing. Storage roll data is stored separately from active roll data, so this query skips the need for integration.

Execution time: **1.32s**. Solution mappings returned: **3**.

SPARQL results | SQL translation

roll	rollid	partner	diam
roll_1678	1678	1679	"572.8"^^double
roll_1679	1679	1678	"572.8"^^double
storage_roll_4631	4631		"572.8"^^double

Figure 4.7: The SPARQL result from previous listing.

4.4 Ontology Validation

Ontology validation is a fundamental requirement when developing ontologies. It is essential to ensure that the quality of an ontology is adequate and the knowledge representation is accurate. There are many ways to validate ontologies; examples include task-based validation, criteria-based validation, data driven validation and expert knowledge validation [46]. In addition, a well known ontology validation tool known as "OntOlogy Pitfall Scanner" (OOPS) [253] has been developed to validate ontologies by detecting common pitfalls aligned to a dimension classification developed in [114]. We use a combination of these approaches to validate SCRO. Additionally, the Protégé IDE includes stream reasoning mechanisms that check the consistency and correctness of an ontology. As we have adopted the Ontop framework, we have opted to use the *Ontop stream reasoner (version 4.1.0)*, which includes these validation checks, and will not allow query answering until these validation checks have been reached. Figure 4.8 displays that there are no inconsistency or correctness errors in the console log when running the stream reasoner, allowing us to query the data.

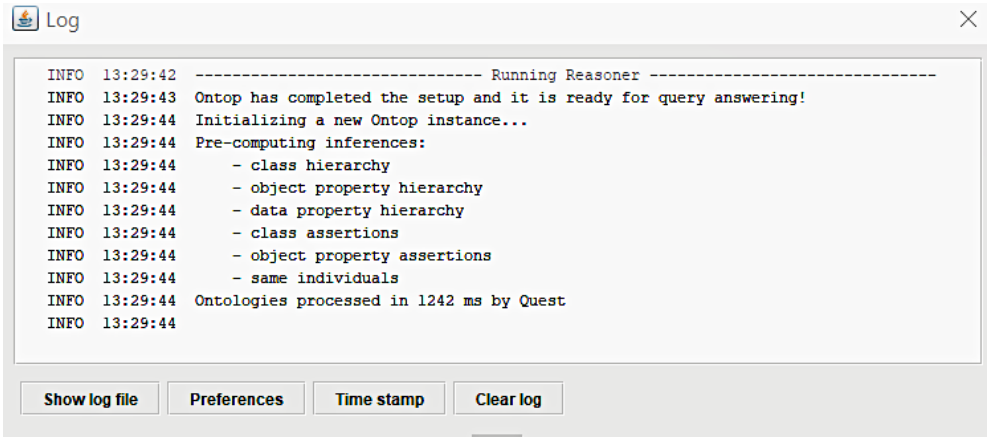


Figure 4.8: The console displaying no inconsistencies or incorrectness messages when starting the ontop stream reasoner.

4.4.1 OntOlogy Pitfall Scanner

Different pitfalls have different impacts and importance. Because of this, OOPS categorises the evaluated results into three different levels: *critical, important and minor*. When evaluating SCRO, OOPS displayed zero critical pitfalls, two important pitfalls, and a handful cases of minor pitfalls. The two important pitfalls are results from the P11 specification *"missing domain or range in properties"*. These include our object properties ``hasComponent'' and ``isComponentOf''. However, according to [142], when using OWL, it is best practice not to specify the domain and ranges of superclasses but instead mention them in their respected subclasses. This is because the domain and ranges in OWL should not be viewed as constraints as this may cause unexpected classification and side effects [142], but rather viewed as `axioms` for reasoning. As a result of this, we have concluded to explicitly not specify the domain and ranges of these properties, but have included the domain and ranges of all the subclasses of these properties. For example, the object property `hasComponent` does not include a domain and range, but its subclass *hasPickleComponent* contains the domain `Cold Rolling Mill` and the range `Pickle Line`. On the other hand, *Minor* pitfalls include some elements missing annotations, or not explicitly declaring the inverse relationships of such object properties. These minor pitfalls do not affect the usability and consistency of the ontology and thus, remain as low-priority future changes.

4.4.2 Expert Knowledge Validation

As this work in linked closely with industry, we have also validated SCRO with domain experts from Tata Steel. We set up a demonstration and presented the ontology to ensure that our understanding of the cold rolling processes were accurate and aligned with the knowledge from the domain experts. This demonstration clarified the questions and ambiguity we had related to some of the cold rolling processes, e.g., how some components in the pickle line were linked and operated, as well as their details and purpose. This was done over a few face-to-face interview-style meetings. Additionally, we gained better understanding of the future goals that the steel industry are working towards and their current limitations, one of which being data integration. One benefit of using ontologies is to use the knowledge graph paradigm to exploit data virtualisation which we also presented in the demonstration.

4.5 Conclusions

To conclude, this chapter presents a novel ontology that models and structures domain knowledge of cold rolling processes and activities within a steel plant. The purpose of the ontology is to improve data semantics and interoperability within the domain of smart manufacturing and a step closer to accomplishing Industry 4.0 standards. To our knowledge, this work is the first to develop an ontology for the cold rolling processes within a steel plant. We focus on capturing the knowledge for the pickle line, accumulators, and mill sub-

processes which are core to a cold rolling mill. The domain knowledge we have captured comes primarily from a case study with Tata Steel at their plant in Port Talbot in the UK.

We have demonstrated how knowledge representation methods can be applied to create virtual knowledge graphs from the data set, integrating data from multiple databases and sources using the semantics of the data. This integration can be produced without an ontology, however, an ontology enhances the clarity and consistency of the data, and provides greater inference capabilities to produce more powerful and richer knowledge graphs.

The ontology was developed using the eXtreme Design Methodology which includes using Ontology Design Patterns. We set up a case study that used real-world cold rolling data sets that were provided by the domain experts which validated the performance and functionality of SCRO. These data sets included roll data, roll refurbishment data, and roll storage data, all of which were in different tables and not integrated. We used the Ontop framework to deploy virtual knowledge graphs for data integration, data access, data querying, and condition-based maintenance purposes. SCRO was evaluated by both the ontology pitfall detection system *OOPS!* and domain experts from Tata Steel. OOPS! confirmed that there were no critical errors or inconsistencies in SCRO, and the domain experts confirmed that the knowledge in SCRO was uniform and accurate.

The domain knowledge encoded in SCRO is aligned with the processes and assets from the Port Talbot plant, which may differ from other plants from other companies. A key future goal will be to look at more cold rolling plants and compare any differences in processes and machinery to generalise the ontology and add flexibility. Another future goal is to add more complex logic axioms formalisations and relationships of the knowledge to enrich the knowledge base. Presently, we have only mentioned basic axioms that show the relationships between classes and their properties. This chapter does not include any logical constraints or logical connectives, whereas the ontology currently contains a few constraints, such as work rolls and backup rolls classes being disjoint. One future goal is to finish developing a full set of constraints for SCRO classes and properties. Finally, another future goal is to use SWRL rule reasoning techniques together with SCRO to perform rule-based reasoning for predictive maintenance purposes.

4.6 Core Reference Ontology for Steelmaking (CROS)

The novelty of SCRO lead to collaborative work between Dr. Qiushi Cao and myself, where we expanded the scope to capture the wider consensus of steelmaking [55]. In this paper, we introduced The Core Reference Ontology for Steelmaking (CROS), an ontology that formally describes the broader and essential concepts of steelmaking.

The knowledge obtained was acquired through collaboration with domain experts from Tata Steel, as well as from publicly available literature, including international standards and existing steel domain ontologies. CROS comprises a total of 276 classes, encompassing a wider array of steelmaking processes, materials, facilities, tools, products, byproducts, and any humans involved in the processes. Figure 4.9 displays a handful of these concepts.

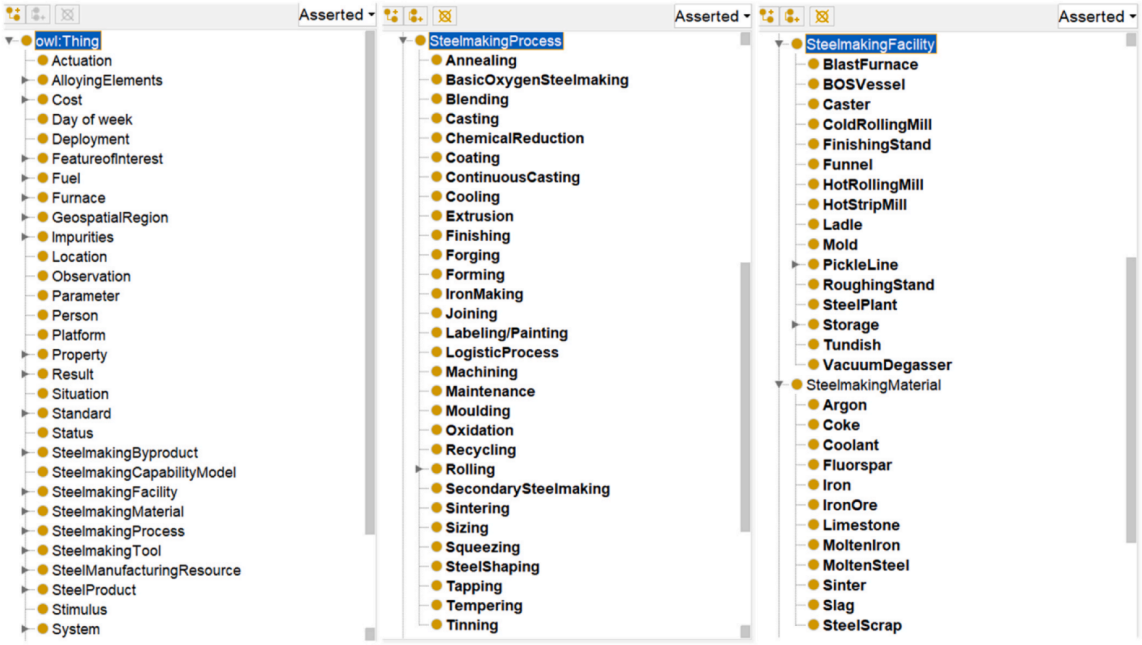


Figure 4.9: The class hierarchy of CROS from [55].

CROS also showcased more ways of applying ontology-based systems in an industrial setting, offering examples of condition-based maintenance for cold rolling mills.

The evaluation of CROS included assessment using the Ontology Pitfall Detection System, *OOPS!*, alongside similar validation methodologies with domain experts at Tata Steel. Further detail and conclusions can be found in [55].

CHAPTER 5

Steelmaking Predictive Analytics Based on Random Forest and Semantic Reasoning

Contents

This chapter introduces a human-in-the-loop framework that integrates machine learning models with semantic technologies to aid decision-making in the domain of steelmaking. To achieve this, we convert a random forest (RF) into rules in a Semantic Web Rule Language (SWRL) format and represent real-world data as a knowledge graph in a Resource Description Framework (RDF) format, capturing the meta-data as part of the model. A rule engine is deployed that applies logical inference on the knowledge graph, resulting in a semantically enriched classification. This new classification is combined with external domain expert knowledge to provide improved, knowledge-guided assistance for the human-in-the-loop system. A case study in the steel manufacturing domain is introduced, where this application is used for real-world predictive analytic purposes.

5.1 Introduction

Ontologies and knowledge graphs have become a well-established and recognised way of modelling and enriching knowledge within a particular domain. In the context of smart manufacturing, semantic technologies have become a promising solution for addressing Industry 4.0 challenges [237] and have many advantages including (1) the ability to provide a shared, machine-understandable vocabulary for data integration and exchange among components [9], (2) the capability to access and query data at a virtual level without physical data integration [335], and (3) simulating cognitive decision-making tasks through logical deductions, rules, and reasoning [327].

Meanwhile, machine learning (ML) models have been widely adopted in manufacturing to optimise, control, troubleshoot, and improve process operations and automatisation [74]. Random Forests (RFs) are one popular and recognised ML model. RFs have a wide range of applications and benefits as mentioned in Chapter 2.6.1.7. We have chosen RFs as our ML model for this chapter for a few reasons. First, the tree-like structure of the decision trees within a random forest can be captured in *IF-THEN* format, which resembles semantic-based rules. Second, some of this work is in collaboration with peers from the material science department, as well as our industrial partners at Tata Steel. Thus, after some discussion, we agreed on following a standardised model among the work, where RFs were deemed the most appropriate model for this research due to their simplicity and accuracy.

However, these ML models are typically faced with challenges including the lack of context-aware information within dynamic production environments and semantic interoperability [338]. Thus, the development of hybrid models that combine semantic technologies and ML has been proposed to address these challenges.

In this chapter, we demonstrate a hybrid model for predictive analytic purposes in the domain of steelmaking. Presently, work rolls during cold rolling operations are refurbished based on the quantity of steel coils produced rather than their physical conditions. One motivation of this chapter is to develop an application to aid operators on the shop floor with critical decision-making tasks in order to optimise the yield, efficiency, and overall life of work rolls. The scheduling of the refurbishment process can then be targeted based on the condition of the rolls rather than an estimation based on the total tonnage produced. Additionally, anomalies and accidents within a steel plant such as spalling and overloads [266] can be identified and avoided pre-emptively with greater semantic interoperability.

To achieve this goal, we propose a human-in-the-loop framework, represented in Figure 5.1, that leverages knowledge representations and reasoning mechanisms and random forests to provide semantically enriched classifications. These classifications are then further combined with expert knowledge to support with decision-making tasks. The support provided by the framework is similar to that of an expert system in the domain of steelmaking; however, the framework itself is not exactly an expert system and serves a more significant purpose. Figure 5.1 is explained further in the methodology section of this chapter.

The contributions of this chapter are twofold: (1) we introduce the Random Forest Ontology (RFO) that captures and models a random forest at a conceptual level, which

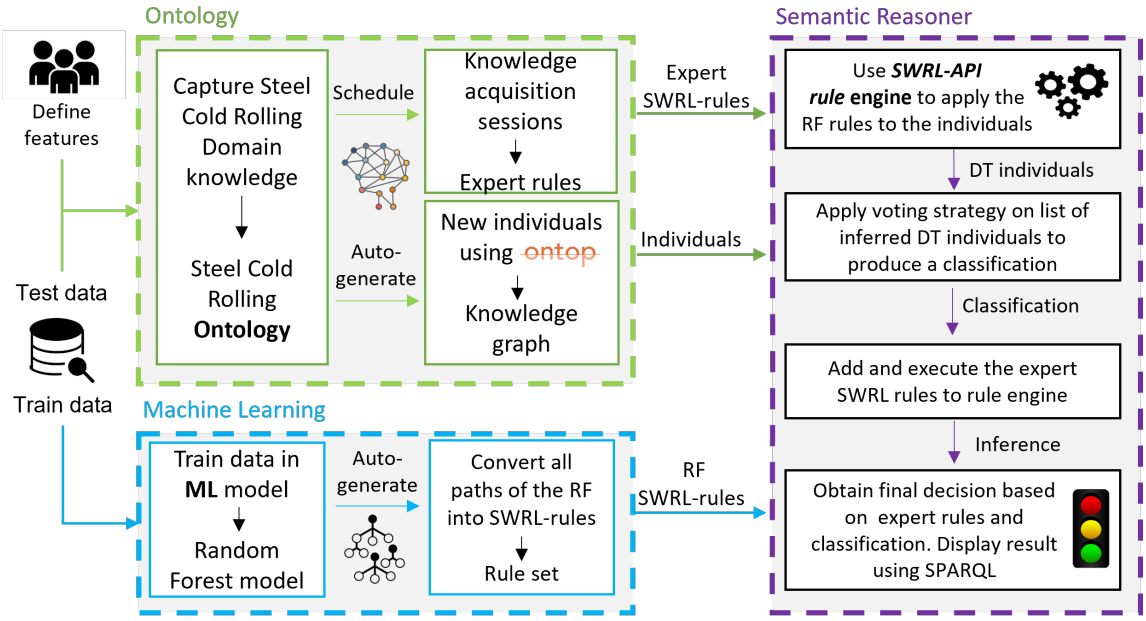


Figure 5.1: The methodology of our proposed framework.

can represent and perform RF classification using rule-based reasoning, where the features are represented within a knowledge graph; and (2) we demonstrate an iterative process to integrate external domain—expert knowledge with RF classification to provide more comprehensible decision-making assistance for the human-in-the-loop system.

The outline of this chapter is as follows. Section 5.2 introduces the related works exploring existing hybrid models that combine semantic reasoning with ML. Section 5.3 introduces the methodology of our proposed approach. In Section 5.4, a use case is presented, where the framework is applied to assist steel operators, and the results of the application are utilised to validate the framework in Section 5.5. Finally, we end with the conclusions in Section 5.6 and future work in Section 5.7.

5.2 Related Work

There exists a significant amount of research that employs both semantic reasoning for inference tasks and ML for predictive tasks, but there are few works that combine the two paradigms. This section investigates the existing literature that combines ML models with semantic reasoning.

Rajbhandari et al. [262] introduced a hybrid model that combines ontology with random forest classification to address the lack of formalisation in systematic models for image object identification. Their model combines two sets of rules: (1) generalised domain knowledge rules from the literature and domain experts, and (2) localised rules obtained from an RF classification to classify landslides. In this chapter, we re-create the RF classification using

rule-based reasoning, where each rule denotes a path of an RF that is later combined with expert rules.

Similarly, Shoaip et al. [292] proposed an interpretable model to detect Alzheimer's disease using rule-based reasoning by combining the Alzheimer's Disease Diagnosis Ontology with a combination of different ML models. The Semantic Web Rule Language (SWRL) rules were obtained by combining a decision tree with the Java repeated incremental pruning model to produce a classification with enhanced reasoning efficiency. The rules were produced in a non-technical manner so that domain experts such as doctors could understand them and provide feedback without prior technical training. Meanwhile, in our framework, two distinct rule sets are employed. The first set comprises expert rules that represent the paths within the RF utilised for rule-based reasoning. The second set consists of domain-expert rules that are applied on top of any newly acquired classifications. Neither of these rule sets are displayed to domain experts.

Jabardi et al. [157] used ontological engineering and SWRL rules to identify and classify fake accounts on Twitter. The authors evaluated their ontology-based classifier results with different machine learning techniques, including naive bayes, logistic regression, and Support Vector Machine (SVM), using the Waikato Environment for Knowledge Analysis (WEKA) tool. For this approach, the SWRL rules were manually written. The same authors expanded their research to cover DTs in [158] but did not cover how the rules were created. In contrast, the SWRL rules representing the RF in this chapter are systematically generated through an algorithm introduced in Section 5.3, while our domain-expert rules are translated from natural language manually.

Johnson et al. [166] developed a method to model ontological-based knowledge into a DT through a generic and interactive process involving domain experts. Their method follows a data-driven rule-learning model that iteratively implements qualitative knowledge from the ontology into the DT until a complete DT is formulated. The authors exemplified the model through a case study focused on predicting food quality. Meanwhile,our framework integrates domain-expert knowledge with RF classification without the need to recreate the RF each time. Additionally, we follow an iterative process to access and validate our domain-expert rule set.

Sarkar et al. [278] introduced CHAIKMAT 4.0, a hybrid AI model that integrates semantic reasoning and machine learning paradigms to advance trusted flexible manufacturing aligned with Industry 4.0 goals. Their approach involves deploying deep learning and machine learning models for predicting machine capability and text analysis, and utilises semantic reasoning to capture common-sense knowledge, enabling the generation of explanations for general tasks within a manufacturing production line. While the authors acknowledge existing technologies capable of achieving such goals, the implementation is left for future work.

Ammar et al. [10] introduced a proof-of-concept recommendation system that leverages machine learning and semantic technologies for explainability in AI. In their paper, the authors developed a hybrid prototype featuring a knowledge-driven recommendation system aimed at improving mental health surveillance based on adverse childhood experiences.

The authors placed significant emphasis on ontological aspects and employed a question-answering agent from the Google DialogFlow engine to serve as a semantic knowledge base. The results in their prototype were compared to those of an ML classification, showcasing the added advantages of explainability. However, the hybrid method was still in the proof-of-concept phase and no concrete implementation was provided.

Bettini et al. [36] introduced proCAVIAR, a hybrid model that combines semi-supervised machine learning models with probabilistic knowledge-based reasoning for activity recognition. In their approach, the authors developed an ontology to capture the knowledge of various activities, including running, sitting, cycling and standing. Afterwards, probabilistic semantic reasoning was applied to comprehend these activities, and the outputs of the reasoner were combined with a ML classifier to generate a final prediction of the user's activity. In our work, we replicate the ML classifier itself using semantic reasoning.

Tofighi-Shirazi et al. [309] proposed a novel approach that combines semantic reasoning and ensemble machine learning classification for a framework designed to detect obfuscation transformations. The authors generated obfuscated samples and used semantic reasoning to extract raw data from these samples. The extracted data were then utilised to train various ensemble models for classification. In our study, we differ in approach, as we do not employ semantic reasoning to extract raw data. Instead, we utilise it to recreate the ensemble classification process of the RF using symbolic methods.

Pukkhem et al. [259] employed decision trees as the basis for generating an ontology with the objective of predicting the number of students graduating at a University. The DTs were created using C4.5/J48 algorithms. The authors emphasised how ontological representations play a key role for predictive purposes and the possibility of using SWRL rules to infer knowledge. However, the implementation details were not included and are unidentified as part of future work.

Finally, Cao et al. [61] integrated symbolic and statistical AI technologies for automation and predictive analysis within the domain of smart manufacturing. The authors adopt ontological reasoning with statistical AI techniques using real-world datasets to generate a rule set in SWRL format. These rules were able to automatically detect machine anomalies within a shop floor. Meanwhile, our contribution focuses on utilising different methodologies to develop a framework for assisting steel operators on the shop with decision-making tasks. Our research focuses on integrating domain expert knowledge within a framework, where the assistance provided can be validated against the knowledge of the steel operator.

5.3 Methodology

Figure 5.1 displays the methodology of the proposed framework. There are three key components: Ontology, Machine Learning, and Semantic Reasoning which are described in the next sections.

5.3.1 Ontology

Ontologies have the capability to explicitly define concepts within a specific domain, along with their semantics [194]. In this chapter, we utilise the Steel Cold Rolling Ontology (SCRO) ontology introduced in the previous chapter.

The initial step of the framework involves converting the dataset from a Structured Query Language (SQL) database into the Resource Description Framework (RDF) format to produce a knowledge graph. To achieve this, we employ the Ontop framework to automatically translate data into an RDF format [336] via mappings, following similar methods in the previous chapter.

Ontop Mappings

There are two key components of an Ontop Mapping: (1) the `Source` containing an SQL query that allows the user retrieval of specific data through a `Select` clause, (2) the `Target` which precisely maps the selected columns of a table to the chosen data or object property of an ontology.

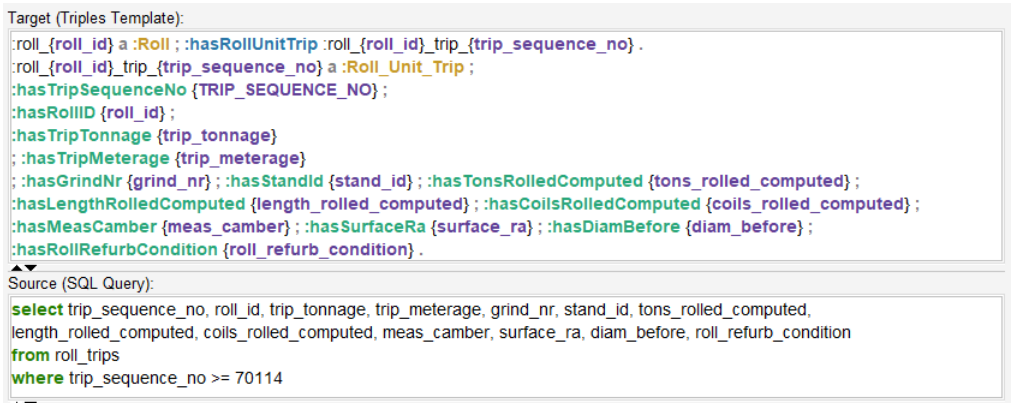


Figure 5.2: The mapping between SQL data and ontology using Ontop.

Ontop Mappings are constructed manually and therefore require some knowledge of the syntax of the language. Figure 5.2 provides an example of an Ontop Mapping. In the source, we select which columns to include in the knowledge graph, while the target maps those features to the corresponding data properties in the ontology. The knowledge graph is then automatically generated by the materialisation process in the Protégé IDE. The creation of the knowledge graph is the initial step in enabling semantic reasoning in our framework.

5.3.2 Machine Learning Models

Decision trees (DT) are capable of inferring rules from historical data to classify data points as belonging to one of a predefined set of categories. They require relatively minimal

data pre-processing steps and generate a definite set of simple rules that assign a unique category to each instance. During the training process, the DT progressively splits the dataset according to the value of a certain feature until a classification is reached for each resulting subset of the dataset. Thus, a DT can be viewed as a set of discrete rules. Each rule is composed of a logical conjunction of elementary formulae, accompanied by a class attribution. Any unseen instance satisfies precisely one of those rules, so that the classification performed by a DT is both human and machine interpretable. They are also advantageous as they have similar constructs to semantic web-based rules. However, DTs suffer from high variance and can quickly overfit the training set.

Meanwhile, a Random Forest (RF) [47] was introduced as a classifier to overcome these shortcomings. An RF classifier is formed by several DTs, each built using a randomly sampled subset of the training set, containing a random subset of features. The predicted category for each instance is determined by combining the output from the trees, which reduces the variance and hence improves the accuracy of the model. An algorithm known as a voting strategy is then used to calculate the final classification from the DTs. There are two main approaches: (1) using a `majority voting` strategy where the final classification is the modal value of all the DT predictions, or (2) using a `soft voting` strategy where the final classification is derived by calculating the average value of all the DT predictions.

We chose RFs over other ensemble methods due to their advantageous and easily comparable rule-like structure, which aligns well with the structure of semantic rules. This is particularly noticeable when contrasted with other machine learning models. Random Forests have been extensively studied, demonstrating high accuracies across various complexities of classification and regression tasks [5]. Additionally, RFs provide a versatile set of techniques such as node splitting and various feature selection methods, further contributing to their suitability for our application [7]. These are some reasons why our industrial partners preferred this ensemble methods over bagging, boosting, XGBoost, and other models.

```
|--- grind_nr >  53.50
|   |--- diam_before <= 573.40
|   |   |--- surface_ra <= 0.81
|   |   |   |--- weights: [0.00, 7.00, 1.00] class: 1.0
|   |   |--- surface_ra >  0.81
|   |   |   |--- weights: [102.00, 36.00, 66.00] class: 0.0
|   |--- diam_before >  573.40
|   |   |--- trip_tonnage <= 1832.38
|   |   |   |--- weights: [57.00, 35.00, 10.00] class: 0.0
|   |   |--- trip_tonnage >  1832.38
|   |   |   |--- weights: [30.00, 0.00, 5.00] class: 0.0
```

Figure 5.3: The structure of a scikit-learn RF in plain text format.

The structure of a random forest is displayed in Figure 5.3, presents a concise snippet in plain text that highlights the straightforward rule-like structure. Each line within the representation contains precisely one condition involving one specific feature. If the condition is met, the traversal continues to the next line, recursively. In case the condition is not

satisfied, the traversal instead travels down the pipe into a new line, typically involving the same feature with a reversed condition. This iterative process is repeated until a leaf node is reached, where weightings and classification information are stored.

In this specific example, each leaf node is characterised by three potential weighted classifications enclosed in square brackets. The weighted values represent the number of training samples that satisfied all the conditions along that path up to the respective leaf node for each class. In a Breiman RF, the final classification is determined by selecting the maximum value among the three values, and simulating a hard voting strategy. In contrast, a soft voting strategy would involve calculating the average of the three values to derive the final classification. This distinction in voting strategies adds to the flexibility and adaptability of the RF model.

Example 5.1 To illustrate the determination of a class, we consider the first leaf node in Figure 5.3. Such node is represented by the following information: weights : `[0,7,1]` `class 1.0`. This indicates that a total of 8 records from the training set satisfy that exact set of inequalities associated with this node. Among these, 0 belong to `class 0`, 7 belong to `class 1`, and 1 belong to `class 2`. Hence, in this example, `class 1` is prioritised for both voting strategies, indicating its higher prevalence in the training set.

Within the framework, after training a RF for a classification task, the RF is stored in plain text format using the `export_tree` method provided by sci-kit learn [244]. There are also other tools available for the exporting process but the built-in functionality by sci-kit learn is sufficient for this task. However, the exported format is stored in plain text which needs to be converted into a standardised semantic format. Thereby, we have developed an algorithm, described in the next section, to convert the RF from plain text into semantic-based rules formatting.

5.3.2.1 Random Forest Ontology and Algorithm

Random Forest Ontology (RFO) was developed to capture, model, and label the generic concepts of a random forest at a conceptual level. RFO includes fundamental classes of a random forest, such as `Random_Forest, Decision_Tree, Path, Voting_Strategy`, and others, displayed in Figure 5.4. RFO can be imported and combined with an existing ontology containing a knowledge graph to reproduce the RF classification process using semantic methods.

The first step of achieving RF classification based on ontological rule reasoning is by converting the RF into a format that supports logical inference and reasoning. We chose to adopt the Semantic Web Rule Language (SWRL) developed by the W3C consortium [144] to represent our rules as it is recognised as the leading rule language, which is well studied in the literature. In the framework, all paths of a RF are translated into SWRL rules by an algorithm we developed, displayed below as Algorithm 1, first introduced in [26].

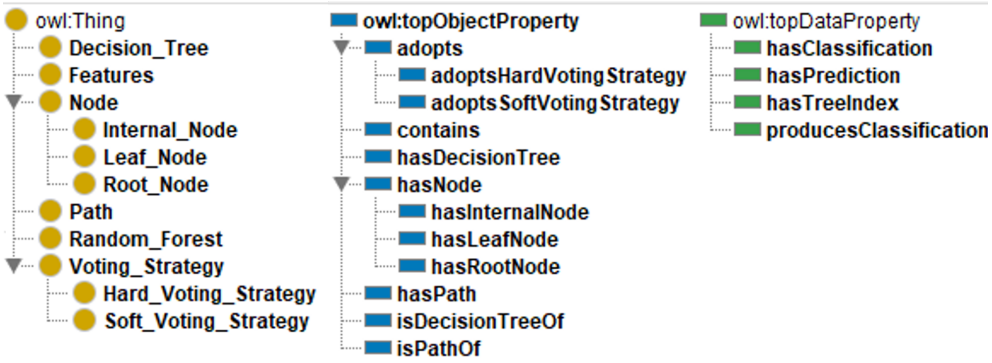


Figure 5.4: The classes, object properties, and data properties of RFO.

Algorithm 1: An algorithm for SWRL-Rule generation based on an existing RF.

$I \leftarrow 0$ {index of trees}
$L \leftarrow [\,]$ {list of features}
for each *tree* in *forest* **do**
  for each *node* in *tree* **do**
    $d \leftarrow$ depth of *node* in *tree*
    if *node* $\neq$ leaf node **then**
      $L[d] \leftarrow$ *node*
    else
      $R \leftarrow$ " " {string variable for forming a rule}
      for $i = 0$ **to** d **do**
        if $i > 0$ **then**
          R += " $\wedge$ "
        end if
        R += "$L[i]$"
      end for
      $p \leftarrow$ prediction {based on weightings}
      R += " $\rightarrow$ " + result$(node, p, I)$
    end if
  end for
  I += 1
end for

The algorithm feeds in two lists as input, which creates a mapping between the features in the training set and their corresponding data properties in the ontology, e.g., `trip_tonnage` as `hasTripTonnage`. The algorithm then traverses through the random forest, creating a new rule for each path it finds and storing it in RFO.

Each SWRL rule contains the `MakeOWLThing` method from the `SWRL-X` library. When the rule is triggered, this method instantiates a new instance of the `Decision_Tree` class,

and the resulting prediction is added to the knowledge graph. Additionally, a `Random_Forest` instance is generated and incorporated into the knowledge graph, establishing the connections between all the `Decision_Tree`s in the RF, capturing their index.

5.3.3 Semantic Reasoning

Semantic Reasoners or Semantic Rule Engines are software that provides a mechanism for inferring logical deductions from a set of asserted axioms using a restricted set of first-order formulas [327, 192]. In simple terms, a rule engine enables the creation of logical rules, which can be applied to a dataset to derive new knowledge from the existing knowledge [326].

The reasoning process involves two inputs: (1) an ontology containing a knowledge graph and (2) a rule set in an SWRL format. When the rule engine is executed, these rules are applied to the data entries in the knowledge graph, leading to the inference of new knowledge.

Thus, in the context of the framework, we are deploying a reasoner to capture the operation of the RF classification process. Thus, when the rule engine is inferred, each data entry yields N uniquely generated DT instances, where N corresponds to the size of the random forest. Afterwards, a voting strategy is applied for each data entry by calculating the average or modal value of the generated DT predictions. Algorithm 2 is an example of applying the rule engine onto a Breiman RF containing three possible classifications, determining the modal value of these classifications. Initially, each class is declared with having zero entries. Then, each decision tree instance generated contains a class value, which gets incremented to the counter of that class. Finally, the modal value between all classes are calculated to produce the final classification.

Example 5.2 We consider a random forest with $N = 10$ decision trees. This can be explicitly represented in the RFO ontology with one instance of the RF class RF_001 and N instances of the DT class, DT_001 to DT_N. These DT instances are acquired by executing the rule engine on a rule set generated by Algorithm 1. Each instance of the DT class is linked to the RF individual via the `isDecisionTreeOf` relation, as well as its inverse relation `hasDecisionTree`. Furthermore, each instance of the DT class contains the data property `hasPrediction` to carry its classification. Afterwards, a voting strategy is executed to calculate the final classification of each individual by calculating the modal value or average of the `hasPrediction` values, as outlined in Algorithm 2.

Afterwards, domain expert knowledge can be integrated to produce a more comprehensible output that provides greater decision-making assistance for the human-in-the-loop. The expert knowledge takes the classification as input, and based on the condition of the rolls at a given interval, provides a recommendation for the operator, where the output can be validated against the knowledge of the operator. To achieve this, the domain expert knowledge has to be acquired from domain experts using well-studied knowledge acquisition methods, and translated into an SWRL format so that logical reasoning can be applied.

In essence, we are embedding the capabilities of a simplified expert system as part of the framework. These results are displayed using the SPARQL Protocol and the RDF Query Language [246].

Algorithm 2: An ontological classifier for a Breiman RF with classes $0, \ldots, n$.

Input: Ontology file incl. knowledge graph with individuals
Text file containing a list of SWRL rules generated from a RF
Output: Updated Ontology file with new acquired knowledge

load Ontology
let $I1$ be a set of individuals relating to the relevant data instances
read in SWRL rules into rule engine
apply the rule engine
let $I2$ be a set of individuals relating to `Decision_Tree` instances
for all r in $I1$ **do**
  for all i from 0 to n **do**
    $C[i] \leftarrow 0$ {initialise count for class i}
  end for
  for all t in $I2$ **do**
    if t is instance of `Decision_Tree` class for r **then**
      let p = prediction value of t
      $C[p]$ += 1
    end if
  end for
  $P \leftarrow \text{mode}(C[0]\ \&\ C[1]\ \&\ \ldots\ \&\ C[n])$ {compute mode of highest class}
  store P as the `hasClassification` property for r
end for
delete all intermediate rules
delete all intermediate `Decision_Tree` individuals
return new ontology file

5.4 An Application of the Framework: Use Case of Cold Rolling

This section demonstrates the applicability of the framework, and introduces an example where the hybrid approach that uses real-world industrial data is used for predictive analytics purposes.

5.4.1 Use Case

This section demonstrates the applicability of the framework and introduces an example where the hybrid approach, utilising real-world industrial data, is employed for predictive analytics purposes. Specifically, we present a use case illustrating how the framework can be applied in a cold rolling environment to assist operators in the crucial task of predicting the optimal time to stop operations to refurbish the rolls, providing helpful and actionable advice for the operator.

5.4.2 Application Use Case

As mentioned, work rolls are often refurbished prematurely or belatedly; hence, their efficiency and yield are suboptimal. Therefore, it is important for operators on the shop floor to halt operations at an optimal point in time in order to maximise the yield of the work rolls, while simultaneously not overworking the work rolls, which in turn results in the production of defective steel adding additional refurbishment costs. Typically, in our study, work rolls begin their life with a diameter of 600mm and are scrapped when approaching 520mm. An average refurbishment on a healthy work roll removes approximately 0.2mm of stock. Meanwhile, if a work roll is damaged or over-worn, it may result in a significantly greater stock reduction. In extreme cases, an over 10mm stock removal may be necessary, which is a significant cut in lifespan.

The acquisition of this knowledge involves active engagement with domain experts and stakeholders. Additionally, further interviews with domain experts are conducted to gather insights regarding the optimal timing to stop operation and refurbish the work rolls. Using this acquired knowledge, we construct a static set of expert rules designed to encapsulate these insights. This rule set is an essential component within the application, as these rules are applied to the real-time condition of the work rolls, influencing the final decision produced by the framework.

Thus, at any given point during cold rolling operations, the real-time conditions of the work rolls can be captured as a timestamp and input into the application. During this process, the data is integrated into a knowledge graph containing historical information about the work rolls, including details such as their previous grindings and stock reduction values. Subsequently, the knowledge graph is passed through a semantic reasoner, which applies logical deduction to predict the live condition of the work rolls. This prediction is accomplished through the application of a rule-based random forest classification, following the steps mentioned in the proposed methodology.

Once this classification is obtained, it is combined with the expert rule set, generating a status for the operator and offering clear advice on whether to proceed with the cold rolling operations, along with insights into the recommended tonnage. Ultimately, the decisions produced by the application are intended to assist the human in the loop with their decision-making process and can be utilised as guidance.

In this use case, we apply the framework to the last 100 roll unit trips of our industrial partners and perform a comparison with domain experts if the assistance provided is useful

and accurate.

5.4.3 Producing a Random Forest

For this study, we deploy a supervised RF model to predict the condition of the work rolls at a given interval. The outcome can be one of three classifications: `class 0`, implying the condition of the work roll is `Bad`, i.e., the roll requires a considerable amount of stock reduction to remove the worn surface; `class 1`, where the condition of the work roll is considered as `Best`, and thus requires minimal stock removal; or `class 2`, implying the condition of the work roll is `Good` and an average stock reduction value is necessary.

There are many impacting factors that affect the rolls, which are collected and used for training and testing the RF model. This includes a combination of dynamic sensory and historical static data of the work rolls. The dynamic data contains live data read from sensors, which include the total tonnage and meterage rolled during a trip, speeds, temperature, as well as coil usage data. This coil usage contains information regarding the steel grades and full chemical composition for each coil processed, e.g., its carbon or silicon values. Meanwhile, the static data provide information regarding the roll historical data, such as their previous grindings, stock reductions, positioning, tons and length rolled, etc. This wide collection of data was explained by domain experts and data scientists from our industrial partners, who assisted in the data collection and aggregation aspects to build our random forest model.

To build the RF classifier in our application, 80% of the original dataset (9781 samples) was used as the training set. The train–test split was performed randomly and in such a way that the original proportion was respected. The value of the hyperparameters `n_estimators` and `max_depth` was set using grid search and validated through the performance of the metrics. Finally, the optimised values for `n_estimators` and `max_depth` were 20 and 22, respectively.

The RF contained a total of 20 decision trees that contained a total of 25,657 paths. The `majority` voting strategy technique was applied to calculate the final classification by computing the modal value of all the decision trees in the RF. The RF was exported to plain text using the export_tree method mentioned previously. The accuracy of the random forest is measured in terms of precision, recall and f1-score and their weighted values are displayed below.

`Precision:`	0.78	`Weighted Precision:`	0.78
`Recall:`	0.75	`Weighted Recall:`	0.78
`F1-score:`	0.76	`Weighted F1-score:`	0.77

When running Algorithm 1 for this particular RF, the 25,657 different paths produced an equal amount of SWRL-rules that were passed to the semantic reasoner for inference. We discovered that one limitation of this approach is the capability of our selected semantic reasoner to reason on such a large scale. Thus, we iteratively cut down the reasoning process into chunks of 100 data entries at a time.

5.4.4 Reasoning

To build our knowledge graph, we created an Ontop Mapping that correlated the last 100 cold rolling trips from our local database into individuals in the ontology. These data entries were an instance of the `Roll_Unit_Trip` class.

First, the 25,657 rules and the 100 data entries were input into the reasoner, initiating the application of rules to the knowledge graph for logical inference. Listing 5.1 displays the syntax and format of one of the 25,657 rules as an example. As shown, the antecedent of every rule starts with `Roll_Unit_Trip(?trip)` to target the corresponding instances of the `Roll_Unit_Trip` class in the knowledge graph that the rule is applied to. Each instance must be linked to an instance of the RF class via the `RFO:hasRandomForest` property provided by RFO (which is typically mapped automatically). Then, each rule contains the features and conditions of the path, in this case `hasGrindNr, hasDiamBefore and hasSurfaceRA` and their conditions, respectively. The end of the antecedent exploits the `makeOWLThing` method from the `SWRL-X` library to instantiate a new, unspecified instance in the knowledge graph, which is declared in the consequent of the rule to be of type `RFO:Decision_Tree`. The consequent also contains the `RFO:hasPrediction` data property to store the classification and the `RFO:treeIndex` data property to store the index of the tree in the RF.

Code Listing 5.1: Example of an SWRL rule.

```
Roll_Unit_Trip(?trip) ^ RFO:hasRandomForest(?trip, ?rf) ^
hasGrindNr(?trip, ?GrindNr) ^ swrlb:greaterThan(?GrindNr, 53.50) ^
hasDiamBefore(?trip, ?DiamBefore) ^ swrlb:lessThanOrEqual(?DiamBefore, 573.40)^
hasSurfaceRA(?trip, ?SurfaceRA) ^ swrlb:lessThanOrEqual(?SurfaceRA, 0.81) ^
swrlx:makeOWLThing(?DT, ?trip) -> RFO:Decision_Tree(?DT) ^ RFO:isDecision
TreeOf(?DT,?rf) ^ RFO:hasPrediction(?DT, 1) ^ RFO:hasTreeIndex(?DT, 1)
```

After the rule engine completed its process, each `Roll_Unit_Trip` individual accumulated a total of 20 unique instances of the `RFO:Decision_Tree` class, each containing a prediction. Afterwards, as this RF utilised a majority voting strategy, the modal value of the 20 classifications was computed to derive the final classification for each `Roll_Unit_Trip` individual. This final classification was then stored in the ontology. Finally, all intermediate values, including the decision tree instances, were purged from the ontology, resulting in a streamlined and concise representation of the final classifications.

5.4.5 Limitations and Validation

The proposed method is computationally expensive for large RFs or large datasets. In our study, the sci-kit learn model was using the remaining 20% of the dataset (1957 samples) for validation, which, when converted into rule-base reasoning, was too large for the default SWRL-API reasoner `Drools`. Therefore, we instead compared the accuracies of a batch of 100 data points iteratively, which overall produced identical results for all data points to the sci-kit learn validation, validating our approach. More concretely, a total of 25,657 paths in the RF translated to 25,657 SWRL rules that were passed to a rule engine. Each

rule was applied to the batch of 100 instances, producing a total of 2,565,700 inferences. Meanwhile, the authors in [187] compared the performance of different rule engines and concluded that Drools is optimised for smaller datasets and has the worst performance with larger datasets when compared to other reasoners. A possible solution for this performance issue is to investigate the use of a different rule engine that is compatible with SWRL-API or investigate optimisation methods to the rule generation process.

5.4.6 Integrating Domain Expert Knowledge

As mentioned, numerous industrial processes within the steel domain heavily rely on knowledge, where plant operators constantly make important decisions based on the scenario and their expertise. Meanwhile, the framework combines domain expert knowledge with ML classification to offer decision-making assistance for the human in the loop. The initial step involves acquiring expert knowledge through one or more well-known knowledge acquisition methods. This knowledge must be captured in a format that is translatable into an SWRL format for compatibility with the rule engine.

Although experts can be wrong, in the context of expert-based systems, expert rules are treated as highly accurate [120].

Knowledge Acquisition Methods

Domain expert knowledge is often considered to be of implicit and tactic nature which contradicts ontologies explicit modelling behaviour [167]. In the context of manufacturing, tacit knowledge refers to the concept of informal learning by simply performing actions and experiences, often where the knowledge is unconsciously retained in individual memory rather than being formally recorded or shared [167]. To overcome this phenomenon, many different ways of extracting tacit knowledge have been widely studied over the years. This includes techniques such as interviewing, questionnaires, protocol analysis, inferential flow analysis, and many more [174].

For our study, we conducted interviews and questionnaires with domain experts and plant operators to construct our domain expert knowledge rule set. Presently, we obtained a small sample of domain expert rules which we aim to increase in quantity and quality over time. Meanwhile, the small sample of domain expert rules demonstrates the capabilities of the framework. Table 5.1 displays some expert rules in a categorised format before they are translated into the SWRL format. Example 5.3 displays a domain expert knowledge obtained during the knowledge acquisition sessions which we translate as rules. As these rules come directly from knowledge acquisition sessions with domain experts, the validity and accuracy of these rules are treated as absolute. In the event that a rule creates an incorrect status, it is reviewed with experts and updated accordingly.

Example 5.3 'For any trips considered to have the 'Bad' condition, if that trip reaches a high level of tonnage and the work rolls recently had severe grindings in their last five trips, then the stopping of operation should be considered.'

ID	Classification	Recent Severe Grinding	Tonnage Limit	Tonnage Reached	Status
1	Best	False	4500	False	Continue rolling: predicted condition is best on a healthy roll. Recommended tonnage of 4500.
2	Best	False	4500	True	Stop rolling: predicted condition is best on a healthy roll. Recommended tonnage of 4500 has been exceeded.
3	Best	True	4000	False	Continue rolling: predicted condition is best but recent high stock removal may affect roll. Recommended tonnage of 4000.
4	Best	True	4000	True	Stop rolling: predicted condition is best but recent high stock removal may affect roll. Recommended tonnage of 4000 has been exceeded.
5	Good	False	4000	False	Continue rolling: predicted condition is good on a healthy roll. Recommended tonnage of 4000.
6	Good	False	4000	True	Stop rolling: predicted condition is good on a healthy roll. Recommended tonnage of 4000 has been exceeded.
7	Good	True	3500	False	Continue rolling: predicted condition is good but recent high stock removal may affect roll. Recommended tonnage of 3500.
8	Good	True	3500	True	Stop rolling: predicted condition is good but recent high stock removal may affect roll. Recommended tonnage of 3500 has been exceeded.
9	Bad	False	3000	False	Continue rolling: predicted condition is bad on a previously healthy roll. Recommended tonnage of 3000.
10	Bad	False	3000	True	Stop rolling: predicted condition is bad on a previously healthy roll. Recommended tonnage of 3000 has been exceeded.
11	Bad	True	2500	False	Stop rolling: predicted condition is bad on a roll that was previously damaged.
12	Bad	True	2500	True	Stop rolling: predicted condition is bad on a roll that was previously damaged. Recommended tonnage of 2500 has been exceeded.

Table 5.1: The domain expert knowledge rules in a simplified view.

Here, we have to explicitly define 'high tonnage' and 'severe grindings' with the help of domain experts. In this case, high tonnage is a dynamic value that changes thresholds depending on the condition of the roll. Meanwhile, the condition of the work roll can be calculated by looking at the previous grindings and whether it has recently had any `severe grindings`. A roll grinding is considered `severe` if any of the last five grindings of the work roll had a stock removal value greater than 0.5 mm and therefore require a lower tonnage threshold. Additionally, the historical data of roll change reasons are also considered as part of the domain expert rule set. A severe grinding could be caused by a mill incident where the impact on the health of the roll is much greater. This knowledge is not captured in the RF but is included as expert rules.

Listing 5.2 displays the rule in Example 5.3 in its SWRL format. These rules are man-

ually translated into the SWRL format. Afterwards, they were passed into the rule engine, which combined the RF prediction with the expert knowledge, producing new knowledge stored in the data property `hasStatus`. The status produced is displayed using SPARQL, a well-known semantic-based querying language that enables querying based on RDF triples and graph data [246]. Listing 5.3 is a SPARQL query, which when executed, prints the operational trips and their corresponding classification and status.

Code Listing 5.2: Example of a domain expert SWRL rule.

```
Roll(?r) ^ hasRollUnitTrip(?r, ?trip) ^ hasInferredCondition(?trip, ?condition) ^
swrlb:equal(?condition, "Bad") ^ hasRecentSevereGrinding(?r, ?severe) ^
swrlb:equal(?severe, true) ^ hasTripTonnage(?trip, ?tonnage) ^ swrlb:lessThan
(?tonnage, 2500) -> hasStatus(?trip, "Highest risk: predicted condition is
bad on a roll that was previously damaged. Recommended tonnage of 2500.")
```

Code Listing 5.3: SPARQL query to retrieve results.

```
PREFIX : <http://www.semanticweb.org/new/ontologies/2023/1/SCRO#>
PREFIX rf: <http://www.semanticweb.org/sadeer/ontologies/2023/0/RF#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX swrl: <http://www.w3.org/2003/11/swrl#>

SELECT ?trip ?classification ?status
WHERE {
    ?roll :hasRollUnitTrip ?trip .
    ?trip :hasRollRefurbConditionInferred ?classification .
    ?trip :hasTripTonnage ?tonnage_of_trip .
    #if any of the last 5 trips contain a stock removal value >0.5
    ?roll :hasRecentHighStockRemoval ?high_stock_removal .
    ?trip :hasStatus ?status .
}
GROUP BY ?trip ?classification ?status
```

5.5 Results and Validation

The purpose of this section is to validate the framework by contextualising it with the results obtained from the application, as illustrated in the use case presented in the preceding section. This validation process aims to assess the effectiveness and reliability of the proposed framework in practical scenarios, providing an evaluation of its performance and utility.

Figure 5.5 is a snippet that displays some results from the last 100 operational trips, which are used to test and validate the framework. These results display the knowledge-guided decisions through the `Status` column, as well as the classifications from semantic-based reasoning in the `classification` column, which are both displayed to the operator. As shown, each classification may be one of three categories, where the status provides a decision on what to do based on expert knowledge and the collected data at the specific timestamp of operation. The accuracy of the RF model is 78%, whereas the accuracy of

the decision-guided assistance is calculated and validated with the help of domain experts in the following subsection using qualitative methods.

SPARQL results | SQL translation

trip	classification	status
:roll_1647_trip_67061	"Bad"	"Stop rolling: predicted condition is bad on a roll that was previously damaged. Recommended tonnage of 2500 has been exceeded."
:roll_1647_trip_66936	"Best"	"Continue rolling: predicted condition is best but recent high stock removal may affect roll. Recommended tonnage of 4000."
:roll_1647_trip_66028	"Bad"	"Stop rolling: predicted condition is bad on a roll that was previously damaged. Recommended tonnage of 2500 has been exceeded."
:roll_1647_trip_65846	"Good"	"Stop rolling: predicted condition is good but recent high stock removal may affect roll. Recommended tonnage of 3500 has been exceeded."
:roll_1647_trip_65314	"Bad"	"Stop rolling: predicted condition is bad on a roll that was previously damaged. Recommended tonnage of 2500 has been exceeded."
:roll_1647_trip_64277	"Best"	"Continue rolling: predicted condition is best but recent high stock removal may affect roll. Recommended tonnage of 4000."
:roll_1647_trip_60465	"Best"	"Continue rolling: predicted condition is best but recent high stock removal may affect roll. Recommended tonnage of 4000."
:roll_1647_trip_59983	"Bad"	"Stop rolling: predicted condition is bad on a roll that was previously damaged. Recommended tonnage of 2500 has been exceeded."
:roll_1647_trip_59682	"Bad"	"Stop rolling: predicted condition is bad on a roll that was previously damaged. Recommended tonnage of 2500 has been exceeded."
:roll_1639_trip_66078	"Best"	"Continue rolling: predicted condition is best on a healthy roll. Recommended tonnage of 4500."
:roll_1639_trip_64659	"Bad"	"Stop rolling: predicted condition is bad on a previously healthy roll. Recommended tonnage of 3000 has been exceeded."
:roll_1639_trip_64367	"Good"	"Continue rolling: predicted condition is good on a healthy roll. Recommended tonnage of 4000."
:roll_1639_trip_60117	"Bad"	"Stop rolling: predicted condition is bad on a previously healthy roll. Recommended tonnage of 3000 has been exceeded."
:roll_1639_trip_59258	"Good"	"Continue rolling: predicted condition is good on a healthy roll. Recommended tonnage of 4000."
:roll_1639_trip_58274	"Bad"	"Stop rolling: predicted condition is bad on a previously healthy roll. Recommended tonnage of 3000 has been exceeded."
:roll_1639_trip_57965	"Bad"	"Continue rolling: predicted condition is bad on a previously healthy roll. Recommended tonnage of 3000."
:roll_1639_trip_57893	"Good"	"Continue rolling: predicted condition is good on a healthy roll. Recommended tonnage of 4000."
:roll_1639_trip_57416	"Bad"	"Stop rolling: predicted condition is bad on a previously healthy roll. Recommended tonnage of 3000 has been exceeded."
:roll_1638_trip_66551	"Bad"	"Stop rolling: predicted condition is bad on a previously healthy roll. Recommended tonnage of 3000 has been exceeded."
:roll_1638_trip_66441	"Best"	"Stop rolling: predicted condition is best on a healthy roll. Recommended tonnage of 4500 has been exceeded."
:roll_1638_trip_66127	"Good"	"Continue rolling: predicted condition is good on a healthy roll. Recommended tonnage of 4000."
:roll_1638_trip_66003	"Bad"	"Continue rolling: predicted condition is bad on a previously healthy roll. Recommended tonnage of 3000."
:roll_1638_trip_63403	"Bad"	"Continue rolling: predicted condition is bad on a previously healthy roll. Recommended tonnage of 3000."
:roll_1638_trip_62805	"Best"	"Continue rolling: predicted condition is best on a healthy roll. Recommended tonnage of 4500."
:roll_1638_trip_61717	"Best"	"Continue rolling: predicted condition is best on a healthy roll. Recommended tonnage of 4500."
:roll_1638_trip_60542	"Good"	"Continue rolling: predicted condition is good on a healthy roll. Recommended tonnage of 4000."

Figure 5.5: The SPARQL query results.

Validation

The purpose and contribution of the framework is to provide improved assistance for the human in the loop. The classifications of the RF model produced `good`, `bad`, and `best` outputs, whereas the final `status` output provided new decision-making knowledge on whether to continue the rolling process, with an estimation of how much more rolling was recommended based on classification and expert knowledge.

Because of this, there was no ground truth or direct labels to compare for validation. However, there are many well-studied validation techniques for situations where no ground truth is available, often deployed during unsupervised learning models, such as interval validation, external validation, domain expertise, twin-sample validation, and cross-validation [235]. In our use case, we adopt external validation and domain expertise validation methodologies to validate our framework. These validation methods have been applied in various domains [268, 76].

Similarly, we followed an iterative screening and refinement process with domain experts and stakeholders to share results and obtain valuable feedback, as highlighted in Figure 5.6.

Within the iterative cycle, the results are displayed, discussed, and validated with the domain experts. The expert rules are refined with any newly obtained knowledge from these instances. Once the rule set is recompiled, it is passed through the semantic reasoner again, producing new results which are displayed to the experts once more, repeating the iterative cycle. By utilising domain experts for validation, we can ensure the accuracy, relevance, and robustness of the framework, providing insights and tactic knowledge that are not apparent from data alone.

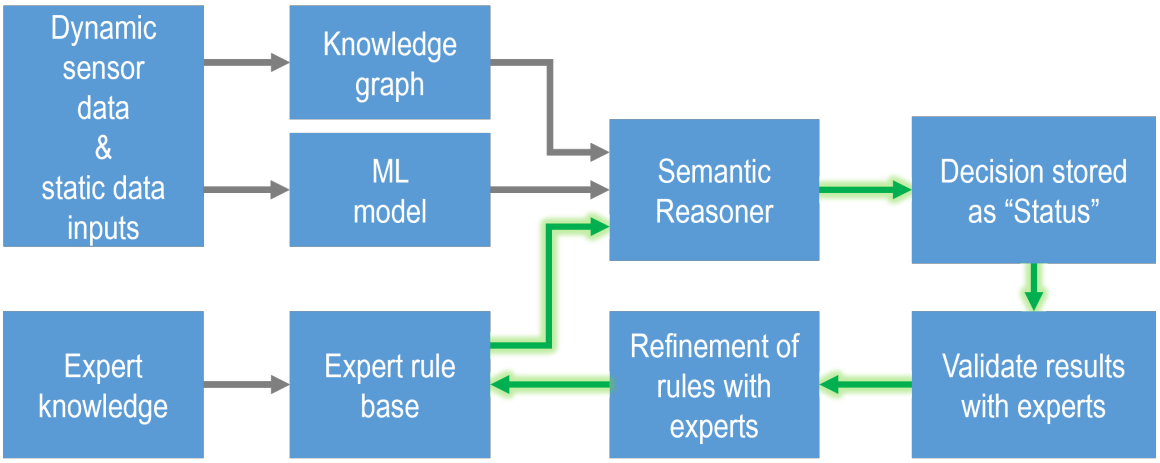


Figure 5.6: The methodology of the iterative process with experts for validation.

With each iteration, we displayed the one hundred data entries. These entries contained the status, classification, expert rules, sensory data, and the actual stock reduction of the roll. Furthermore, we categorised the entries into two groups: (1) expected output consisting of 80% of the entries where the status of the entry matched the amount of stock reduction for that roll, e.g., "stop rolling" for entries where the stock reduction was greater than the expected average value, and (2) unexpected output where 20% of entries where the opposite interaction occurred. All entries were displayed to the experts; however, it would be revealed to be a time consuming task to iterate through all the entries, so we further refined and handpicked the most interesting results and compared the predictions with those of domain experts.

For the expected output category where the majority of data points were, ten entries were displayed to the experts. All entries were studied and approved by the experts, stating that they would have made the same decision based on the data. Two out of the ten entries are displayed in Table 5.2. This confirmed that the support provided by the framework was accurate for the expected category.

Similarly, ten entries from the unexpected output category were also displayed to the experts. In three of those cases, the status was 'stop rolling' despite a low diameter reduction and an accurate 'good' classification. In these cases, an expert rule was triggered that prioritised stopping operation if there was a high tonnage produced on a pair of work rolls that were recently damaged. Having investigated these results with domain experts, we discovered that there was a 'pinch' on one of the rolls, which is required to be removed before further operation. This positive feedback confirmed that the framework was able to provide accurate results in these cases.

Meanwhile, there were three instances where the status produced was 'stop rolling' by a similar expert rule. However, the experts stated that although the expert rule was correct, some knowledge was missing, stating that work rolls in Stand 3 are expected to handle more work load and pressure. The experts began to describe the refinement process in more

Table 5.2: Two reports from the 'expected' category.

Report ID	01
Roll and Trip ID	roll_1728_trip_67496
Category	Expected.
Trip details	(1) Trip has a high tonnage threshold; (2) Work roll had a high stock reduction in the last five refurbishments.
Classification	Bad condition.
Status	Stop rolling: predicted condition is bad on a roll that was previously damaged. Recommended tonnage of 2500 is exceeded.
Expert comment	'The status is accurate. The roll has done high tonnage in the current trip and should be taken out. Looking at the historical data of roll 1728, it had a significant cut in stock recently and should be treated carefully. If it was removed earlier, the stock reduction would be lowered.'
Report ID	02
Roll and Trip ID	roll_1609_trip_67249
Category	Expected.
Trip details	(1) Trip had medium tonnage; (2) Work roll is brand new and had no recently high stock reductions.
Classification	Best condition.
Status	Continue rolling: predicted condition is best on a healthy roll. Recommended tonnage of 4500.
Expert comment	'The status is accurate to continue rolling. This roll is new and is expected to roll the maximum amount possible. The recommended tonnage could have been slightly higher than 4500 before being refurbished.'

detail, explaining how work rolls in different stands have different tonnage expectations and require different handling, which we adapt into our expert knowledge rule base for the next iteration. Finally, the remaining four instances provided an inaccurate status as the ML classification inaccurately predicted the roll condition. Two entries from the unexpected category are displayed in Table 5.3.

For the second iteration, we refined our expert knowledge rule set to include the newly obtained knowledge of stand information, which we displayed to the domain experts again. Once more, we produced two categories of results in the same manner as the first iteration. We first revisited the same entries from the first iteration. This time, the entries that were in Stand 3 produced a more accurate 'continue rolling' status, aligning with the expectations of the experts. Additionally, we continued to review ten new outputs for the second iteration for each category. This process established new knowledge regarding grinder stone diameter and roll hardness that can be used to improve the quality of the expert rules, followed by further iterations of the process if required.

From the qualitative validation, the accuracy of the expected category results for the first and second iterations were both 100%. Although each iteration had ten entries, it offered confidence to the experts using the system as 80% of the overall entries were in the

expected category. Meanwhile, the accuracy of the unexpected column improved after the refinement of the expert rules in the second iteration, and it can be improved with further refinement.

Table 5.3: Two reports from the 'unexpected' category.

Report ID	03
Roll and Trip ID	roll_1534_trip_67845
Category	Unexpected.
Trip details	(1) Trip had high tonnage; (2) Work roll had a significant stock reduction in the last five refurbishments.
Classification	Good condition.
Status	Stop rolling: predicted condition is good but recent high stock removal may affect roll.
Expert comment	'The status is accurate. After investigating roll 1534, there was a pinch on the work roll which needed to be removed before further operations.'
Report ID	04
Roll and Trip ID	roll_1647_trip_65846
Category	Unexpected.
Trip details	(1) Trip had high tonnage; (2) Work roll had a significant stock reduction in the last five refurbishments.
Classification	Good condition.
Status	Stop rolling: predicted condition is good but recent high stock removal may affect roll. Recommended tonnage of 3500 is exceeded.
Expert comment	'The status is not completely accurate for this roll. This is because the roll is in stand three. Rolls in stand three are expected to do more total tonnage than other stands before being refurbished, and are expected to withstand stronger forces. The expected result in this situation is to continue rolling for a few hundred more tons.'

In addition, the experts we engaged with provided overwhelming positive feedback of the framework in general. One expert said that it is reassuring having a second opinion on difficult decision-making situations, where usually they may consult a fellow worker or manager for a second opinion. Meanwhile, another expert claimed that the framework has very good potential with the improvement of stronger expert rules and machine learning models. Finally, the iterative process itself was perceived as useful: it enabled interaction and continuous improvement of the decision-making tool for the experts, while also providing us as non-experts with greater domain knowledge and understanding of the cold rolling processes.

Overall, the results demonstrated the capabilities of the framework, and its ability to assist operators with decision-making tasks. One fundamental contribution of the proposed framework is the ability to encode RF classification semantically so that the meta-data is also included in the model, which can be combined with external knowledge for further assistance in decision-making. Our results used real-world examples to demonstrate that

such an approach is possible and advantageous. Ultimately, the knowledge is provided to assist and support the operator with their decision-making tasks.

5.6 Conclusions

This chapter introduces a human-in-the-loop framework that combines a random forest model with semantic technologies where the resulting semantic ML classification is enhanced by domain expert knowledge. There are two key contributions and components of the framework: (1) the Random Forest Ontology that models the concepts of a random forest, where an RF-based knowledge graph can be generated as an alternative way to encode the RF classification process and knowledge using rule-based reasoning, and (2) the integration of expert knowledge with the classification to provide semantically enriched, knowledge-guided decisions as recommendations for the human in the-loop system . A use case for predictive analytics in smart manufacturing is demonstrated that uses real-world data from industrial partners, which displays the capabilities, applicability, advantages, and limitations of the proposed framework.

Within smart manufacturing, a fundamental goal is to improve machine interoperability and interpretability; this chapter proposes a method of improving machine interpretability via semantic technologies, providing one example of how ML models can be represented using semantic technologies.

5.7 Future Work

The major constraint of the proposed approach was the capabilities of the chosen rule engine. Drools, the default reasoner of SWRL-API, struggled to provide logical inferences on a very large dataset and hence limited the size of the knowledge graph. One future goal is to investigate the different available reasoners that have stronger inference capabilities to speed up and stabilise the inference process. On the other hand, the quality of the framework depends on the quality of the ML model and the quality of the training set, as well as the quality of the domain expert knowledge. Presently, we aim to hold more knowledge acquisition sessions to improve the quality of the expert rules. It is also necessary to take the time to evaluate and further validate the impact and effectiveness of the framework with the industrial partners after extended usage.

CHAPTER 6

Towards a Framework for Continuous Streaming with Neurosymbolic AI

Contents

This chapter displays the early work towards developing a semantic framework for simulating data streams, where reasoning is applied onto data streams in real-time, incorporating a range of rules and classifications.

In particular, we demonstrate a use case for steelmaking predictive maintenance, where random forest classification and expert knowledge is integrated and applied onto the data stream semantically, enabling reasoning on the fly.

6.1 Introduction

The continuous production of dynamic data collected from data streams, including sensors and networks, is known as *stream data* [80]. Due to the widespread adoption of digitisation

in manufacturing, many processes generate a surplus amount of data, generally captured and stored for various purposes, such as predictive tasks, monitoring, checks, and many others. These data elements are derived from time-varying unbounded sequences of data [80], often obtained from heterogeneous data sources. Reasoning with this kind of data is known as Stream Reasoning (SR) [19].

In many cases, domain knowledge is necessary for extracting meaningful insights from the data streams [42]. We have previously demonstrated how knowledge-driven AI technologies, such as ontologies, play a significant role for modelling domain knowledge and enable the integration of heterogeneous data within a manufacturing system [43]. In this chapter, we aim to expand our finding, and apply novel methodologies on data streams, simulating the real-world nature of operations in steelmaking.

In SR, input data is dynamic and can be infinite. To manage the unboundedness of the data, stream reasoners require a *window operation*, typically in the form of snapshots, which collect and segment the generated data [208]. There are two fundamental windows, *Time-based* windows, where input data is calculated within a fixed amount of time, or *Tuple-based* windows, where input is limited by a specific quantity of input data [238].

In this chapter, we utilise semantic methodologies for applying reasoning on data streams, specifically for predictive maintenance purposes in cold rolling. Predictive maintenance is an important aspect in manufacturing, attempting to predict errors or future faults of systems, enabling actionable maintenance before failure occurs [210]. With the ongoing trends of Industry 4.0, the ability to collect and process sensor data has significantly enhanced many predictive maintenance operations [243]. Given the inherent nature of operations and strong forces involved in cold rolling, work rolls inevitably suffer from roll wear and will eventually need refurbishing.

There exist machine learning models that are designed for data streams, including Hoeffding Trees [182]. These models apply incremental learning, where the dataset is updated incrementally. However, in this chapter, we aim to deploy the methods we developed in our previous chapter in a continuous manner. Thus, we stick with the deployment of random forests.

We set up a use case, providing one example of how random forest classification, as well as domain expert knowledge can be applied on data streams to predict the optimal timing to replace the work rolls before exceeding the limit of the work rolls. This novel approach enables real-time condition monitoring based on dynamically generated data streams, which conventionally require pausing operations for manual checks. The contributions of this chapter lie in the development of a framework, specifically for applying neuro-symbolic AI on data streams, that is, the combination of statistical machine learning methods with semantic technologies. To the best of our knowledge, there is currently no existing framework that seamlessly applies machine learning classification, while also incorporating domain-expert knowledge on top of the resulting classification on the fly. We demonstrate a method where these two models are combined for reasoning, without the need of pausing operations.

6.2 Related Work

In this section, we look at related literature that have demonstrated success in continuous streaming and reasoning, highlighting the key technologies and languages used for each implementation.

Chandrasekaran et al. [64] were among the first to develop a framework for processing continuous dataflow. They introduced TelegraphCQ, a dataflow system designed for processing continuous queries over data streams. Developed at the University of California at Berkeley in 2003, TelegraphCQ showcased the ability to (1) create data streams, (2) establish sources for simultaneously pushing data to the same data stream, (3) create wrappers enabling user-defined functions such as `init, next` or `done` to process the data from external sources, and (4) support continuous queries, allowing users to issue long-running continuous queries over defined time windows [181].

Walavalkar et al. [323] deployed the reasoning capabilities of TelegraphCQ and combined them with the Jena Rule Engine for reasoning in 2008. The authors demonstrated how this combination of technologies can perform RDFS inferences, utilising the Spire project ontology to classify different plants and invasive species.

Barbieri et al. [20] have developed the Continuous-SPARQL (C-SPARQL) framework in 2009, an extension of SPARQL that supports continuous queries over RDF data streams. The main features include continuous queries, where a data stream can be continuously monitored, and any results are printed as the data changes [21]. In the paper, the authors introduce *Windowing*, a feature that allows for temporal constraints on the data. For example, the user can specify a time window that specifically looks at the last five minutes of data that moved over the data stream.

Anicic et al. [11] has developed the Event Processing-SPARQL (EP-SPARQL) framework in 2011, a unified language for event processing and stream reasoning. This language extends the SPARQL language, providing features for event processing and stream reasoning capabilities using Logical Programming. EP-SPARQL is able to model complex events by leveraging temporally situating real-time streaming data, deploying ontologies to enable stream reasoning. EP-SARQL focuses more on detecting RDF triples in a specific temporal order compared to C-SPARQL, which is typically determined on a static RDF based on a query.

Özçep et al. [232] developed the Spatial and Temporal ontology Access with a Reasoning-based Query Language (STARQL) framework for accessing and querying heterogeneous sensor data through ontologies. The initial research focused on creating a querying tool for Ontology Based Data Access (OBDA), while STARQL is extends the OBDA framework by integrating traditional data queries with stream processing. STARQL consists of two main components: (1) an ontology that models the semantics of data and (2) an Embedded Constraint Language (ECL) for composing queries [81]. As part of this extension, the authors expand on the sliding window concept, providing window operations such as clauses for expressing event matching and the integration of static and streamed data. Inference relies on temporal annotations of the streamed data, created using a sequence of time-

annotated ontologies.

Mileo et al. [207] developed StreamRule, a non-monotonic stream reasoning system for the semantic web. Their methodology first adopts a stream query processor, which abstracts and filters the raw streaming data using declarative query patterns as filters. The filtered stream is passed onto a middle-layer processor, which processes matching patterns to generate input facts. These facts are subsequently passed onto a non-monotonic stream reasoner, along with a declarative encoding of the problem at hand, producing a set of solutions as output. The filtering process uses the Continuous Query Evaluation over Linked Stream (CQELS) Java engine, which supports continuous query evaluation over streams of RDF data.

Beck et al. [24] developed the Logic-based framework for Analysing Reasoning over Streams (LARS) framework for stream reasoning. The LARS framework provides a formal model of streams with built-in modalities and window constructors that can be applied to streams. LARS also provides two languages for reasoning over streams, (1) LARS Formulas, a language using first-order formalism to reason over extensional data only, and (2) LARS Programs, for more involved applications that necessitate reasoning over auxiliary or intentional predicates, expressing and analysing stream reasoning.

Jajaga et al. [162] developed a stream reasoning framework for monitoring the quality of water on the fly. They propose a conceptual architecture named *StreamJess*, which consists of four layers: data, ontology, stream filter and aggregation, and rules layer. StreamJess is implemented using Java, allowing users to import an ontology, build a working memory, and provides a rule engine for inference tasks. The rules in this framework have to be in Jess format instead of SWRL, as the authors state that SWRL rules lack the required expressivity level to reason over stream data. Their streaming process adopts C-SPARQL querying, enabling filtering and aggregating of the data streams. Once the stream data has been filtered and aggregated, the rule engine applies the Jess rules to the newly published temporary observation facts, producing an output for classifying individual pH observations into appropriate Water Framework Directive (WFD) statuses. The authors extend their work in [161], where they replaced the JessRule engine with the Continuous-SWRL (C-SWRL) engine, leveraging SWRL-API, a Java library that extends OWL-API, to continuously apply SWRL rules. In this chapter, we follow a similar process of adopting C-SPARQL to query the data stream, and use SWRL-API to capture and continuously apply our rule sets to the data streams. In our application, we have two rule sets, one denoting random forest classification and another denoting expert rules which are applied on top of the generated classification. Both rule sets are applied to the streamed data on the fly for predictive maintenance purposes.

6.3 Methodology

The methodology of the proposed application can be divided into two main sections, as displayed in Figure 6.1. On the left-hand side, we have the static elements of the framework. This includes the process of training a Random Forest (RF) model for classification. To

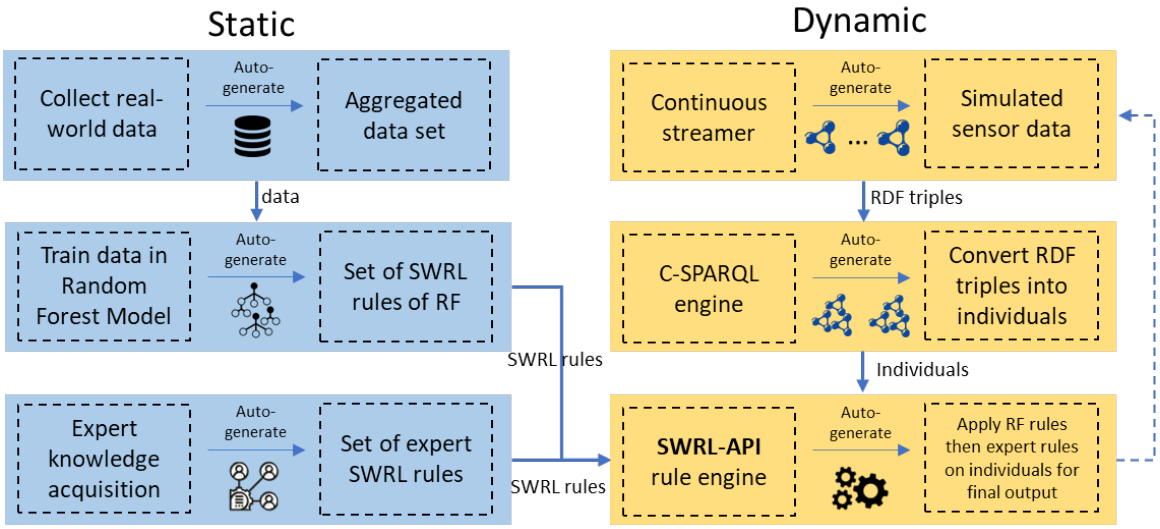


Figure 6.1: The methodology of the proposed application.

achieve this, the real-world dataset must be collected, aggregated, and passed as training data to the RF model. Once this RF is trained, it is converted into a set of SWRL rules using the semantic methods mentioned in the previous chapter. Once the RF is represented as a set of rules in SWRL format, the RF classification process can be applied to the generated individuals in the ontology in real-time using rule-base reasoning. Additionally, the remaining part of the static section is the acquisition of expert rules. These additional rules can be applied on top of the RF classification to generate new insights and actionable decisions.

On the other hand, the dynamic section illustrates the continuous streaming and reasoning process. The first step involves setting up a continuous streamer, which in our case, is achieved by setting up a Java project and utilising available online libraries. This streamer automatically generates data in Resource Description Language (RDF) format, simulating realistic real-world data to replicate the ongoing operational processes. In our example, this process replicates the cold rolling process and data, attempting to determine when a work roll is considered "damaged". Subsequently, we employ the C-SPARQL engine, capable of collecting the generated RDF triples and converting them into unique individuals. This is achieved by capturing a snapshot within a time window e.g., 40-second intervals, and accumulating the RDF triples within that timestamp using a SPARQL query. Finally, we use the SWRL-API library, which contains a SWRL rule engine, to apply the RF rule set to the individuals, generating a classification for each. Once this classification is produced, it undergoes another round of inference in the rule engine, combining the external set of SWRL rule representing the domain and expert rule set. This entire process is continuous; while the C-SPARQL engine and the SWRL-API rule engine are executing, the streamer continually generates a new batch of RDF data for the next iteration.

6.4 Use Case and Data

Work rolls get worn during cold rolling processes. When they are worn, they are considered as damaged and need to be refurbished to remove the worn surface. To adapt to the roll wear, it is necessary to predict the optimal maintenance schedule for the rolls.

To accurately predict the optimal, automated maintenance schedule for the rolls, it is essential that all the impacting factors that contribute to roll wear are properly understood. To achieve this, several datasets have been collected from Tata Steel UK, and aggregated with the aim of linking the production information and chemical compositions of the coils to the reasons for changing the rolls. This new dataset, we call *Roll_Unit_Trips_aggregated (RUTA)*, contains: (1) sensor information recorded with high resolution, (2) production data including the chemical composition of the coils, (3) historical refurbishment records, and (4) tracking of the life-cycle of each roll. The semantics of these values can be viewed in Table 6.1.

RUTA contains a *trip sequence* as the primary key, as well as the trip time start and trip time finish in the form of a date. These dates are used to link dynamic sensory data with existing static data from the specifications of the coil suppliers. Other key features include roll data, trip tonnage and meterage, as well as all the chemical compositions of the coils. All these features are captured and integrated into the SCRO ontology as data properties. In total, there are 120 properties that are used as features for the RF classifier which will be used on the generated individuals. Some of these are displayed in Table 6.1.

Table 6.1: The data properties used for classification.

Property	**Min Value**	**Max Value**
hasMillID	1	2
hasStandID	2	7
hasTopRollID	5	2000
hasTopRollGrindingSequenceNo	1	300
hasTopRollChockingSequenceNo	1	300
hasBottomRollID	5	200
hasBottomRollGrindingSequenceNo	1	300
hasBottomRollChockingSequenceNo	1	300
hasTripTonnage	0	9000
hasTripMeterage	0	1000
hasRollChangeReason	1	200
hasChangeReasonID	1	200
hasDamageID	1	13
hasCoilpartLengthExit	100	5000
hasCoilpartThickExit	0.3	3
hasGradeFiltered	2500	5000
hasExitGauge	0.3	3
hasExitWidth	800	2000

Table 6.1: Properties table, continued on next page

Table 6.1: The data properties used for classification.

Property	**Min Value**	**Max Value**
hasExitWeight	0.7	30
hasInputCoilID	1114000	9000000
hasHRGauge	2	5
hasCRGauge	0.3	3
hasInputWidth	850	2000
hasInputWeight	0	28
hasHeatNumber	51000	70000
hasC	0.001233333	0.158000007
hasSI	0.001	0.273749996
hasS	0.002	0.02
hasP	0.006	0.035075472
hasMN	0.066333331	1.785333316
hasNI	0.008	0.028
hasCU	0.00625	0.059
hasSN	0.001	0.017000001
hasCR	0.009666667	0.560000002
hasArsenic	0.001	0.005
hasMO	0.001	0.00325
hasN2	0.00175	0.007866667
hasALT	0.024	0.056333333
hasALS	0.0225	0.052333333
hasTI	0.001	0.062
hasB	0.000197619	0.003
hasCA	0.0003	0.004033333
hasCE	0.0015	0.00777
hasCO	0.002090909	0.008
hasNB	0.0005	0.02915625
hasV	0.001	0.006875
hasCEV1	0.0178	0.543612506
hasCEV2	0.015699999	0.543100003
hasCEV3	0.0127	0.429125007
hasFREEC1	0	0.167400003
hasFREEC2	0.001033333	0.157900006
hasXSTI1	0	0.0353
hasXSTI2	0	0.042811111
hasXSTI3	0	0.052700001
hasALN2	5.131599903	25.3366998
hasCUP	0.671450004	6.555600166
hasBN2	0.025625	1.515475005

Table 6.1: Properties table, continued on next page

Table 6.1: The data properties used for classification.

Property	Min Value	Max Value
hasSP	0.011255556	0.046555554
hasCUNICRMOSN	0.043333334	0.671000004
hasCREQ	92.08333333	2344.958333
hasSIFORM	0.017000001	0.322187502
hasMNS	9.260187929	648.1547668
hasMNSI	6.461200237	246
hasFREEC3	0	0.157900006
hasFE	97.09173965	99.79535711
hasLIQUIDUS	1510.02948	1535.459961
hasTonsRolledComputed	0	1300000
hasLengthRolledComputed	8.74	800000
hasCoilsRolledComputed	1	65000
hasWidthReducedComputed	0	300000
hasWeightReducedComputed	0	25000

In this framework, we apply constraint checking to ensure the generated instances contain realistic values between the min and max value for each data property.

6.5 Random Forest Creation

To build the RF classifier in our application, 75% of the original dataset (conformed by 3629 samples) was used as the training set. The train-test split was performed randomly and in such a way that the original proportion of damaged rolls to not damaged rolls (20% - 80%) was respected. The value of the hyperparameters `n_estimators, max_depth` and `min_samples_leaf` was set using grid search and validating the out-of-bag (OOB) score over the training set. The performance reached in terms of both standard and weighted metrics, where the positive class is the damaged class is as follows:

`Precision:`	0.83	`Weighted Precision:`	0.85
`Recall:`	0.41	`Weighted Recall:`	0.86
`F1-score:`	0.54	`Weighted F1-score:`	0.84

Lastly, the output generated a RF consisting of 200 DTs, which is stored in plain text format using the `export_tree` method provided by sci-kit learn. This RF has to be converted into a set of SWRL rules in order to be applied on data streams in real-time. Therefore, we utilise the same algorithm introduced in the previous chapter to automatically apply this conversation. One example of a path in the forest forest in SWRL format is represented in Listing 6.1. The antecedent of the rule states the necessary conditions for the consequent to apply, which provides the prediction.

Code Listing 6.1: Example of an SWRL rule.

```
Roll_Unit_Trip(?trip) ^ hasMillID(?trip, ?MillID) ^ swrlb:lessThanOrEqual(?MillID, "1.50"^^xsd:double) ^
    hasC(?trip, ?C) ^ swrlb:lessThanOrEqual(?C, "0.06"^^xsd:double) ^ hasTripMeterage(?trip, ?
    TripMeterage) ^ swrlb:greaterThan(?TripMeterage, "52.33"^^xsd:double) ^ hasP(?trip, ?P) ^ swrlb:
    greaterThan(?P, "0.02"^^xsd:double) ^ swrlx:makeOWLThing(?DT, ?trip) -> Decision_Tree(?DT) ^
    isDecisionTreeOf(?DT,?trip) ^ hasPrediction(?DT, "1.0"^^xsd:double) ^ hasTreeIndex(?DT, 1)
```

6.6 Application

This use case demonstrates the deployment of our framework in a cold rolling setting. Here, sensor data is simulated through a continuous data stream, which is collected using C-SPARQL querying at timed intervals, and then passed into a rule engine for reasoning. Algorithm 3 outlines the operational flow.

Algorithm 3: Speudocode of the streaming and reasoning process.

```
Import ontology
Create SWRL rule engine
Import RF rule set into rule engine
Import expert rule set into rule engine
Initialise C-SPARQL Engine
Define Query to retrieve specific streamed data
Register an RDF Stream
Start streaming
while streaming is running do
    Start timer for C-SPARQL window
    Generate RDF tuples continuously every 3 seconds
    if timer reaches 40 seconds then
        Use C-SPARQL query to filter RDF tuples {produces individuals}
        Apply SWRL-API rule engine to individuals {applies RF classification}
        Apply SWRL-API rule engine to individuals {applies expert rules}
        Reset timer
    end if
end while
```

Figure 6.2 displays the Java console that prints the output when the application is initiated. In this figure, we can see the two rule sets being imported before the streamer starts. Afterwards, a new message appears every few seconds for each batch of RDF triples generated. This happens continuously.

When starting the streamer, the C-SPARQL window is instantiated with a time frame set to 40 seconds. This value is flexible but we have chosen 40 seconds to provide enough

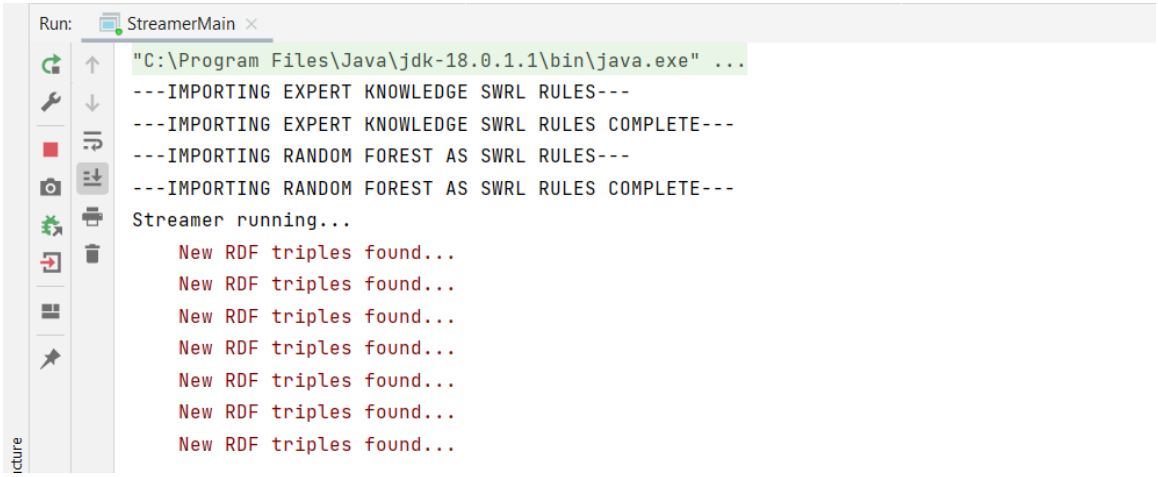


Figure 6.2: The application demonstrating the creation of RDF triples.

time to generate a handful of instances while also limiting this value to ensure the capacity of the reasoner is not reached. During this period, the streamer generates a batch of RDF tuples every few seconds, representing the generated RUTA data set. This conveys that a total of 120 RDF tuples are generated every few seconds, one for each data property. The value of each data property is randomly selected between the `min` and `max` values displayed in Table 6.1.

After the time window elapses, the application invokes the SPARQL query to retrieve all newly generated RDF triples from the data stream. A snippet of the SPARQL query is displayed in Listing 6.2. Instead of presenting all 120 properties, we reduce the size of the query in the listing to save space.

This SPARQL query introduces a new variable, represented by a question mark, for each property to be queried. This variable, such as `?MillID` for the `:hasMillID` data property or `?StandID` variable for the `:hasStandID`, is linked to the generated individual stored in the `?trip` variable. The query ensured that all 120 data properties are selected for each `?trip` individual, preventing any missing data.

Code Listing 6.2: SPARQL query to retrieve RDF triples.

```
REGISTER STREAM query AS
PREFIX : <http://www.semanticweb.org/sadee/ontologies/2021/1/Tatatology#>
SELECT ?trip ?MillID ?StandID ?TopRollID ?TopRollGrindingSequenceNo ?TopRollChockingSequenceNo ?
    BottomRollID ?BottomRollGrindingSequenceNo ?BottomRollChockingSequenceNo ?TripTonnage ?
    TripMeterage ?RollChangeReason ?InMill ?ChangeReasonID ?DamageID ?CoilpartLengthExit ?
    CoilpartThickExit ?GradeFiltered ?ExitGauge ?ExitWidth ?ExitWeight ?InputCoilID ?HRGauge ?
    CRGauge ?InputWidth ?InputWeight ?HeatNumber ?C ?SI ?S ?P ?MN ?NI ?CU ... ?TonsRolledComputed ?
    LengthRolledComputed ?CoilsRolledComputed ?WidthReducedComputed ?WeightReducedComputed
FROM STREAM <http://www.semanticweb.org/sadee/ontologies/2021/1/Tatatology#> [RANGE 40s TUMBLING]
WHERE {
  ?trip :hasMillID ?MillID .
  ?trip :hasStandID ?StandID .
  ?trip :hasTopRollID ?TopRollID .
```

```
  .
  . *snipped to save space
  .
  ?trip :hasLengthRolledComputed ?LengthRolledComputed .
  ?trip :hasCoilsRolledComputed ?CoilsRolledComputed .
  ?trip :hasWidthReducedComputed ?WidthReducedComputed .
}
GROUP BY ?trip ?MillID ?StandID ?TopRollID ?TopRollGrindingSequenceNo ?TopRollChockingSequenceNo ?
     BottomRollID ?BottomRollGrindingSequenceNo ?BottomRollChockingSequenceNo ?TripTonnage ?
     TripMeterage ?RollChangeReason ?InMill ?ChangeReasonID ?DamageID ?CoilpartLengthExit ?
     CoilpartThickExit ?GradeFiltered ?ExitGauge ?ExitWidth ?ExitWeight ?InputCoilID ?HRGauge ?
     CRGauge ?InputWidth ?InputWeight ?HeatNumber ?C ?SI ?S ?P ?MN ?NI ?CU ... ?TonsRolledComputed ?
     LengthRolledComputed ?CoilsRolledComputed ?WidthReducedComputed ?WeightReducedComputed
```

The results of the query are displayed in Figure 6.3. In this particular instance, from starting the streamer until the first time window, elapsing a total of forty seconds, a total of 13 unique individuals were generated and displayed. Each of these individuals have 120 data properties, enabling the simulation of the real-world dataset during cold rolling.

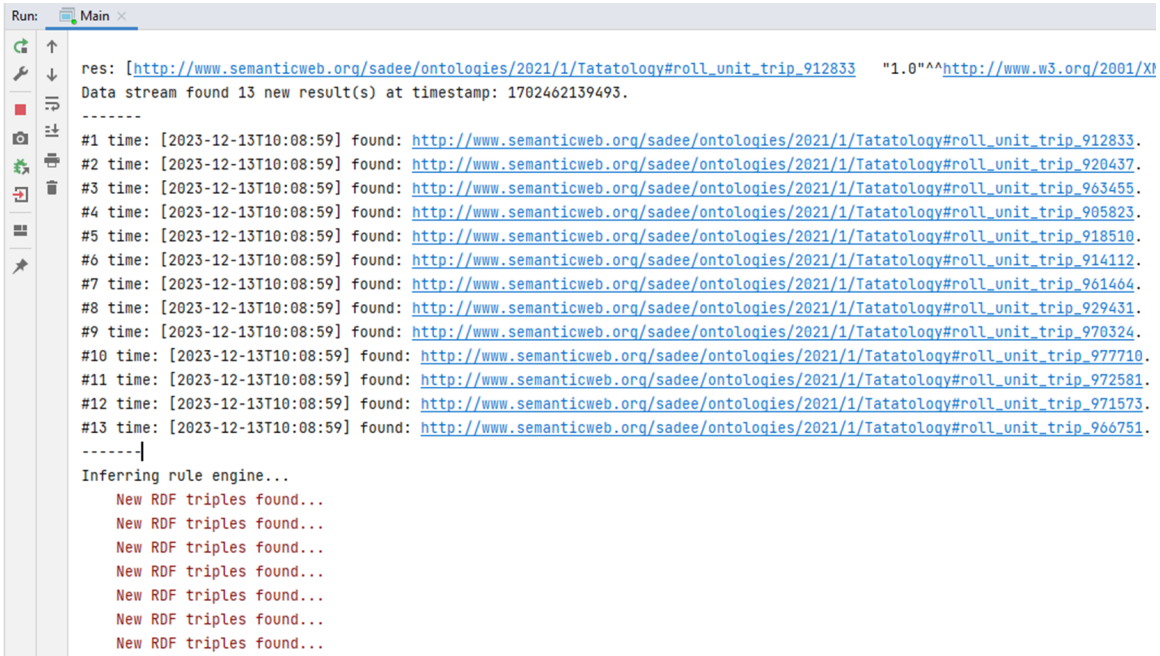


Figure 6.3: The application demonstrating the C-SPARQL query creating individuals from the RDF data.

Following the methodology workflow, the generated individuals are then passed into the rule engine, applying the two distinct set of SWRL rules. The first rule set processes the 120 data properties of each individual, employing rule-based reasoning to execute the RF classification. This results in a binary output indicating whether each roll is considered damaged, determined by the model. Afterwards, the rule engine is invoked once more to apply the domain expert rule set. These rules incorporate additional, domain-specific

knowledge, leveraging the RF classification output as an input. Hence, two successive iterations of the rule engine are essential: the first for RF classification and the second for the application of domain expert rules which incorporate the classifications as input.

In the background, simultaneous with the rule engines, new RDF triples are being generated for the next time window. Following the completion of the first round of rule engine execution, each created instance of the `roll_unit_trip` received a damage flag prediction: either a 0 for no damage or 1 for suspected damage. Figure 6.4 illustrates this output.

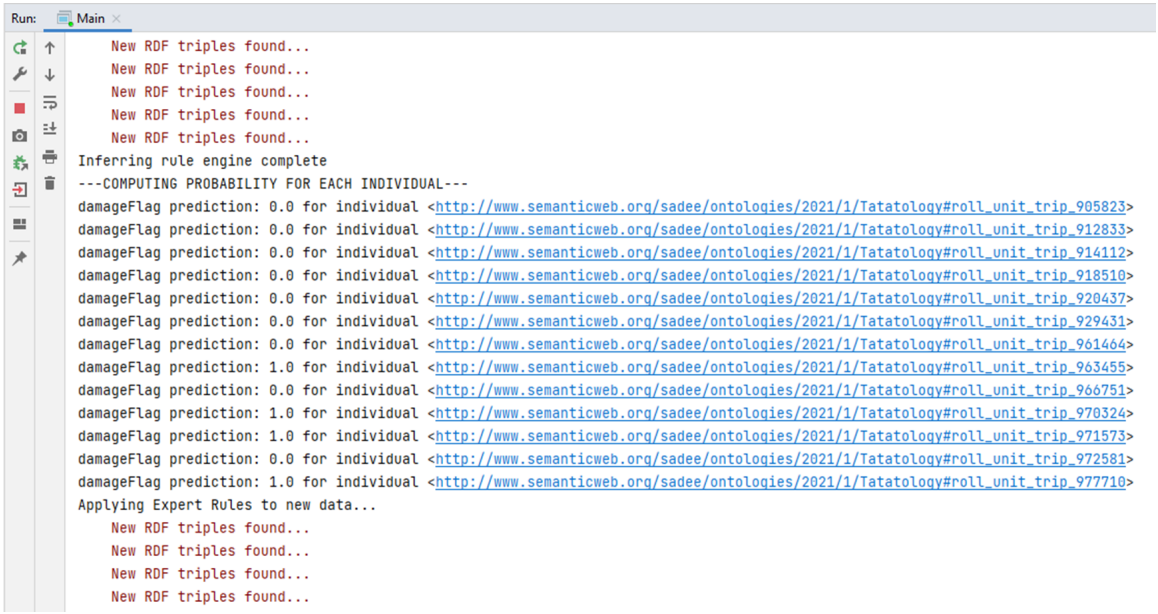


Figure 6.4: The application demonstrating the classification outputs of inferring the rule engine.

Subsequently, the domain expert rules are incorporated and executed by the rule engine. These expert rules use the previously generated classifications as input, integrating them with domain-specific knowledge to offer guidance or advice. In this instance, three expert rules are defined: (1) any damage flag of 0 shall return the result of “Continue rolling”, (2) any damage flag of 1 with low tonnage shall also return “Continue rolling”, (3) any damage flag 1 with high tonnage shall return “Stop rolling”. This configuration is depicted in Figure 6.5.

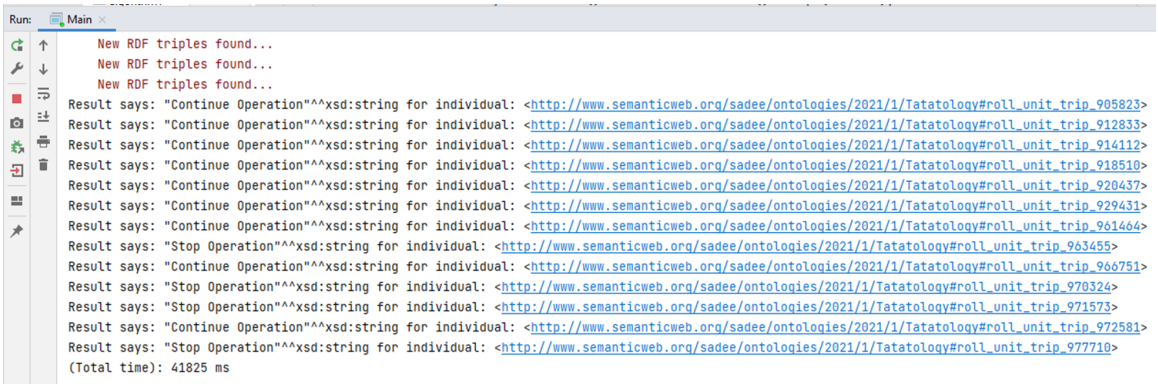


Figure 6.5: The application demonstrating the final results produced for each generated individual.

6.7 Conclusion and Future Work

This brief chapter introduces a novel proof-of-concept framework designed for processing a continuous data stream. We highlight a methodology for the continuous application of reasoning random forest classification and additional expert knowledge to data streams. The dynamic nature of the data stream simulates real-world dynamics in cold rolling processes, where these techniques find practical application for predictive maintenance purposes.

This chapter presents a use case of the framework's application in cold rolling, specifically in the continuous monitoring of roll conditions to predict potential issues. The purpose of such an application is to assist operators on the shop floor with enhanced knowledge and support for efficient decision-making.

The future work of this chapter entails further collaboration with domain experts to acquire a robust set of expert rules to replace the dummy rules employed to present this proof-of-concept. Additionally, upon obtaining these expert rules, the next step involves validating the framework.

Part III

Conclusions

CHAPTER 7

Conclusions, Reflections, and Future Work

Contents

7.1 Conclusions and Reflections

In the context of Industry 4.0, ongoing challenges are associated with achieving semantic interoperability among components and systems. For effective machine communication, a certain level of understanding of data and available data exchange protocols is necessary. While some international standards like OPC Unified Architecture or Automation Markup Language have been developed to address these requirement, achieving complete autonomy of operation remains an unsolved challenge.

To tackle these requirements, we have introduced the Semantic Web, an extension of the World Wide Web capable of converting web document contents into a machine-readable format, and applied it within an Industry 4.0 setting. The integration of semantic-based technologies, including ontologies and knowledge graphs, have enabled the interconnection of data, while also incorporating semantic meaning into the model.

A part of this work, the development of the Steel Cold Rolling Ontology (SCRO) has been introduced, developed to capture and model the underlying concepts associated with cold rolling. The creation and adoption of SCRO brings many benefits and usages, providing a structured and standardised representation of knowledge in the domain of cold rolling

processes. Additionally, the novelty of SCRO led to further efforts in developing another ontology, designed to capture and represent the broader processes of steelmaking.

Furthermore, this thesis has explored the intersection of machine learning with semantic web technologies, emphasising the need and benefits of integrating these two AI methods. Our findings highlights the potential to utilise data-driven insights from ML, coupled with the knowledge representation and reasoning capabilities inherent in semantic methods. We demonstrated one contribution, which exemplified how this hybrid combination can enhance decision support for steel operators. Finally, the foundational blocks of applying a similar hybrid approach in the context of continuous reasoning on data streams has been illustrated.

7.2 Reflecting Over the Research Questions

In this section, we reflect over and recall the research question and contributions of the thesis.

- ***(RQ1)*** *What are the foundational standards and ongoing challenges in Industry 4.0 that relate to semantics?* In Chapter 3, a comprehensive survey was presented, addressing a gap of knowledge concerning semantic-based implementations for the Asset Administration Shell (AAS). The focus was specifically on adopting semantic methodologies to capture and model the communication and information layers of RAMI 4.0. The initial survey was presented at a conference in 2020 and has been updated to include relevant papers between 2021–2024.

  This survey identified three open challenges in the context of semantic-based AAS implementations. These include (1) the lack of standardised information models for semantic-based AAS, (2) inherent limitations in statistical data processing functionalities when compared to machine learning models, and (3) the lack of human behaviour considerations withing AAS implementations. The insights derived from this chapter shaped the direction of the thesis, guiding the focus on three main objects. Specifically, these are: (1) employing the application of semantic technologies in an Industry 4.0 setting to ensure consistent and uniform representation of processes and components in steelmaking, (2) the integration of machine learning models with semantic technologies to address the statistical data processing limitations identified in the survey paper, and (3) the development of a framework for incorporating humans factors, giving an example of how to to AI can be deployed to support with decision-making tasks for steelmaking operators on the shop floor.

- ***(RQ2)*** *How can the data be seamlessly integrated and understood through the use of semantics within smart manufacturing?* In Chapter 4, we presented two ontological frameworks to capture and model steel processes. The central contribution was the creation of the Steel Cold Rolling Ontology (SCRO), which further led to the development of the Core Reference Ontology for Steelmaking (CROS). Through these frameworks, we demonstrated how data can be integrated at a virtual and semantic level, transforming unconnected SQL databases into a cohesive knowledge graph

using SWRL and SPARQL. The presented use case illustrated how these methods can be deployed for data access, data integration, data querying, and condition-based maintenance purposes in the context of steel processes.

- ***(RQ3)*** *Is there a way to integrate ML with semantic methodologies in smart manufacturing? What are the existing AI methods available for combining these technologies?* Before exploring a hybrid approach, it was necessary to investigate statistical methods. The outcome of this investigation led to the creation of a random forest model deployed to predict the condition of work rolls during cold rolling operations, as detailed in Chapter 5. Once this model was established, we presented a framework that utilised semantic methodologies to apply rule-based reasoning. This approach replicated the RF classification process on knowledge graph data in RDF format, instead of the traditional CSV or SQL data used to train the RF model. This framework incorporated meta-data as part of the model, and was enriched with domain expert knowledge obtained through iterative knowledge acquisition session with domain experts from Tata Steel. The resulting hybrid method provided enhanced classification capabilities and provided valuable assistance for the operators on the shop floor.

  The initial motivation for this hybrid model was to use the domain expert knowledge to modify and prune the semantic-based random forest rule set before invoking the rule engine, with the ultimate goal of improving the accuracy of the RF model. However, several challenges including the "Knowledge acquisition bottleneck", where acquiring tacit knowledge of processes was inherently time consuming, and external factors such as reduced engagement with domain partners due to COVID, made obtaining a robust set of domain expert rules within the required time frame impossible. Despite these challenges, the study successfully demonstrated a method of combining expert rules with semantic rule-based random forest classification, resulting in enhanced classification capabilities for operators on the shop floor during cold rolling processes.

- ***(RQ4)*** *Can we apply continuous reasoning to data streams, incorporating both machine learning classification and domain expert knowledge on the fly?* In Chapter 6, we showcased the preliminary development of a framework that seamlessly integrated random forest classification with external domain expert knowledge, enabling enhanced, real-time classification continuously, without interrupting ongoing real-world operations. Future work includes replacing the dummy set of expert rules to produce more meaningful results.

7.2.1 Reflecting Over the RAMI 4.0 Model

The role of the RAMI 4.0 model has been critical in guiding the scope of the thesis. However, in my opinion, the sheer scale of the vision of Industry 4.0 made the RAMI 4.0 model overwhelming and difficult to comprehend.

The overall requirements were difficult to understand because they are segmented into different layers. For instance, taking the semantic requirements as an example, these requirements fall under three separate layers: (1) *Integration*, stating how industrial data can be defined and conceptualised, (2) *Communication*, defining the data exchange protocols between different systems and assets, and (3) *Information*, which states how the data should be structured and represented, ensuring that the information is adequately interpretable by the numerous systems.

I found this especially the case with the introduction of the asset administration shell as the standardised interface to describe every asset, which encompasses all the standards mentioned in each layer. Most literature available on the topic of AAS were theoretical and not concrete implementations. In our case too, it was unrealistic to implement an AAS as there are many other requirements to take into consideration which fall out of the scope of our research.

7.3 Future Directions

This ICASE research introduced a broad spectrum of concepts, technologies, and methodologies within the scope of the Semantic Web and Industry 4.0. The beauty of doing research lies in the many loose threads and undiscovered territories. In this thesis, we have intertwined concepts and technologies that are not conventionally interconnected.

The primary future direction is to continue the trajectory of the continuous stream reasoner in Chapter 6. While the foundational work of the framework has been laid, the domain expert rules are currently hypothetical—enabling the fundamental capabilities for the proof-of-concept—but lacking real-world applicable results. The future goal requires more interaction with domain experts to obtain a robust set of domain expert rules. Additionally, there is room to expand the knowledge representations in the ontologies introduced, or improve the random forest models through improved data filtering and aggregation techniques, or by exploring and comparing alternative ML methods within this specific setting.

Bibliography

[1] M. S. Acharya, A. Armaan, and A. S. Antony. A Comparison of Regression Models for Prediction of Graduate Admissions. In *2019 International Conference on Computational Intelligence in Data Science (ICCIDS)*, pages 1–5. IEEE, 2019.

[2] P. Adolphs, H. Bedenbender, D. Dirzus, M. Ehlich, U. Epple, M. Hankel, R. Heidel, M. Hoffmeister, H. Huhle, B. Karcher, H. Koziolek, R. Pichler, S. Pollmeier, F. Schewe, A. Walter, B. Waser, and M. Wollschlaeger. Reference Architecture Model Industrie 4.0 (RAMI4.0) ZVEI and VDI, Status Report. Technical report, ZVEI and VDI, 2015.

[3] E. C. Aifantis. The Physics of Plastic Deformation. *International Journal of Plasticity*, 3(3):211–247, 1987.

[4] M. Al-Mashari. Enterprise resource planning (ERP) systems: a research agenda. *Industrial Management & Data Systems*, 103(1):22–27, 2003.

[5] J. Ali, R. Khan, N. Ahmad, and I. Maqsood. Random Forests and Decision Trees. *International Journal of Computer Science Issues (IJCSI)*, 9(5):272, 2012.

[6] E. Alpaydin. *Introduction to Machine Learning*. MIT Press, 2020.

[7] N. Altman and M. Krzywinski. Ensemble Methods: Bagging and Random Forests. *Nature Methods*, 14(10):933–935, 2017.

[8] J. Alzubi, A. Nayyar, and A. Kumar. Machine Learning from Theory to Algorithms: An Overview. In *Journal of Physics: Conference Series*, volume 1142, page 012012. IOP Publishing, nov 2018.

[9] F. Ameri, C. Urbanovsky, and C. Mcarthur. A Systematic Approach to Developing Ontologies for Manufacturing Service Modeling. *CEUR Workshop Proceedings*, 886, 01 2012.

[10] N. Ammar, A. Shaban-Nejad, et al. Explainable Artificial Intelligence Recommendation System by Leveraging the Semantics of Adverse Childhood Experiences: Proof-of-concept Prototype Development. *JMIR Medical Informatics*, 8(11):e18752, 2020.

[11] D. Anicic, P. Fodor, S. Rudolph, and N. Stojanovic. EP-SPARQL: A Unified Language for Event Processing and Stream Reasoning. In *Proceedings of the 20th International Conference on World Wide Web*, pages 635–644, 2011.

[12] G. Antoniou and F. v. Harmelen. *Web Ontology Language: OWL*, pages 91–110. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[13] J. Aylen. Megabytes for Metals: Development of Computer Applications in the Iron and Steel Industry. *Ironmaking & Steelmaking*, 31(6):465–478, 2004.

[14] S. R. Bader and M. Maleshkova. The semantic asset administration shell. In M. Acosta, P. Cudré-Mauroux, M. Maleshkova, T. Pellegrini, H. Sack, and Y. Sure-Vetter, editors, *Semantic Systems. The Power of AI and Knowledge Graphs*, pages 159–174, Cham, 2019. Springer International Publishing.

[15] S. Badillo, B. Banfai, F. Birzele, I. I. Davydov, L. Hutchinson, T. Kam-Thong, J. Siebourg-Polster, B. Steiert, and J. D. Zhang. An Introduction to Machine Learning. *Clinical Pharmacology & Therapeutics*, 107(4):871–885, 2020.

[16] T. Bagosi, D. Calvanese, J. Hardi, S. Komla-Ebri, D. Lanti, M. Rezk, M. Rodríguez-Muro, M. Slusnys, and G. Xiao. The Ontop Framework for Ontology Based Data Access. In D. Zhao, J. Du, H. Wang, P. Wang, D. Ji, and J. Z. Pan, editors, *The Semantic Web and Web Science*, pages 67–77, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[17] J. Bakakeu, M. Brossog, J. Zeitler, J. Franke, S. Tolksdorf, H. Klos, and J. Peschke. Automated Reasoning and Knowledge Inference on OPC UA information Models. In *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, pages 53–60. IEEE, 2019.

[18] Q. Bao, J. Wang, and J. Cheng. Research on Ontology Modeling of Steel Manufacturing Process Based on Big Data Analysis. In *MATEC Web of Conferences*, volume 45, page 04005. EDP Sciences, 2016.

[19] D. Barbieri, D. Braga, S. Ceri, E. Della Valle, and M. Grossniklaus. Stream Reasoning: Where We Got So Far. In *Proceedings of the 4th Workshop on New Forms of Reasoning for the Semantic Web: Scalable & Dynamic*, 01 2010.

[20] D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle, and M. Grossniklaus. C-SPARQL: SPARQL for Continuous Querying. In *Proceedings of the 18th International Conference on World Wide Web*, pages 1061–1062, 2009.

[21] D. F. Barbieri, D. Braga, S. Ceri, E. D. Valle, and M. Grossniklaus. Querying RDF Streams With C-SPARQL. *ACM SIGMOD Record*, 39(1):20–26, 2010.

[22] P. Barbrook-Johnson and A. S. Penn. Bayesian Belief Networks. In *Systems Mapping: How to Build and Use Causal Models of Systems*, pages 97–112. Springer, 2022.

[23] J. Barwise. An Introduction to First-order Logic. In *Studies in Logic and the Foundations of Mathematics*, volume 90, pages 5–46. Elsevier, 1977.

[24] H. Beck, M. Dao-Tran, and T. Eiter. LARS: A Logic-based Framework for Analytic Reasoning Over Streams. *Artificial Intelligence*, 261:16–70, 2018.

[25] K. Beck and E. Gamma. *Extreme Programming Explained: Embrace Change*. An Alan R. Apt Book Series. Addison-Wesley, 2000.

[26] S. Beden and A. Beckmann. Towards an Ontological Framework for Integrating Domain Expert Knowledge with RF Classification. In *IEEE 17th International Conference on Semantic Computing*, 2023.

[27] S. Beden, Q. Cao, and A. Beckmann. Semantic Asset Administration Shells in Industry 4.0: A Survey. In *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*, pages 31–38, 2021.

[28] J. G. Bekker, I. K. Craig, and P. C. Pistorius. Modeling and Simulation of an Electric Arc Furnace Process. *ISIJ international*, 39(1):23–32, 1999.

[29] P. Belford. Hot Blast Iron Smelting in the Early 19th Century: a Re-appraisal. *Historical Metallurgy*, 46(1):32–44, 2012.

[30] M. Belgiu and L. Drăguţ. Random forest in Remote Sensing: A Review of Applications and Future Directions. *ISPRS Journal of Photogrammetry and Remote Sensing*, 114:24–31, 2016.

[31] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, and A. Secret. The World-Wide Web. *Communications of the ACM*, 37(8):76–82, 1994.

[32] T. Berners-Lee, R. Fielding, and L. Masinter. Rfc2396: Uniform resource identifiers (uri): Generic syntax, 1998.

[33] T. Berners-Lee, J. Hendler, and O. Lassila. A New Form of Web Content that is Meaningful to Computers will Unleash a Revolution of New Possibilities. *Scientific American*, 284(5):34–43, 2001.

[34] H. Bessemer. On the Manufacture of Malleable Iron and Steel. In *Minutes of the Proceedings of the Institution of Civil Engineers*, volume 18, pages 525–546. Thomas Telford-ICE Virtual Library, 1859.

[35] H. Bessemer. *Sir Henry Bessemer, FRS: An Autobiography, with a Concluding Chapter*, volume 451. Woodhead Publishing Limited, 1905.

[36] C. Bettini, G. Civitarese, D. Giancane, and R. Presotto. Procaviar: Hybrid Data-driven and Probabilistic Knowledge-based Activity Recognition. *IEEE Access*, 8:146876–146886, 2020.

[37] J.-P. Birat. Alternative Ways of Making Steel: Retrospective and Prospective. *Metallurgical Research & Technology*, 101(11):937–955, 2004.

[38] J. Block. Assimilation, Accommodation, and the Dynamics of Personality Development. *Child Development*, pages 281–295, 1982.

[39] E. Blomqvist, V. Presutti, E. Daga, and A. Gangemi. Experimenting with eXtreme Design. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 120–134. Springer, 2010.

[40] E. Blomqvist, V. Presutti, E. Daga, and A. Gangemi. Experimenting with eXtreme Design. In P. Cimiano and H. S. Pinto, editors, *Knowledge Engineering and Management by the Masses*, pages 120–134, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[41] H. Boley, S. Tabet, and G. Wagner. Design Rationale for RuleML: A Markup Language for Semantic Web Rules. In *SWWS*, volume 1, pages 381–401, 2001.

[42] P. Bonte, F. De Turck, and F. Ongenae. Towards an Evaluation Framework for Expressive Stream Reasoning. In *The Semantic Web: ESWC 2021 Satellite Events: Virtual Event, June 6–10, 2021, Revised Selected Papers 18*, pages 76–81. Springer, 2021.

[43] P. Bonte, F. D. Turck, and F. Ongenae. Bridging the Gap Between Expressivity and Efficiency in Stream Reasoning: a Structural Caching Approach for IoT Streams. *Knowledge and Information Systems*, 64(7):1781–1815, 2022.

[44] S. Borgo and P. Leitão. Foundations for a Core Ontology of Manufacturing. In *Ontologies*, pages 751–775. Springer, 2007.

[45] D. Boswell. Introduction to Support Vector Machines. *Departement of Computer Science and Engineering University of California San Diego*, 11, 2002.

[46] J. Brank. Gold Standard Based Ontology Evaluation using Instance Assignment. In *Inproceedings Of The EON 2006 Workshop*, 2006.

[47] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[48] A. Breit, L. Waltersdorfer, F. J. Ekaputra, M. Sabou, A. Ekelhart, A. Iana, H. Paulheim, J. Portisch, A. Revenko, A. t. Teije, et al. Combining Machine Learning and Semantic Web: A Systematic Mapping Study. *ACM Computing Surveys*, 2023.

[49] British Standards Institution (BSI). *Smart Manufacturing — Reference Architecture Model Industry 4.0 (RAMI4.0).* BSI Standards Limited 2017, 2017.

[50] British Standards Institution (BSI). *Smart manufacturing — Reference architecture model industry 4.0 (RAMI4.0).* BSI Standards Limited 2017, 2017.

[51] B. G. Buchanan and R. G. Smith. Fundamentals of Expert Systems. *Annual Review of Computer Science*, 3(1):23–58, 1988.

[52] M. Buron, F. Goasdoué, I. Manolescu, and M.-L. Mugnier. Ontology-based RDF Integration of Heterogeneous Data. In *EDBT 2020-23rd International Conference on Extending Database Technology*, pages 299–310, 2020.

[53] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, and G. Xiao. Ontop: Answering SPARQL Queries Over Relational Databases. *Semantic Web*, 8, 02 2016.

[54] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi, and D. F. Savo. The MASTRO System for Ontology-based Data Access. *Semantic Web*, 2:43–53, 2011.

[55] Q. Cao, S. Beden, and A. Beckmann. A Core Reference Ontology for Steelmaking Process Knowledge Modelling and Information Management. *Computers in Industry*, 135, 2022.

[56] Q. Cao, F. Giustozzi, C. Zanni-Merk, F. de Bertrand de Beuvron, and C. Reich. Smart Condition Monitoring for Industry 4.0 Manufacturing Processes: An ontology-based Approach. *Cybernetics and Systems*, 50(2):82–96, 2019.

[57] Q. Cao, A. Samet, C. Zanni-Merk, F. de Bertrand de Beuvron, and C. Reich. Combining Chronicle Mining and Semantics for Predictive Maintenance in Manufacturing Processes. *Semantic Web*, (Preprint):1–22, 2020.

[58] Q. Cao, A. Samet, C. Zanni-Merk, F. d. B. de Beuvron, and C. Reich. An Ontology-based Approach for Failure Classification in Predictive Maintenance using Fuzzy C-means and SWRL rules. *Procedia Computer Science*, 159:630–639, 2019.

[59] Q. Cao, C. Zanni-Merk, and C. Reich. Ontologies for Manufacturing Process Modeling: A Survey. In *International Conference on Sustainable Design and Manufacturing*, pages 61–70. Springer, 2018.

[60] Q. Cao, C. Zanni-Merk, A. Samet, F. d. B. de Beuvron, and C. Reich. Using Rule Quality Measures for Rule Base Refinement in Knowledge-Based Predictive Maintenance Systems. *Cybernetics and Systems*, 51(2):161–176, 2020.

[61] Q. Cao, C. Zanni-Merk, A. Samet, C. Reich, F. Beuvron, A. Beckmann, and C. Giannetti. KSPMI: A Knowledge-based System for Predictive Maintenance in Industry 4.0. *Robotics and Computer-Integrated Manufacturing*, 74, 2022.

[62] S. Cavalieri, S. Mulé, and M. G. Salafia. OPC UA-based Asset Administration Shell. In *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, volume 1, pages 2982–2989, 2019.

[63] A. D. Chandler. Anthracite Coal and the Beginnings of the Industrial Revolution in the United States. *Business History Review*, 46(2):141–181, 1972.

[64] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. R. Madden, F. Reiss, and M. A. Shah. TelegraphCQ: Continuous Dataflow Processing. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 668–668, 2003.

[65] B. Chen, J. Wan, L. Shu, P. Li, M. Mukherjee, and B. Yin. Smart factory of Industry 4.0: Key Technologies, Application Case, and Challenges. *IEEE Access*, 6:6505–6519, 2017.

[66] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang. A Vision of IoT: Applications, Challenges, and Opportunities with china Perspective. *IEEE Internet of Things Journal*, 1(4):349–359, 2014.

[67] G. O. Consortium. The Gene Ontology (GO) Database and Informatics Resource. *Nucleic Acids Research*, 32(suppl_1):D258–D261, 2004.

[68] G. O. Consortium. The Gene Ontology Resource: 20 Years and Still GOing Strong. *Nucleic Acids Research*, 47(D1):D330–D338, 2019.

[69] O. Corcho, M. Fernández-López, A. Gómez-Pérez, and A. López-Cima. Building Legal Ontologies with METHONTOLOGY and WebODE. *Law and the Semantic Web: Legal Ontologies, Methodologies, Legal Information Retrieval, and Applications*, pages 142–157, 2005.

[70] C. Cortes and V. Vapnik. Support-vector Networks. *Machine Learning*, 20:273–297, 1995.

[71] G. Culot, G. Nassimbeni, G. Orzes, and M. Sartor. Behind the Definition of Industry 4.0: Analysis and Open Questions. *International Journal of Production Economics*, 226:107617, 2020.

[72] P. Cunningham and S. J. Delany. k-Nearest Neighbour Classifiers: A Tutorial. *ACM Computing Surveys (CSUR)*, 54(6):1–25, 2021.

[73] R. Cyganiak, D. Wood, M. Lanthaler, G. Klyne, J. J. Carroll, and B. McBride. RDF 1.1 Concepts and Abstract Syntax. *W3C Recommendation*, 25(02):1–22, 2014.

[74] J. Dalzochio, R. Kunst, E. Pignaton, A. Binotto, S. Sanyal, J. Favilla, and J. Barbosa. Machine Learning and Reasoning for Predictive Maintenance in Industry 4.0: Current Status and Challenges. *Computers in Industry*, 123, 2020.

[75] A. Daneels and W. Salter. What is SCADA? *International Conference on Accelerator and Large Experimental Physics Control Systems*, 1999.

[76] R. Daneshjou, M. Yuksekgonul, Z. R. Cai, R. A. Novoa, and J. Zou. SkinCon: A Skin Disease Dataset Densely Annotated by Domain Experts for Fine-grained Debugging and Analysis. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[77] J. Day and R. Tylecote. *The Industrial Revolution in Metals*. Book (Institute of Metals). Institute of Metals, 1991.

[78] S. Decker, S. Melnik, F. Harmelen, D. Fensel, M. Klein, M. Erdmann, and I. Horrocks. The Semantic Web: The Roles of XML and RDF. *IEEE Internet Computing*, 4, 10 2000.

[79] S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, and I. Horrocks. The Semantic Web: the Roles of XML and RDF. *IEEE Internet Computing*, 4(5):63–73, 2000.

[80] E. Della Valle, S. Ceri, D. F. Barbieri, D. Braga, and A. Campi. A First Step Towards Stream Reasoning. In *Future Internet–FIS 2008: First Future Internet Symposium*, pages 72–81. Springer, 2009.

[81] D. Dell'Aglio, E. Della Valle, F. van Harmelen, and A. Bernstein. Stream Reasoning: A Survey and Outlook. *Data Science*, 1(1-2):59–83, 2017.

[82] H. M. W. Delport. *The Development of a DRI Process for Small Scale EAF-based Steel Mills*. PhD thesis, Stellenbosch: Stellenbosch University, 2010.

[83] D. Dering, C. Swartz, and N. Dogan. Dynamic Modeling and Simulation of Basic Oxygen Furnace (BOF) Operation. *Processes*, 8(4):483, 2020.

[84] F. P. Dewey. The History of Electric Heating Applied to Metallurgy. *Journal of the American Chemical Society*, 18(3):287–307, 1896.

[85] S. A. Dick. Coded Conduct: Making MACSYMA Users and the Automation of Mathematics. *BJHS Themes*, 5:205–224, 2020.

[86] M. Dobrev, D. Gocheva, and I. Batchkova. An Ontological Approach for Planning and Scheduling in Primary Steel Production. In *2008 4th International IEEE Conference Intelligent Systems*, volume 1, pages 6–14. IEEE, 2008.

[87] R. Drath, M. Rentschler, and M. Hoffmeister. The AutomationML Component Description in the context of the Asset Administration Shell. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1278–1281, 2019.

[88] M. Dürst and M. Suignard. Internationalized Resource Identifiers (IRIs). Technical report, W3C Consortium, 2005. Available at: https://www.w3.org/2001/Talks/0912-IUC-IRI/paper.html.

[89] C. d'Amato. Machine Learning for the Semantic Web: Lessons Learnt and Next Research Directions. *Semantic Web*, 11(1):195–203, 2020.

[90] A. Eckhardt, S. Müller, and L. Leurs. An Evaluation of the Applicability of OPC UA Publish Subscribe on Factory Automation use Cases. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1071–1074, 2018.

[91] B. Efron. *The Jackknife, the Bootstrap and Other Resampling Plans*. SIAM, 1982.

[92] B. Esmaeilian, S. Behdad, and B. Wang. The Evolution and Future of Manufacturing: A Review. *Journal of Manufacturing Systems*, 39:79–100, 2016.

[93] R. Falco, A. Gangemi, S. Peroni, D. Shotton, and F. Vitali. Modelling OWL Ontologies with Graffoo. In *The Semantic Web: ESWC 2014 Satellite Events: ESWC 2014 Satellite Events, Anissaras, Crete, Greece, May 25-29, 2014, Revised Selected Papers 11*, pages 320–325. Springer, 2014.

[94] M. D. Fenton. *Mineral Commodity Profiles, Iron and Steel*. US Geological Survey Reston, VA, 2005.

[95] E. Filos. Four years of 'Factories of the Future'in Europe: Achievements and Outlook. *International Journal of Computer Integrated Manufacturing*, 30(1):15–22, 2017.

[96] C. L. Forgy. Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. In *Readings in Artificial Intelligence and Databases*, pages 547–559. Elsevier, 1989.

[97] G. Frege. Begriffsschrift, a Formula Language, Modeled upon that of Arithmetic, for Pure Thought. *From Frege to Gödel: A Source Book in Mathematical Logic*, 1931:1–82, 1879.

[98] R. Fruehan. Overview of Steelmaking Processes and Their Development. *The Making, Shaping and Treating of Steel: Steelmaking and Refining*, pages 2–3, 1998.

[99] J. Frysak, C. Kaar, and C. Stary. Benefits and Pitfalls Applying RAMI4.0. In *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, pages 32–37, 05 2018.

[100] J. Fuchs, J. Schmidt, J. Franke, K. Rehman, M. Sauer, and S. Karnouskos. I4. 0-compliant Integration of Assets Utilizing the Asset Administration Shell. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1243–1247. IEEE, 2019.

[101] J. Fuchs, J. Schmidt, J. Franke, K. Rehman, M. Sauer, and S. Karnouskos. I4.0-compliant Integration of Assets Utilizing the Asset Administration Shell. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1243–1247, 2019.

[102] J. Fuchs, J. Schmidt, J. Franke, K. Rehman, M. Sauer, and S. Karnouskos. I4.0-compliant Integration of Assets Utilizing the Asset Administration Shell. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1243–1247, 2019.

[103] W. Gale. The Bessemer Steelmaking Process. *Transactions of the Newcomen Society*, 46(1):17–26, 1973.

[104] A. Gangemi. Ontology Design Patterns for Semantic Web Content. In *International Semantic Web Conference*, pages 262–276. Springer, 2005.

[105] A. Gangemi and V. Presutti. *Ontology Design Patterns*, pages 221–243. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[106] A. d. Garcez and L. C. Lamb. Neurosymbolic AI: The 3rd Wave. *Artificial Intelligence Review*, pages 1–20, 2023.

[107] M. D. Garvey, S. Carnovale, and S. Yeniyurt. An Analytical Framework for Supply Network Risk Propagation: A Bayesian Network Approach. *European Journal of Operational Research*, 243(2):618–627, 2015.

[108] M. R. Genesereth. The Difficulties of Using MACSYMA and the Function of User Aids. In *Proc. 1977 MACSYMA Users' Conference*, pages 291–307, 1977.

[109] C. Geng, S. Li, C. Yue, and Y. Ma. Pyrolysis Characteristics of Bituminous Coal. *Journal of the Energy Institute*, 89(4):725–730, 2016.

[110] D. Geng, Y. Wen, and Q. Gao. Research on Semantic Extension Method of Real-time Data for OPC UA. In *3rd International Conference on Materials Science, Machinery and Energy Engineering (MSMEE 2018)*, 2018.

[111] German Electrical and Electronic Manufacturers' Association. Examples of the Asset Administration Shell for Industrie 4.0 Components – Basic Part. Technical report, German Electrical and Electronic Manufacturers' Association, 2017.

[112] M. Ghobakhloo. Industry 4.0, Digitization, and Opportunities for Sustainability. *Journal of Cleaner Production*, 252:119869, 2020.

[113] A. Ghosh and A. Chatterjee. *Iron Making and Steelmaking: Theory and Practice*. PHI Learning Pvt. Ltd., 2008.

[114] A. Gómez-Pérez. Ontology Evaluation. In *Handbook on Ontologies*, pages 251–273. Springer, 2004.

[115] A. Gómez-Pérez and O. Corcho. Ontology Languages for the Semantic Web. *IEEE Intelligent Systems*, 17(1):54–60, 2002.

[116] I. Grangel-González, P. Baptista, L. Halilaj, S. Lohmann, M.-E. Vidal, C. Mader, and S. Auer. The Industry 4.0 Standards Landscape from a Semantic Integration Perspective. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 09 2017.

[117] I. Grangel-González, L. Halilaj, S. Auer, S. Lohmann, C. Lange, and D. Collarana. 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)*, pages 1–8. IEEE, 2016.

[118] I. Grangel-González, L. Halilaj, G. Coskun, S. Auer, D. Collarana, and M. Hoffmeister. Towards a Semantic Administrative Shell for Industry 4.0 Components. In *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)*, pages 230–237. IEEE, 2016.

[119] I. Grangel-González, L. Halilaj, G. Coskun, S. Auer, D. Collarana, and M. Hofmeister. Towards a Semantic Administrative Shell for Industry 4.0 Components. In *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)*, pages 230–237, 02 2016.

[120] C. Grosan, A. Abraham, C. Grosan, and A. Abraham. Rule-based Expert Systems. *Intelligent Systems: A Modern Approach*, pages 149–185, 2011.

[121] B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description Logic Programs: Combining Logic Programs with Description Logic. In *Proceedings of the 12th International Conference on World Wide Web*, pages 48–57, 2003.

[122] T. Gruber. *Ontology*, pages 1963–1965. Springer US, Boston, MA, 2009.

[123] T. R. Gruber. Ontolingua: A Mechanism to Support Portable Ontologies, 1992.

[124] T. R. Gruber. Toward Principles for the Design of Ontologies used for Knowledge Sharing? *International Journal of Human-Computer Studies*, 43(5):907–928, 1995.

[125] S. Grüner, J. Pfrommer, and F. Palm. RESTful Industrial Communication With OPC UA. *IEEE Transactions on Industrial Informatics*, 12(5):1832–1841, 2016.

[126] M. Grüninger. Using the PSL Ontology. In *Handbook on Ontologies*, pages 423–443. Springer, 2009.

[127] M. Gruninger and M. S. Fox. The Design and Evaluation of Ontologies for Enterprise Engineering. In *Workshop on Implemented Ontologies, European Conference on Artificial Intelligence (ECAI)*, 1994.

[128] N. Guarino, D. Oberle, and S. Staab. What is an Ontology? *Handbook on Ontologies*, pages 1–17, 2009.

[129] K. Hamilton, A. Nayak, B. Božić, and L. Longo. Is neuro-symbolic ai meeting its promises in natural language processing? a structured review. *Semantic Web*, (Preprint):1–42, 2022.

[130] M. Hankel and B. Rexroth. The Reference Architectural Model Industrie 4.0 (RAMI 4.0). *ZVEI*, 2(2):4–9, 2015.

[131] T. Hannelius, M. Salmenpera, and S. Kuikka. Roadmap to Adopting OPC UA. In *2008 6th IEEE International Conference on Industrial Informatics*, pages 756–761. IEEE, 2008.

[132] D. Harel, D. Kozen, and R. Parikh. Process Logic: Expressiveness, Decidability, Completeness. *Journal of Computer and System Sciences*, 25(2):144–170, 1982.

[133] H. Haskamp, M. Meyer, R. Möllmann, F. Orth, and A. W. Colombo. Benchmarking of Existing OPC UA Implementations for Industrie 4.0-compliant Digitalization Solutions. In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, pages 589–594, 2017.

[134] H. Haskamp, M. Meyer, R. Möllmann, F. Orth, and A. W. Colombo. Benchmarking of existing OPC UA implementations for Industrie 4.0-compliant digitalization solutions. In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, pages 589–594, 2017.

[135] R. Heidel. *Industrie 4.0: The Reference Architecture Model RAMI 4.0 and the Industrie 4.0 Component*. Beuth Verlag GmbH, 2019.

[136] G. Henderson. Steel Company of Wales/British Steel Corporation Production System Computer Projects at Abbey Works, Port Talbot 1965–1973: A Brief History, August 2019. Available at: https://archivesit.org.uk/wp-content/uploads/2019/08/Steel-Company-of-Wales-British-Steel-Corporation-production-system-computer-projects-1965-1973.pdf.

[137] E. F. Hill. *Jess in Action: Java Rule-based Systems*. Manning Publications Co., 2003.

[138] P. Hitzler and M. Sarker. *Neuro-Symbolic Artificial Intelligence: The State of the Art*. Frontiers in Artificial Intelligence and Applications. IOS Press, 2022.

[139] J. R. Hobbs and F. Pan. Time Ontology in OWL. *W3C working draft*, 27:133, 2006.

[140] L. Holappa. Historical Overview on the Development of Converter Steelmaking From Bessemer to Modern Practices and Future Outlook. *Mineral Processing and Extractive Metallurgy*, 128(1-2):3–16, 2019.

[141] A. Holzinger, A. Saranti, C. Molnar, P. Biecek, and W. Samek. Explainable AI Methods: A Brief Overview. In *International Workshop on Extending Explainable AI Beyond Deep Models and Classifiers*, pages 13–38. Springer, 2020.

[142] M. Horridge, H. Knublauch, A. Rector, R. Stevens, and C. Wroe. A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools, 08 2004.

[143] I. Horrocks and P. F. Patel-Schneider. Three Theses of Representation in the Semantic Web. In *Proceedings of the 12th International Conference on World Wide Web*, pages 39–47, 2003.

[144] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, et al. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. *W3C Member Submission*, 21(79):1–31, 2004.

[145] D. Horvath and R. Szabo. Driving Forces and Barriers of Industry 4.0: Do Multinational and Small and Medium-sized Companies have Equal Opportunities? *Technological Forecasting and Social Change*, 146:119–132, 06 2019.

[146] Y. Hua and B. Hein. Concept Learning in AutomationML with Formal Semantics and Inductive Logic Programming. In *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pages 1542–1547, 2018.

[147] Y. Huang, S. Dhouib, L. P. Medinacelli, and J. Malenfant. Semantic Interoperability of Digital Twins: Ontology-based Capability Checking in AAS Modeling Framework. In *2023 IEEE 6th International Conference on Industrial Cyber-Physical Systems (ICPS)*, pages 1–8. IEEE, 2023.

[148] IEC PAS 63088. Smart Manufacturing–Reference Architecture Model Industry 4.0 (RAMI4.0), 2017.

[149] I. F. Ilyas, T. Rekatsinas, V. Konda, J. Pound, X. Qi, and M. Soliman. SAGA: A Platform for Continuous Construction and Serving of Knowledge at Scale. In *Proceedings of the 2022 International Conference on Management of Data*, pages 2259–2272, 2022.

[150] Industrial Internet Consortium. The Industrial Internet of Things Volume G1: Reference Architecture. Technical report, Industrial Internet Consortium, 6 2019.

[151] Industry IoT Consortium and Others. The Industrial Internet Reference Architecture. Technical report, Industry IoT Consortium and Others, 2022.

[152] International Electrotechnical Commission. IEC 61512-4:2009, Batch control. Available at: `https://webstore.iec.ch/publication/5531`, 2009. International Standard.

[153] International Electrotechnical Commission. IEC 62264-1:2013, Enterprise-control System inIegration. Available at: `https://www.iso.org/standard/57308.html`, 2013. International Standard.

[154] International Electrotechnical Commission. IEC 61360-1:2017, Standard Data Element Types with Associated Classification Scheme. Available at: `https://webstore.iec.ch/publication/28560`, 2017. International Standard.

[155] International Electrotechnical Commission. IEC 62890:2020, Industrial-process Measurement, Control and Automation - Life-cycle-management for Systems and Components. Available at: `https://webstore.iec.ch/publication/30583`, 2020. International Standard.

[156] International Electrotechnical Commission and others. Engineering Data Exchange Format for Use in Industrial Automation Systems Engineering–Automation Markup Language–Part 1: Architecture and General Requirements. *International Electrotechnical Commission: Durham, NC, USA*, 2018.

[157] M. Jabardi and A. Hadi. Twitter Fake Account Detection and Classification Using Ontological Engineering and Semantic Web Rule Language. *Karbala International Journal of Modern Science*, 2020.

[158] M. H. Jabardi and A. S. Hadi. Using Machine Learning to Inductively Learn Semantic Rules. *Journal of Physics: Conference Series*, 1804(1):012099, feb 2021.

[159] P. Jackson. *Introduction to Expert Systems*. Addison-Wesley Pub. Co., Reading, MA, 1986.

[160] M. Jacoby, B. Jovicic, L. Stojanovic, and N. Stojanović. An Approach for Realizing Hybrid Digital Twins using Asset Administration Shells and Apache StreamPipes. *Information*, 12(6):217, 2021.

[161] E. Jajaga and L. Ahmedi. C-SWRL: SWRL for Reasoning Over Stream Data. In *2017 IEEE 11th International Conference on Semantic Computing (ICSC)*, pages 395–400. IEEE, 2017.

[162] E. Jajaga, L. Ahmedi, and F. Ahmedi. StreamJess: A Stream Reasoning Framework for Water Quality Monitoring. *International Journal of Metadata, Semantics and Ontologies*, 11(4):207–220, 2016.

[163] A. J. Jara, A. C. Olivieri, Y. Bocchi, M. Jung, W. Kastner, and A. F. Skarmeta. Semantic Web of Things: An Analysis of the Application Semantics for the IoT Moving Towards the IoT Convergence. *International Journal of Web and Grid Services*, 10(2-3):244–272, 2014.

[164] A. Jena. Apache jena: A Free and Open Source Java Framework for Building Semantic Web and Linked Data Applications. *accessed June*, 2, 2020.

[165] E. Jiménez-Ruiz, E. Kharlamov, D. Zheleznyakov, I. Horrocks, C. Pinkel, M. G. Skjæveland, E. Thorstensen, and J. Mora. BootOX: Practical Mapping of RDBs to

OWL 2. In M. Arenas, O. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. Groth, M. Dumontier, J. Heflin, K. Thirunarayan, and S. Staab, editors, *The Semantic Web - ISWC 2015*, pages 113–132, Cham, 2015. Springer International Publishing.

[166] I. Johnson, J. Abécassis, B. Charnomordic, S. Destercke, and R. Thomopoulos. Making Ontology-based Knowledge and Decision Trees Interact: An Approach to Enrich Knowledge and Increase Expert Confidence in Data-driven Models. *Lecture Notes in Computer Science*, 2010.

[167] T. L. Johnson, S. R. Fletcher, W. Baker, and R. L. Charles. How and Why We Need to Capture Tacit Knowledge in Manufacturing: Case Studies of Visual Inspection. *Applied Ergonomics*, 74:1–9, 2019.

[168] D. M. Jones, T. J. M. Bench-Capon, and P. R. S. Visser. Methodologies for ontology development. Available at: `https://cgi.csc.liv.ac.uk/~tbc/publications/itknows.pdf`, 1998.

[169] A. Kalyanpur, B. Parsia, E. Sirin, B. C. Grau, and J. Hendler. Swoop: A Web Ontology Editing Browser. *Journal of Web Semantics*, 4(2):144–153, 2006.

[170] A. Kataria and M. Singh. A Review of Data Classification Using K-nearest Neighbour Algorithm. *International Journal of Emerging Technology and Advanced Engineering*, 3(6):354–360, 2013.

[171] B. Katti, C. Plociennik, M. Ruskowski, and M. Schweitzer. SA-OPC-UA: Introducing Semantics to OPC-UA Application Methods. In *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pages 1189–1196, 2018.

[172] B. Katti, C. Plociennik, and M. Schweitzer. SemOPC-UA: Introducing Semantics to OPC-UA Application Specific Methods. *IFAC-PapersOnLine*, 51(11):1230–1236, 2018. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018.

[173] M. R. Khondoker and P. Mueller. Comparing Ontology Development Tools based on an Online Survey. In *Proceedings of the World Congress on Engineering (WCE 2010), London, UK*, volume 1, 2010.

[174] A. Kidd. *Knowledge Acquisition for Expert Systems: A Practical Handbook*. Springer Science & Business Media, 2012.

[175] H. Klaus, M. Rosemann, and G. G. Gable. What is ERP? *Information Systems Frontiers*, 2:141–162, 2000.

[176] S. Kleene. Representation of Events in Nerve Nets and Finite Automata. *Automata Studies: Annals of Mathematics Studies*, (34):3, 1956.

[177] J. Kletti. *Manufacturing Execution Systems—MES*. Springer, 2007.

[178] K. I. Kotis, K. Zachila, and E. Paparidis. Machine Learning Meets the Semantic Web. *Artificial Intelligence Advances*, 3(1):63–70, 2021.

[179] M. Kovačič, K. Stopar, R. Vertnik, and B. Šarler. Comprehensive Electric Arc Furnace Electric Energy Consumption Modeling: A Pilot Study. *Energies*, 12(11):2142, 2019.

[180] A. Krenker, J. Bešter, and A. Kos. Introduction to the Artificial Neural Networks. *Artificial Neural Networks: Methodological Advances and Biomedical Applications*, pages 1–18, 2011.

[181] S. Krishnamurthy, S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Madden, F. Reiss, and M. A. Shah. TelegraphCQ: An Architectural Status Report. *IEEE Data Eng. Bull.*, 26(1):11–18, 2003.

[182] A. Kumar, P. Kaur, and P. Sharma. A survey on Hoeffding tree stream data classification algorithms. *CPUH-Research Journal*, 1(2):28–32, 2015.

[183] V. P. Kumar and I. Sowmya. A Review on Pros and Cons of Machine Learning Algorithms. *Journal of Engineering Sciences*, 12(10):272–276, 2021.

[184] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann. Industry 4.0. *Business & Information Systems Engineering*, 6:239–242, 2014.

[185] S. Lemaignan, A. Siadat, J.-Y. Dantan, and A. Semenenko. MASON: A Proposal for an Ontology of Manufacturing Domain. In *IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)*, pages 195–200. IEEE, 2006.

[186] H. J. Levesque. Knowledge Representation and Reasoning. *Annual Review of Computer Science*, 1(1):255–287, 1986.

[187] S. Liang and P. Fodor. OpenRuleBench: An Analysis of the Performance of Rule Engines. In *Proceedings of the 18th International Conference on World Wide Web*, 2009.

[188] S. Liang, P. Fodor, H. Wan, and M. Kifer. OpenRuleBench: An Analysis of the Performance of Rule Engines. In *Proceedings of the 18th International Conference on World Wide Web*, pages 601–610, 2009.

[189] S.-W. Lin, B. Miller, J. Durand, R. Joshi, P. Didier, A. Chigani, R. Torenbeek, D. Duggal, R. Martin, G. Bleakley, et al. Industrial Internet Reference Architecture. *Industrial Internet Consortium (IIC), Tech. Rep*, 2015.

[190] P. Lord. Components of an Ontology. *Ontogenesis*, 2010.

[191] A. Lüder, M. Schleipen, N. Schmidt, J. Pfrommer, and R. Henßen. One Step Towards an Industry 4.0 Component. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pages 1268–1273, 2017.

[192] A. I. Maarala, X. Su, and J. Riekki. Semantic Reasoning for Context-aware Internet of Things Applications. *IEEE Internet of Things Journal*, pages 461–473, 2016.

[193] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, R. Metz, and B. A. Hamilton. Reference Model for Service Oriented Architecture 1.0. *OASIS Standard*, 12(S18):1–31, 2006.

[194] A. Maedche and S. Staab. Ontology Learning for the Semantic Web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.

[195] P. Marcon, C. Diedrich, F. Zezulka, T. Schröder, A. Belyaev, J. Arm, T. Benesl, Z. Bradac, and I. Vesely. The Asset Administration Shell of Operator in the Platform of Industry 4.0. In *2018 18th International Conference on Mechatronics - Mechatronika (ME)*, pages 1–5, 2018.

[196] G. Marcus. Deep Learning: A Critical Appraisal. *arXiv Preprint arXiv:1801.00631*, 2018.

[197] G. Marcus. The Next Decade in AI: Four Steps Towards Eobust Artificial Intelligence. *arXiv Preprint arXiv:2002.06177*, 2020.

[198] W. A. Martin and R. J. Fateman. The MACSYMA System. In *Proceedings of the second ACM symposium on Symbolic and algebraic manipulation*, pages 59–75, 1971.

[199] S. G. Mathias, S. Schmied, and D. Grossmann. An Investigation on Database Connections in OPC UA Applications. *Procedia Computer Science*, 170:602–609, 2020. The 11th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 3rd International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops.

[200] D. Maulud and A. M. Abdulazeez. A Review on Linear Regression Comprehensive in Machine Learning. *Journal of Applied Science and Technology Trends*, 1(4):140–147, 2020.

[201] D. Mazumdar and J. Evans. *Modeling of Steelmaking Processes*. CRC Press, 2009.

[202] J. McCarthy. *History of LISP*, page 173–185. Association for Computing Machinery, New York, NY, USA, 1978.

[203] D. L. McGuinness, R. Fikes, J. Hendler, and L. A. Stein. DAML+ OIL: an Ontology Language for the Semantic Web. *IEEE Intelligent Systems*, 17(5):72–80, 2002.

[204] D. L. McGuinness, F. Van Harmelen, et al. OWL Web Ontology Language Overview. *W3C Recommendation*, 10(10):2004, 2004.

[205] Metalworking World Magazine. A short sheet metal history, 2014. Available at: `https://www.metalworkingworldmagazine.com/a-short-sheet-metal-history/`.

[206] D. Meyer and F. Wien. Support Vector Machines. *R News*, 1(3):23–26, 2001.

[207] A. Mileo, A. Abdelrahman, S. Policarpio, and M. Hauswirth. Streamrule: A Nonmonotonic Stream Reasoning System for the Semantic Web. In *Web Reasoning and Rule Systems: 7th International Conference, RR 2013, Mannheim, Germany, July 27-29, 2013. Proceedings 7*, pages 247–252. Springer, 2013.

[208] A. Mileo, M. Dao-Tran, T. Eiter, and M. Fink. Stream Reasoning. *Encyclopedia of Database Systems*, 2017.

[209] T. Miller, J. Jimenez, A. Sharan, and D. Goldstein. Oxygen Steelmaking Processes. *The Making, Shaping and Treating of Steel-Steelmaking and Refining volume*, pages 475–524, 1998.

[210] R. K. Mobley. *An Introduction to Predictive Maintenance*. Elsevier, 2002.

[211] H. Mohajan. The First Industrial Revolution: Creation of a New Global Human Era. *Journal of Social Sciences and Humanities*, 5(4):377–387, 2019.

[212] J. Mokyr and R. H. Strotz. The Second Industrial Revolution, 1870-1914. *Storia dell'economia Mondiale*, 21945(1), 1998.

[213] M. Molnar and C. Houtman. The Advanced Manufacturing Partnership and The Advanced Manufacturing National Program Office. *National Institute of Standards and Technology*, 2011.

[214] D. C. Montgomery, E. A. Peck, and G. G. Vining. *Introduction to Linear Regression Analysis*. John Wiley & Sons, 2021.

[215] W. Moore. Twenty Year Advance in Electric Arc Furnaces for the Production of Iron and Steel. *Transactions of The Electrochemical Society*, 60(1):165, 1931.

[216] M. Mora, F. Wang, J. M. Gómez, and G. Phillips-Wren. Development Methodologies for Ontology-based Knowledge Management Systems: A Review. *Expert Systems*, 39(2):e12851, 2022.

[217] F. Moreno-Seco, L. Micó, and J. Oncina. A Modification of the LAESA Algorithm for Approximated k-NN Classification. *Pattern Recognition Letters*, 24(1-3):47–53, 2003.

[218] S. Muggleton. Alan Turing and the Development of Artificial Intelligence. *AI Communications*, 27(1):3–10, 2014.

[219] M. A. Musen. The Protégé Project: A Look Back and a Look Forward. *AI Matters*, 1(4):4–12, 2015.

[220] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, and S. D. Brown. An Introduction to Decision Tree Modeling. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 18(6):275–285, 2004.

[221] J. Mylopoulos. An Overview of Knowledge Representation. *ACM SIGART Bulletin*, (74):5–12, 1980.

[222] J. Navasaitis, A. Selskienė, and G. Žaldarys. The Study of Trace Elements in Bloomery Iron. *Mater. Sci*, 16(2):113–118, 2010.

[223] A. Newell, J. C. Shaw, and H. A. Simon. Elements of a Theory of Human Problem Solving. *Psychological Review*, 65(3):151, 1958.

[224] A. Newell, J. C. Shaw, and H. A. Simon. Report on a General Problem Solving Program. In *IFIP congress*, volume 256, page 64. Pittsburgh, PA, 1959.

[225] A. Newell and H. Simon. The Logic Theory Machine–A Complex Information Processing System. *IRE Transactions on Information Theory*, 2(3):61–79, 1956.

[226] Y. Nishioka. Industrial Value Chain Reference Architecture, 2017. Available at: `http://files.messe.de/abstracts/75738_Industrial_Value_Chain_Reference_Archite.pdf`.

[227] N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson, and J. Taylor. Industry-scale Knowledge Graphs: Lessons and Challenges: Five diverse technology companies show how it's done. *Queue*, 17(2):48–75, 2019.

[228] N. F. Noy, M. Crubézy, R. W. Fergerson, H. Knublauch, S. W. Tu, J. Vendetti, and M. A. Musen. Protégé-2000: an Open-source Ontology-development and Knowledge-acquisition environment. In *AMIA... Annual Symposium Proceedings. AMIA Symposium*, pages 953–953, 2003.

[229] H.-J. Odenthal, A. Kemminger, F. Krause, L. Sankowski, N. Uebber, and N. Vogl. Review on Modeling and Simulation of the Electric Arc Furnace (EAF). *Steel Research International*, 89(1):1700098, 2018.

[230] OPC Foundation. OPC UA Part 1 - Overview and Concepts. Technical report, OPC Foundation, 2015. Available at: `https://reference.opcfoundation.org/Core/Part1/v104/docs/`.

[231] S. Oster. The Diffusion of Innovation Among Steel Firms: The Basic Oxygen Furnace. *The Bell Journal of Economics*, pages 45–56, 1982.

[232] Ö. L. Özçep, R. Möller, and C. Neuenstadt. A Stream-temporal Query Language for Ontology Based Data Access. In *KI 2014: Advances in Artificial Intelligence: 37th Annual German Conference on AI, Stuttgart, Germany, September 22-26, 2014. Proceedings 37*, pages 183–194. Springer, 2014.

[233] M. O'Connor, S. Tu, C. Nyulas, A. Das, and M. Musen. Querying the Semantic Web with SWRL. In *Advances in Rule Interchange and Applications: International Symposium, RuleML 2007, Orlando, Florida, October 25-26, 2007. Proceedings 1*, pages 155–159. Springer, 2007.

[234] M. J. O'Connor and A. Das. The SWRLTab: An Extensible Environment for Working with SWRL Rules in Protégé-OWL. In *Proc. 2nd Int. Conf. Rules Rule Markup Lang. Semantic Web*, pages 1–2, 2006.

[235] J.-O. Palacio-Niño and F. Berzal. Evaluation Metrics for Unsupervised Learning Algorithms. *arXiv Preprint arXiv:1905.05667*, 2019.

[236] H. Panetto, M. Dassisti, and A. Tursi. ONTO-PDM: Product-driven ONTOlogy for Product Data Management Interoperability within Manufacturing Process Environment. *Advanced Engineering Informatics*, 26(2):334–348, 2012.

[237] P. Patel, M. I. Ali, and A. Sheth. From Raw Data to Smart Manufacturing: AI and Semantic Web of Things for Industry 4.0. *IEEE Intelligent Systems*, 33(4):79–86, 2018.

[238] K. Patroumpas and T. Sellis. Window Specification Over Data Streams. In *International Conference on Extending Database Technology*, pages 445–464. Springer, 2006.

[239] J. G. Peacey and W. G. Davenport. *The Iron Blast Furnace: Theory and Practice*. Elsevier, 2016.

[240] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[241] J. Pearl. *From Bayesian Networks to Causal Networks*, pages 157–182. Springer US, Boston, MA, 1995.

[242] K. Pearson. Contributions to the Mathematical Theory of Evolution. *Philosophical Transactions of the Royal Society of London*, 185:71–110, 1894.

[243] M. Pech, J. Vrchota, and J. Bednář. Predictive maintenance and Intelligent Sensors in Smart Factory. *Sensors*, 21(4):1470, 2021.

[244] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[245] J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. *ACM Transactions on Database Systems (TODS)*, 34(3):1–45, 2009.

[246] J. Pérez, M. Arenas, and C. Gutierrez. Semantics and Complexity of SPARQL. *ACM Transactions on Database Systems (TODS)*, 34(3):1–45, 2009.

[247] S. Peroni. SAMOD: an Agile Methodology for the Development of Ontologies. In *Proceedings of the 13th OWL: Experiences and Directions Workshop and 5th OWL Reasoner Evaluation Workshop (OWLED-ORE 2016)*, pages 1–14, 2016.

[248] D. Pfisterer, K. Romer, D. Bimschas, O. Kleine, R. Mietz, C. Truong, H. Hasemann, A. Kröller, M. Pagel, M. Hauswirth, et al. SPITFIRE: Toward a Semantic Web of Things. *IEEE Communications Magazine*, 49(11):40–48, 2011.

[249] C. Pinkel, C. Binnig, E. Jiménez-Ruiz, W. May, D. Ritze, M. G. Skjæveland, A. Solimando, and E. Kharlamov. RODI: A Benchmark for Automatic Mapping Generation in Relational-to-Ontology Data Integration. In *The Semantic Web. Latest Advances and New Domains*, pages 21–37, Cham, 2015. Springer International Publishing.

[250] Platform Industrie 4.0. Details of the Asset Administration Shell - Part 1, Version 2.0. Technical report, ZVEI, 2019.

[251] Plattform Industrie 4.0. Details of the Asset Administration Shell from idea to implementation. Technical report, Plattform Industrie 4.0, 2018.

[252] Plattform Industrie 4.0. Details of the Asset Administration Shell - From Idea to Implementation. Technical report, Plattform Industrie 4.0, 2019.

[253] M. Poveda-Villalón, A. Gómez-Pérez, and M. C. Suárez-Figueroa. OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(2):7–34, 2014.

[254] V. Presutti, E. Daga, A. Gangemi, and E. Blomqvist. eXtreme Design with Content Ontology Design Patterns. In *Proc. Workshop on Ontology Patterns*, pages 83–97, 2009.

[255] F. Priyatna, O. Corcho, and J. Sequeda. Formalisation and Experiences of R2RML-Based SPARQL to SQL Query Translation Using Morph. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, page 479–490, New York, NY, USA, 2014. Association for Computing Machinery.

[256] M. Proctor. Drools: A Rule Engine for Complex Event Processing. In *Applications of Graph Transformations with Industrial Relevance: 4th International Symposium, AGTIVE 2011, Budapest, Hungary, October 4-7, 2011, Revised Selected and Invited Papers 4*, pages 2–2. Springer, 2012.

[257] S. Profanter, A. Tekat, K. Dorofeev, M. Rickert, and A. Knoll. OPC UA versus ROS, DDS, and MQTT: Performance Evaluation of Industry 4.0 Protocols. In *2019 IEEE International Conference on Industrial Technology (ICIT)*, pages 955–962, 2019.

[258] G. C. Publio, D. Esteves, A. Ławrynowicz, P. Panov, L. Soldatova, T. Soru, J. Vanschoren, and H. Zafar. ML-schema: Exposing the Semantics of Machine Learning with Schemas and Ontologies. *arXiv preprint arXiv:1807.05351*, 2018.

[259] N. Pukkhem. A semantic-based approach for representing successful graduate predictive rules. In *16th International Conference on Advanced Communication Technology*, pages 222–227, 2014.

[260] J. Qadir. Engineering Education in the Era of ChatGPT: Promise and Pitfalls of Generative AI for Education. In *2023 IEEE Global Engineering Education Conference (EDUCON)*, pages 1–9. IEEE, 2023.

[261] J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1:81–106, 1986.

[262] S. Rajbhandari, J. Aryal, J. Osborn, R. Musk, and A. Lucieer. Benchmarking the Applicability of Ontology in Geographic Object-based Image Analysis. *ISPRS International Journal of Geo-Information*, 2017.

[263] D. Rajpathak and R. Chougule. A Generic Ontology Development Framework for Data Integration and Decision Support in a Distributed Environment. *International Journal of Computer Integrated Manufacturing*, 24(2):154–170, 2011.

[264] T. Rattanasawad, M. Buranarach, K. R. Saikaew, and T. Supnithi. A Comparative Study of Rule-based Inference Engines for the Semantic Web. *IEICE TRANSACTIONS on Information and Systems*, 101(1):82–89, 2018.

[265] T. Rattanasawad, K. R. Saikaew, M. Buranarach, and T. Supnithi. A Review and Comparison of Rule Languages and Rule-based Inference Engines for the Semantic Web. In *2013 International Computer Science and Engineering Conference (ICSEC)*, pages 1–6. IEEE, 2013.

[266] A. Ray, K. Mishra, G. Das, and P. Chaudhary. Life of Rolls in a Cold Rolling Mill in a Steel Plant-operation Versus Manufacture. *Engineering Failure Analysis*, 7:55–67, 02 2000.

[267] A. K. Ray, K. Mishra, G. Das, and P. Chaudhary. Life of Rolls in a Cold Rolling Mill in a Steel Plant-operation Versus Manufacture. *Engineering Failure Analysis*, 7(1):55–67, 2000.

[268] A. Razavi, H. Gill, H. Ahlfeldt, and N. Shahsavar. Predicting Metastasis in Breast Cancer: Comparing a Decision Tree with Domain Experts. *Journal of Medical Systems*, 31:263–73, 09 2007.

[269] D. Reinsel, J. Gantz, and J. Rydning. *The Digitization of the World*. From Edge to Core, 2018.

[270] M. Rewaj. Rolling Back the Years. *Aluminium International Today*, 29(3):27–27, 2017.

[271] W. L. Roberts. *Cold Rolling of Steel*. Routledge, 1978.

[272] M. Rodríguez-Muro and M. Rezk. Efficient SPARQL-to-SQL with R2RML Mappings. *Journal of Web Semantics*, 33:141–169, 2015. Ontology-based Data Access.

[273] S. Rongen, N. Nikolova, and M. van der Pas. Modelling with AAS and RDF in Industry 4.0. *Computers in Industry*, 148, 2023.

[274] B. K. Rout, G. Brooks, M. A. Rhamdhani, Z. Li, F. N. Schrama, and J. Sun. Dynamic Model of Basic Oxygen Steelmaking Process Based on Multi-zone Reaction Kinetics: Model Derivation and Validation. *Metallurgical and Materials Transactions B*, 49:537–557, 2018.

[275] C. Rudin. Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[276] M. Ruta, F. Scioscia, and E. Di Sciascio. Enabling the Semantic Web of Things: Framework and Architecture. In *2012 IEEE Sixth International Conference on Semantic Computing*, pages 345–347. IEEE, 2012.

[277] A. L. Samuel. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[278] A. Sarkar, M. R. Naqvi, L. Elmhadhbi, D. Sormaz, B. Archimede, and M. H. Karray. CHAIKMAT 4.0-Commonsense Knowledge and Hybrid Artificial Intelligence for Trusted Flexible Manufacturing. In *International Conference on Flexible Automation and Intelligent Manufacturing*, pages 455–465. Springer, 2022.

[279] A. Sattar, E. S. M. Surin, M. N. Ahmad, M. Ahmad, and A. K. Mahmood. Comparative Analysis of Methodologies for Domain Ontology Development: A Systematic Review. *International Journal of Advanced Computer Science and Applications*, 11(5), 2020.

[280] L. Sauder and S. Williams. A Practical Treatise on the Smelting and Smithing of Bloomery Iron. *Historical metallurgy*, 36(2):122–131, 2002.

[281] R. Schiekofer, S. Grimm, M. M. Brandt, and M. Weyrich. A Formal Mapping Between OPC UA and the Semantic Web. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 33–40. IEEE, 2019.

[282] B. Schmidt and L. Wang. Cloud-enhanced Predictive Maintenance. *The International Journal of Advanced Manufacturing Technology*, 99(1):5–13, 2018.

[283] K. H. Schroder. A Basic Understanding of the Mechanics of Rolling Mill Rolls. *Eisenwerk Sulzau-Werfen, ESW-Handbook*, pages 54–56, 2003.

[284] K. Schweichhart. Reference Architectural Model Industrie 4.0 (RAMI4.0), 2016. Available at: `https://ec.europa.eu/futurium/en/system/files/ged/a2-schweichhart-reference_architectural_model_industrie_4.0_rami_4.0.pdf`.

[285] Science Museum. The Pegasus Computer, may 2015. Available at: `https://blog.sciencemuseum.org.uk/the-pegasus-computer/`.

[286] F. Scioscia and M. Ruta. Building a Semantic Web of Things: Issues and Perspectives in Information Compression. In *2009 IEEE International Conference on Semantic Computing*, pages 589–594. IEEE, 2009.

[287] A. Seif, C. Toro, and H. Akhtar. Implementing Industry 4.0 Asset Administrative Shells in Mini Factories. *Procedia Computer Science*, 159:495 – 504, 2019. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 23rd International Conference KES2019.

[288] C. Sellars and J. Whiteman. Recrystallization and Grain Growth in Hot Rolling. *Metal Science*, 13(3-4):187–194, 1979.

[289] J. F. Sequeda, S. h. Tirmizi, O. Corcho, and D. P. Miranker. Review: Survey of Directly Mapping SQL Databases to the Semantic Web. *Knowl. Eng. Rev.*, 26(4):445–486, Dec. 2011.

[290] S. Shanker. *Gödel's Theorem in Focus*. Psychology Press, 1989.

[291] N. Sharma. The Origin of Data Information Knowledge Wisdom (DIKW) Hierarchy. *Preuzeto*, 25:2021, 2008.

[292] N. Shoaip, A. Rezk, S. El-Sappagh, T. Abuhmed, S. Barakat, and M. Elmogy. Alzheimer's Disease Diagnosis Based on a Semantic Rule-Based Modeling and Reasoning Approach. *Computers, Materials and Continua*, 2021.

[293] A. Sidorenko, W. Motsch, M. Van Bekkum, N. Nikolakis, K. Alexopoulos, and A. Wagner. The MAS4AI Framework for Human-centered Agile and Smart Manufacturing. *Frontiers in Artificial Intelligence*, 6, 2023.

[294] W. C. Siemens. XXVI.—On Smelting Iron and Steel. *Journal of the Chemical Society*, 26:661–678, 1873.

[295] A. Singh, N. Thakur, and A. Sharma. A Review of Supervised Machine Learning Algorithms. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 1310–1315. IEEE, 2016.

[296] A. Singh, A. Yadav, and A. Rana. K-means with Three different Distance Metrics. *International Journal of Computer Applications*, 67(10), 2013.

[297] T. Slimani. Ontology Development: A comparing Study on Tools, Languages and Formalisms. *Indian Journal of Science and Technology*, 8(24):1–12, 2015.

[298] K. Smith. A brief history of metal rolling for sheet and plate products, 2017. Available at: `https://www.innovaltec.com/history-metal-rolling-blog/`.

[299] A. Soylu, E. Kharlamov, D. Zheleznyakov, E. Jimenez-Ruiz, M. Giese, M. G. Skjaeveland, D. Hovland, R. Schlatte, S. Brandt, H. Lie, and I. Horrocks. OptiqueVQS: A Visual Query System Over Ontologies for Industry. *Semantic Web*, 9(5), August 2018. The final publication is available at IOS Press through http://dx.doi.org/10.3233/sw-180293.

[300] L. Sparling. (A) Hot and Cold Rolling. *International Metals Reviews*, 22(1):303–313, 1977.

[301] J. M. Stanton. Galton, Pearson, and the Peas: A Brief History of Linear Regression for Statistics Instructors. *Journal of Statistics Education*, 9(3), 2001.

[302] J. K. Stone. The Origins of Modern Oxygen Steelmaking. *Steel Times*, 228(9):328, 2000.

[303] M. C. Suárez-Figueroa, A. Gómez-Pérez, and M. Fernández-López. The NeOn Methodology for Ontology Engineering. In *Ontology Engineering in a Networked World*, pages 9–34. Springer, 2011.

[304] S. Sun, X. Zheng, J. Villalba-Díez, and J. Ordieres-Meré. Data Handling in Industry 4.0: Interoperability based on Distributed Ledger Technology. *Sensors*, 20(11):3046, 2020.

[305] E. Tantik and R. Anderl. Integrated Data Model and Structure for the Asset Administration Shell in Industrie 4.0. *Procedia CIRP*, 60:86–91, 12 2017.

[306] The Philadelphia Inquirer. Obituary: David Thomas, Iron Manufacturer, 1882. Available at: `https://www.newspapers.com/article/the-philadelphia-inquirer-obituary-davi/58123233/`.

[307] G. W. Thomas. Operational Research in the Steel Company of Wales Limited. *OR*, 14(3):247–262, 1963.

[308] A. S. Thuluva, D. Anicic, and S. Rudolph. Semantic Web of Things for Industry 4.0. *Semantic Web*, 11:883, 2017.

[309] R. Tofighi-Shirazi, I. M. Asăvoae, and P. Elbaz-Vincent. Fine-grained Static Detection of Obfuscation Transforms Using Ensemble-learning and Semantic Reasoning. In *Proceedings of the 9th Workshop on Software Security, Protection, and Reverse Engineering*, pages 1–12, 2019.

[310] Y. N. Toulouevski, I. Y. Zinurov, Y. N. Toulouevski, and I. Y. Zinurov. Modern Steelmaking in Electric Arc Furnaces: History and Development. *Innovation in Electric Arc Furnaces: Scientific Basis for Selection*, pages 1–24, 2013.

[311] J. V. Tucker. The Computer Revolution and Us: Computer Science at Swansea University from the 1960s, 2020. Available at: `https://collections.swansea.ac.uk/s/swansea-2020/page/computer-science`.

[312] A. M. Turing. *Computing Machinery and Intelligence*. Springer, 2009.

[313] O. Ugwu, C. J. Anumba, A. Thorpe, and T. Arciszewski. Building Knowledge Level Ontology for the Collaborative Design of Steel Frame Structures. In *Advances in Intelligent Computing in Engineering—Proceedings of 9th International Workshop of the European Group of Intelligent Computing in Engineering (EG-ICE), August*, pages 01–03, 2002.

[314] M. Uschold. Building Ontologies: Towards a Unifed Methodology. In *Proceedings of 16th Annual Conference of the British Computer Society Specialists Group on Expert Systems*. Citeseer, 1996.

[315] M. Uschold and M. Gruninger. Ontologies: Principles, Methods and Applications. *The Knowledge Engineering Review*, 11(2):93–136, 1996.

[316] M. Uschold and M. Gruninger. Ontologies and Semantics for Seamless Connectivity. *ACM SIGMod Record*, 33(4):58–64, 2004.

[317] M. Uschold and M. King. *Towards a Methodology for Building Ontologies*. Citeseer, 1995.

[318] M. Uslar, A. Göring, R. Heidel, C. Neureiter, D. Engel, and S. Schulte. An Open Source 3D Visualization for the RAMI 4.0 Reference Model. In *VDE-Kongress 2016-Internet der Dinge*, pages P1–5. VDE-Verlag, 2016.

[319] S. Vaidya, P. Ambad, and S. Bhosle. Industry 4.0–A Glimpse. *Procedia Manufacturing*, 20:233–238, 2018.

[320] M. Vegetti, G. P. Henning, and H. P. Leone. Product Ontology: Definition of an Ontology for the Complex Product Modelling Domain. In *Proceedings of the Mercosur Congress on Process Systems Engineering*, 2005.

[321] J. Völker. *Learning Expressive Ontologies*, volume 2. IOS Press, 2009.

[322] C. Wagner, J. Grothoff, U. Epple, R. Drath, S. Malakuti, S. Grüner, M. Hoffmeister, and P. Zimermann. The Role of the Industry 4.0 Asset Administration Shell and the Digital Twin during the Life Cycle of a Plant. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2017.

[323] O. Walavalkar, A. Joshi, T. Finin, Y. Yesha, et al. Streaming Knowledge Bases. In *Proceedings of the Fourth International Workshop on Scalable Semantic Web knowledge Base Systems*, 2008.

[324] S. Wang and O. Xu. Semantic Information Modeling and Implementation Method for Water Conservancy Equipment. *IEEE Access*, 11:133879–133890, 2023.

[325] X. Wang, T. Wong, and Z.-P. Fan. Ontology-based Supply Chain Decision Support for Steel Manufacturers in China. *Expert Systems with Applications*, 40(18):7519–7533, 2013.

[326] X. Wang and D. Zhang. Ontology based context modeling and reasoning using OWL. In *IEEE Annual Conference on Pervasive Computing and Communications Workshops*, 2004.

[327] X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung. Ontology Based Context Modeling and Reasoning using OWL. In *IEEE Annual Conference on Pervasive Computing and Communications Workshops. Proceedings of the Second*, pages 18–22. IEEE, 2004.

[328] K. Wei, J. Sun, and R. Liu. A Review of Asset Administration Shell. In *2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 1460–1465. IEEE, 2019.

[329] S. Wei, J. Hu, Y. Cheng, Y. Ma, and Y. Yu. The Essential Elements of Intelligent Manufacturing System Architecture. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pages 1006–1011. IEEE, 2017.

[330] A. Weiss and S. Ihlenfeldt. Integration of OPC UA Information Models into Enterprise Knowledge Graphs. *Journal of Machine Engineering*, 22(2), 2022.

[331] M. Wopata. The Leading Industry 4.0 Companies, 2019. Available at: `https://iot-analytics.com/the-leading-industry-4-0-companies-2019/`.

[332] World Steel Association. Fact Sheet: Steel and Raw Materials, 2021. Available at: `https://worldsteel.org/wp-content/uploads/Fact-sheet-raw-materials-2023.pdf`.

[333] World Wide Web Consortium. W3C Opens Data on the Web with SPARQL, 2008.

[334] Z. Wusatowski. *Fundamentals of Rolling*. Elsevier, 2013.

[335] G. Xiao, L. Ding, B. Cogrel, and D. Calvanese. Virtual Knowledge Graphs: An Overview of Systems and Use Cases. *Data Intelligence*, 1:201–223, 05 2019.

[336] G. Xiao, D. Lanti, R. Kontchakov, S. Komla-Ebri, E. Güzel-Kalaycı, L. Ding, J. Corman, B. Cogrel, D. Calvanese, and E. Botoeva. The Virtual Knowledge Graph System Ontop. In J. Z. Pan, V. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne, and L. Kagal, editors, *The Semantic Web – ISWC 2020*, pages 259–277, Cham, 2020. Springer International Publishing.

[337] F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, and J. Zhu. Explainable AI: A Brief Survey on History, Research Areas, Approaches and Challenges. In *Natural Language Processing and Chinese Computing: 8th CCF International Conference, NLPCC*, pages 563–574. Springer, 2019.

[338] Y. Xu, Y. Sun, X. Liu, and Y. Zheng. A Digital-Twin-Assisted Fault Diagnosis Using Deep Transfer Learning. *IEEE Access*, 7:19990–19999, 2019.

[339] M. Yahya, J. G. Breslin, and M. I. Ali. Semantic Web and Knowledge Graphs for Industry 4.0. *Applied Sciences*, 11(11):5110, 2021.

[340] R. W. Yates. Discovery of the Process for Making Anthracite Iron. *The Pennsylvania Magazine of History and Biography*, 98(2):206–223, 1974.

[341] X. Ye and S. Hong. An AutomationML/OPC UA-based Industry 4.0 Solution for a Manufacturing System. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 543–550, 09 2018.

[342] X. Ye and S. H. Hong. An AutomationML/OPC UA-based Industry 4.0 Solution for a Manufacturing System. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 543–550. IEEE, 2018.

[343] E. D. Zamani, C. Smyth, S. Gupta, and D. Dennehy. Artificial Intelligence and Big Data Analytics for Supply Chain Resilience: A Systematic Literature Review. *Annals of Operations Research*, 327(2):605–632, 2023.

[344] M. Zeleny. Management Support Systems: Towards Integrated Knowledge Management. *Human Systems Management*, 7(1):59–70, 1987.

[345] F. Zezulka, P. Marcon, I. Vesely, and O. Sajdl. Industry 4.0 – An Introduction in the Phenomenon. *IFAC-PapersOnLine*, 49(25):8 – 12, 2016. 14th IFAC Conference on Programmable Devices and Embedded Systems PDES 2016.

[346] H. Zhou, T. Shen, X. Liu, Y. Zhang, P. Guo, and J. Zhang. Survey of Knowledge Graph Approaches and Applications. *Journal on Artificial Intelligence*, 2(2):89–101, 2020.

Part IV

Appendices

APPENDIX A

Appendix A

£1.4m computer link for SCOW

By PEARCE WRIGHT, Science Reporter

The steel industry is becoming one of the biggest users of advanced computer systems.

In one of the most ambitious single projects undertaken by heavy industry, the Steel Company of Wales is embarking on a £1,400,000 scheme involving six computers.

These will form part of a data-communications network linking production plants centred on Port Talbot, Velindre (Swansea) and Trostre (Llanelly). For this purpose I.B.M. is providing two 360 model-40 computers valued at £630,000, Elliott Automation is supplying two Arch 102 systems worth £120,000, and English Electric Computers is providing a system 4-50 and 4-10 valued at £280,000.

The I.B.M. and Elliott Automation computers will be used at Port Talbot as part of a management information and communication system. It will provide a direct link between production shops and a central production control department. The Elliott Arch machines will act as message-handling exchanges. Information will pass from 24 data-gathering units in the steel mill via one-mile-long transmission cables into the I.B.M. computers.

In a two-way communication system, the central production control department will have immediate information of the situation in the works; the mill operators will receive information about the material they are processing.

The two English Electric machines will be for tinplate mills at Velindre and Trostre. They will be linked over a distance of 10 miles by leased telephone line.

The English Electric systems replace two I.B.M. machines that were installed two years ago.

Figure A.1: The Times article on the 1.4 million investment for six new computers for the Port Talbot steelworks.

BE ALERT —DON'T GET HURT

The Dragon

2,000 DAYS IN SAFETY
See Page 4

THE MONTHLY PUBLICATION OF THE STEEL COMPANY OF WALES LIMITED

MAY 1967

£1.4m. COMPUTER SCHEME

Company embarks on big development programme

OUR Company, which has long been amongst the most advanced users of computing equipment in this country, particularly for production control and process control applications, has now embarked on a new phase of development in this field.

Together with the new computer centre now under construction, the expansion scheme will cost £1.4 million.

The Company currently operates 13 electronic digital computers of various types. Several new projects have recently been initiated.

The new computer centre will cost £400,000 and is being built adjacent to the head office at Abbey Works.

The Company has already taken delivery of one IBM 360 Model 40 computer and will install a second machine of the same type at the end of this year, when the new computer centre will be completed. One of these machines will be used for production control work and the other for accounting and technical applications.

Each 360 computer will have 131,000 bytes of core memory and 7 magnetic disk drives and are together worth £630,000. They will replace an IBM 1401 and an IBM 1410 computer currently in use at Port Talbot.

The Company also operates an ICT Pegasus computer which is used for operational research work. This work will be carried out on the IBM 360 computers in future and the Pegasus has been offered to the Llandaff Technical College, Cardiff.

Telephone link

The Tinplate Division will acquire from English Electric Computers Limited, for installation in May, 1968, two System 4 computers worth £280,000 for use in production control and accounting applications. The equipment will comprise a System 4/50 computer with 65,000 bytes of core memory and four disk files at Velindre Works and a System 4/10 computer with 8,000 bytes of core memory at Trostre Works. The two computers will be about 10 miles apart and are to be directly connected by a leased telephone line capable of transmitting information at the rate of 200 characters per second.

The computer at Trostre will act as an input and output unit of the main machine for data relating to the operations at the works.

These two English Electric computers will replace two IBM 1440 computers which were installed in 1965 and are now in use for more than 120 hours a week.

The Company has ordered from Elliott Automation a set of 24 plant data terminals and two Arch 102 computers worth £120,000 for direct connection to the Company's IBM 360/40 computers. These terminals will be situated in the cold mills at Abbey works and will provide an on-line data collection facility. The equipment forms the first step towards a comprehensive real-time production control system in the Port Talbot Divisions.

Three IBM 1050 data communication terminals have been ordered. These terminals will be situated at the computer centres at Port Talbot, Velindre and Trostre and will be connected by telephone lines. They will be used for rapid communication of orders and notification of deliveries of hot rolled coil which is produced at Port Talbot and used at Trostre and Velindre for making tinplate.

Figure A.2: The Dragon article on the 1.4 million investment for the six new computers.

APPENDIX B

Appendix B

Code for the Algorithms Introduced in the Thesis

The following code is the source code for converting a plain text random forest into a set of semantic-web based rules. The code is written in Java.

Code Listing B.1: Java code for converting a RF into SWRL rules.

```
package algorithm1;

import org.apache.commons.lang3.StringUtils;

import java.io.FileDescriptor;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.PrintStream;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import static com.google.common.base.CaseFormat.LOWER_UNDERSCORE;
import static com.google.common.base.CaseFormat.UPPER_CAMEL;

public class GenerateRules2023 {


    //in db as
// private final static ArrayList<String> features = new ArrayList<>(Arrays.asList("
    Grade_filtered", "trip_sequence_no", "date_time_trip_start", "
    date_time_trip_finish", "mill_id", "stand_id", "top_roll_id", "
    top_roll_grinding_sequence_no", "top_roll_chocking_sequence_no", "bottom_roll_id",
     "bottom_roll_grinding_sequence_no", "bottom_roll_chocking_sequence_no", "
    trip_tonnage", "trip_meterage", "roll_change_reason", "in_mill", "change_reason_id
    ", "damage_flag", "damage_id", "coilpart_length_exit", "coilpart_thick_exit", "
    grade_filtered", "exit_gauge", "exit_width", "exit_weight", "input_coil_id", "
    hr_gauge", "cr_gauge", "input_width", "input_weight", "heatnumber", "c", "si", "s
    ", "p", "mn", "ni", "cu", "sn", "cr", "arsenic", "mo", "n2", "alt", "als", "ti", "
    b", "ca", "ce", "co", "nb", "v", "cev1", "cev2", "cev3", "freec1", "freec2", "
    xsti1", "xsti2", "xsti3", "aln2", "cup", "bn2", "sp", "cunicrmosn", "creq", "
    siform", "mns", "mnsi", "freec3", "fe", "liquidus", "width_reduction", "
    weight_reduction", "c_weighted", "si_weighted", "s_weighted", "p_weighted", "
    mn_weighted", "ni_weighted", "cu_weighted", "sn_weighted", "cr_weighted", "
    arsenic_weighted", "mo_weighted", "n2_weighted", "alt_weighted", "als_weighted", "
    ti_weighted", "b_weighted", "ca_weighted", "ce_weighted", "co_weighted", "
    nb_weighted", "v_weighted", "cev1_weighted", "cev2_weighted", "cev3_weighted", "
    freec1_weighted", "freec2_weighted", "xsti1_weighted", "xsti2_weighted", "
    xsti3_weighted", "aln2_weighted", "cup_weighted", "bn2_weighted", "sp_weighted", "
    cunicrmosn_weighted", "creq_weighted", "siform_weighted", "mns_weighted", "
    mnsi_weighted", "freec3_weighted", "fe_weighted", "liquidus_weighted", "
    tons_rolled_computed", "length_rolled_computed", "coils_rolled_computed", "
    width_reduced_computed", "weight_reduced_computed"));
    private final static ArrayList<String> features = new ArrayList<>(Arrays.asList("
        trip_tonnage", "trip_meterage", "grind_nr", "stand_id", "tons_rolled_computed"
        , "length_rolled_computed", "coils_rolled_computed", "meas_camber", "
        surface_ra","diam_before"));
```

```
// private final static ArrayList<String> newFeatures = new ArrayList<>(Arrays.asList
    ("tripTonnage","tripMeterage","grind_nr","stand_id","tons_rolled_computed","
    length_rolled_computed","coils_rolled_computed","meas_camber","surface_ra"));
// private final static ArrayList<String> newFeatures = new ArrayList<>(Arrays.asList
    ("GradeFiltered", "TripSequenceNo", "DateTimeTripStart", "DateTimeTripFinish", "
    MillID", "StandID", "TopRollID", "TopRollGrindingSequenceNo", "
    TopRollChockingSequenceNo", "BottomRollID", "BottomRollGrindingSequenceNo", "
    BottomRollChockingSequenceNo", "TripTonnage", "TripMeterage", "RollChangeReason",
    "InMill", "ChangeReasonID", "DamageFlag", "DamageID", "CoilpartLengthExit", "
    CoilpartThickExit", "GradeFiltered", "ExitGauge", "ExitWidth", "ExitWeight", "
    InputCoilID", "HRGauge", "CRGauge", "InputWidth", "InputWeight", "HeatNumber", "C
    ", "SI", "S", "P", "MN", "NI", "CU", "SN", "CR", "Arsenic", "MO", "N2", "ALT", "
    ALS", "TI", "B", "CA", "CE", "CO", "NB", "V", "CEV1", "CEV2", "CEV3", "FREEC1", "
    FREEC2", "XSTI1", "XSTI2", "XSTI3", "ALN2", "CUP", "BN2", "SP", "CUNICRMOSN", "
    CREQ", "SIFORM", "MNS", "MNSI", "FREEC3", "FE", "LIQUIDUS", "WidthReduction", "
    WeightReduction", "CWeighted", "SIWeighted", "SWeighted", "PWeighted", "MNWeighted
    ", "NIWeighted", "CUWeighted", "SNWeighted", "CRWeighted", "ArsenicWeighted", "
    MOWeighted", "N2Weighted", "ALTWeighted", "ALSWeighted", "TIWeighted", "BWeighted
    ", "CAWeighted", "CEWeighted", "COWeighted", "NBWeighted", "VWeighted", "
    CEV1Weighted", "CEV2Weighted", "CEV3Weighted", "FREEC1Weighted", "FREEC2Weighted",
     "XSTI1Weighted", "XSTI2Weighted", "XSTI3Weighted", "ALN2Weighted", "CUPWeighted",
     "BN2Weighted", "SPWeighted", "CUNICRMOSNWeighted", "CREQWeighted", "
    SIFORMWeighted", "MNSWeighted", "MNSIWeighted", "FREEC3Weighted", "FEWeighted", "
    LIQUIDUSWeighted", "TonsRolledComputed", "LengthRolledComputed", "
    CoilsRolledComputed", "WidthReducedComputed", "WeightReducedComputed"));

    public static enum Type {
        SCIKITLEARN,
        BRIEMAN,
        FULLWEIGHTING
    }

// private static final String FILE_NAME = "files/rules_77b.txt";
    private static final String FILE_NAME = "files/rules_77_0409.txt";
// private static final String FILE_NAME_OUTPUT = "files/rules_swrl_77b.txt";
    private static final String FILE_NAME_OUTPUT = "files/rules_swrl_77_0409.txt";

    public static ArrayList<String> getNewFeatures(ArrayList<String> features) {
        ArrayList<String> newFeatures = new ArrayList<>();
        for (String feature : features) {
            String result = LOWER_UNDERSCORE.to(UPPER_CAMEL, feature);
// System.out.println("result: " + result);
            newFeatures.add(result);
        }

        return newFeatures;

    }

    public static void main(String[] args) throws Exception {
// getNewFeatures(features);

        //read the file
        String content = new String(Files.readAllBytes(Paths.get(FILE_NAME)));

        //select the RF
        Type type = Type.BRIEMAN;

```

```
62          System.out.println("Type of tree: " + type);
63          //split the file
64          String temp = split(content, type, FILE_NAME_OUTPUT);
65
66  // String newfile = new String(Files.readAllBytes(Paths.get(temp)));
67          toSPARQL(temp, FILE_NAME_OUTPUT);
68      }
69
70      public static void convertRulesToSPARQL(Type rfType, String inputFileName, String
            outputFileName) throws IOException {
71          System.out.println("Converting Random Forest into SPARQL rules...");
72          String content = new String(Files.readAllBytes(Paths.get(inputFileName)));
73          String temp = split(content, rfType, inputFileName);
74          toSPARQL(temp, outputFileName);
75
76      }
77
78      public static String split(String file, Type type, String outputFileName) throws
            IOException {
79          String[] lines = file.split("\n");
80          //send all print statements to output.txt file instead of console
81
82
83          /* This section loops through each line in the file and stores each decision
                tree in an arraylist.
84          Then store all those decision trees in a forest arraylist.
85          * */
86          List<String> decisionTree = new ArrayList<>();
87          List<List<String>> forest = new ArrayList<>();
88
89          for (String line : lines) {
90              StringBuilder temp = new StringBuilder();
91              while (line.contains("|")) {
92                  temp.append("|");
93                  line = line.replaceFirst("\\|", "");
94              }
95              if (line.contains("---")) {
96                  line = line.replace("---", "");
97              }
98              if (!line.contains("******")) {
99                  temp.append(line.trim());
100                 decisionTree.add(temp.toString());
101             } else if (line.contains("*****")) {
102                 forest.add(decisionTree);
103                 decisionTree = new ArrayList<>();
104             }
105         }
106         /* This section of code loops through each tree in the forest array until it
                finds a leaf node.
107          It then creates the path to that leaf node and stores that path in a <tree
                path> arraylist.
108          Then we store all those paths in a <paths> arraylist. Each tree will have a
                list of <tree paths>
109          We then have one arraylist for all the trees and their paths <forest paths>
110          * */
111         int count = 0;
112         PrintStream printStream = new PrintStream(new FileOutputStream(outputFileName))
                ;
```

```
113         System.setOut(printStream);
114         int treeIndex = 1;
115         for (List<String> tree : forest) {
116 
117             StringBuilder rule;
118             List<String> previousConditions = new ArrayList<>();
119             /* a node looks either like:
120              ||| trip_meterage <= 52.33
121              |||| weights: [0.00, 31.00] class: 1.0
122             */
123             for (String node : tree) {
124                 //depth counts the number of pipes |
125                 int depth = (countChar(node, '|')) - 1;
126                 node = node.replace("|", "");
127                 //if node is a condition, remove pipes and add ^ sign afterwards. Then
                        store it in an arraylist with index
128                 if (!(node.contains("class: 1.0") || node.contains("class: 0.0") ||
                        node.contains("class: 2.0")) || node.contains("*****")) {
129                     previousConditions.add(depth, node);
130                     //if node is leaf node, add
131                 } else if (node.contains("class: 1.0") || node.contains("class: 2.0")
                        || node.contains("class: 0.0")) {
132                     count++;
133                     rule = new StringBuilder();
134                     rule.append("Roll_Unit_Trip(?trip) ^ rf:hasRandomForest(?trip, ?rf)
                            ^ ");
135                     //only add the conditions up to the depth
136                     for (int i = 0; i < depth; i++) {
137                         rule.append(previousConditions.get(i));
138                         rule.append(" ^ ");
139                     }
140                     if (node.contains("class: 1.0")) {
141                         if (type.equals(Type.SCIKITLEARN)) {
142                             double[] classes = getClasses(node);
143                             rule.append("swrlx:makeOWLThing(?DT, ?trip) -> rf:
                                    Decision_Tree(?DT) ^ rf:hasDataFrom(?DT, ?trip) ^ rf:
                                    isDecisionTreeOf(?DT,?rf) ^ rf:hasPrediction(?DT, \"").
                                    append(getProbability(classes[1], classes)).append("\"^^
                                    xsd:double) ^ rf:hasTreeIndex(?DT, ").append(treeIndex).
                                    append(")");
144                             System.out.println(rule);
145                         } else if (type.equals(Type.BRIEMAN)) {
146                             rule.append("swrlx:makeOWLThing(?DT, ?trip) -> rf:
                                    Decision_Tree(?DT) ^ rf:hasDataFrom(?DT, ?trip) ^ rf:
                                    isDecisionTreeOf(?DT,?rf) ^ rf:hasPrediction(?DT,
                                    \"1.0\"^^xsd:double) ^ rf:hasTreeIndex(?DT, ").append(
                                    treeIndex).append(")");
147                             System.out.println(rule);
148                         }
149                     } else if (node.contains("class: 0.0")) {
150                         if (type.equals(Type.SCIKITLEARN)) {
151                             double[] classes = getClasses(node);
152                             rule.append("swrlx:makeOWLThing(?DT, ?trip) -> rf:
                                    Decision_Tree(?DT) ^ rf:hasDataFrom(?DT, ?trip) ^ rf:
                                    isDecisionTreeOf(?DT,?rf) ^ rf:hasPrediction(?DT, \"").
                                    append(getProbability(classes[0], classes)).append("\"^^
                                    xsd:double) ^ rf:hasTreeIndex(?DT, ").append(treeIndex).
                                    append(")");
```

```
153                         System.out.println(rule);
154                     } else if (type.equals(Type.BRIEMAN)) {
155                         rule.append("swrlx:makeOWLThing(?DT, ?trip) -> rf:
                                Decision_Tree(?DT) ^ rf:hasDataFrom(?DT, ?trip) ^ rf:
                                isDecisionTreeOf(?DT,?rf) ^ rf:hasPrediction(?DT,
                                \"0.0\"^^xsd:double) ^ rf:hasTreeIndex(?DT, ").append(
                                treeIndex).append(")");
156                         System.out.println(rule);
157                     }
158                 } else if (node.contains("class: 2.0")) {
159                     if (type.equals(Type.BRIEMAN)) {
160                         rule.append("swrlx:makeOWLThing(?DT, ?trip) -> rf:
                                Decision_Tree(?DT) ^ rf:hasDataFrom(?DT, ?trip) ^ rf:
                                isDecisionTreeOf(?DT,?rf) ^ rf:hasPrediction(?DT,
                                \"2.0\"^^xsd:double) ^ rf:hasTreeIndex(?DT, ").append(
                                treeIndex).append(")");
161                         System.out.println(rule);
162                     } else if (type.equals(Type.SCIKITLEARN)){
163                         double[] classes = getClasses(node);
164                         rule.append("swrlx:makeOWLThing(?DT, ?trip) -> rf:
                                Decision_Tree(?DT) ^ rf:hasDataFrom(?DT, ?trip) ^ rf:
                                isDecisionTreeOf(?DT,?rf) ^ rf:hasPrediction(?DT, \"").
                                append(getProbability(classes[2], classes)).append("\"^^
                                xsd:double) ^ rf:hasTreeIndex(?DT, ").append(treeIndex).
                                append(")");
165                         System.out.println(rule);
166
167                     }
168                 }
169
170             }
171         }
172         treeIndex++;
173     }
174     System.err.println("Total leaf nodes found: " + count);
175     return new String(Files.readAllBytes(Paths.get(outputFileName)));
176 }
177 private static double getProbability (double clas, double[] classes) {
178     double sum = 0;
179     for (int i = 0; i < classes.length; i++) {
180         sum+= classes[i];
181     }
182
183     return clas/sum;
184 }
185
186 public static double[] getClasses(String input) {
187     Matcher m = Pattern.compile("\\[(.*?)(])").matcher(input);
188     while (m.find()) {
189         String vals = m.group(1);
190         double val1 = Double.parseDouble(StringUtils.substringBefore(vals,","));
191         double val2 = Double.parseDouble(StringUtils.substringBetween(vals, ",", ",
                "));
192         double val3 = Double.parseDouble(StringUtils.substringAfterLast(vals,","));
193         return new double[]{val1, val2, val3};
194
195     }
196     return new double[]{999.0};
```

```
197     }
198
199     public static void toSPARQL(String input,String output) throws IOException {
200         ArrayList<String> newFeatures = getNewFeatures(features);
201 // for (String newFeature : newFeatures) {
202 // System.err.println(newFeature);
203 // }
204         input = input.replaceAll("^\\R", "");
205         input = input.replaceAll(" {2}", " ");
206         input = input.replaceAll(" && ", " ^ ");
207
208         for (int i = 0; i < newFeatures.size(); i++) {
209             //less than or equal
210             input = input.replaceAll(" " + features.get(i) + " <= (-?\\d*\\.\\d*)",
211                     " has" + newFeatures.get(i) + "(?trip, ?" + newFeatures.get(i) + ")
                            ^ swrlb:lessThanOrEqual(?" + newFeatures.get(i) + ", \"$1\"^^xsd
                            :double)");
212             input = input.replaceAll(" " + features.get(i) + " > (-?\\d*\\.\\d*)",
213                     " has" + newFeatures.get(i) + "(?trip, ?" + newFeatures.get(i) + ")
                            ^ swrlb:greaterThan(?" + newFeatures.get(i) + ", \"$1\"^^xsd:
                            double)");
214         }
215
216         input = input.replaceAll("\\)\\)", ")");
217         input = input.replaceAll("-> class: (\\d).0", "-> hasDamageFlag(?trip, \"$1
                .0\"^^xsd:double)");
218 // System.err.println(newfile);
219         //write back data to file
220
221 // PrintStream printStream = new PrintStream(new FileOutputStream("files/RF-output-
        Kayal.txt"));
222 // System.setOut(printStream);
223         Files.writeString(Paths.get(output), input, StandardOpenOption.CREATE);
224         System.setOut(new PrintStream(new FileOutputStream(FileDescriptor.out)));
225         System.out.println("total features: " + features.size());
226         System.out.println("--- Done ---");
227 // FileUtils.writeStringToFile(new File("RF-output-Kayal2.txt"), newfile,
        StandardCharsets.UTF_8);
228     }
229
230     public static int countChar(String str, char c) {
231         int count = 0;
232
233         for (int i = 0; i < str.length(); i++) {
234             if (str.charAt(i) == c) {
235                 count++;
236             }
237         }
238         return count;
239     }
240 }
```

The following code uses the SWRL-API library, which is an extension of the OWL-API library to provide a semantic reasoning. We utilise this library and reasoning for applying random forest classification. This code is written in Java.

Code Listing B.2: Java code for applying RF classification.

```
1 package algorithm2;
```

```
2
3  import algorithm1.GenerateRulesNew;
4  import com.google.common.collect.Multimap;
5  import org.semanticweb.owlapi.apibinding.OWLManager;
6  import org.semanticweb.owlapi.model.*;
7  import org.semanticweb.owlapi.search.EntitySearcher;
8  import org.semanticweb.owlapi.util.OWLEntityRemover;
9  import org.semanticweb.owlapi.util.SimpleIRIMapper;
10 import org.swrlapi.core.SWRLAPIRule;
11 import org.swrlapi.core.SWRLRuleEngine;
12 import org.swrlapi.exceptions.SWRLBuiltInException;
13 import org.swrlapi.factory.SWRLAPIFactory;
14 import org.swrlapi.parser.SWRLParseException;
15
16 import java.io.BufferedReader;
17 import java.io.File;
18 import java.io.FileReader;
19 import java.io.IOException;
20 import java.nio.file.Files;
21 import java.nio.file.Path;
22 import java.nio.file.Paths;
23 import java.nio.file.StandardOpenOption;
24 import java.util.*;
25 import java.util.regex.Matcher;
26 import java.util.regex.Pattern;
27
28 public class RFClassification {
29     protected static IRI iri() {
30         return IRI.create("http://www.semanticweb.org/sadee/ontologies/2021/1/
              Tatatology#");
31     }
32     protected static IRI iri_rf() {
33         return IRI.create("http://www.semanticweb.org/sadeer/ontologies/2023/0/RF");
34     }
35
36
37     private static final String SPARQL_FILE = "files/rules_swrl_77b.txt";
38
39     public static void main(String[] args) throws OWLOntologyCreationException,
           OWLOntologyStorageException {
40
41         GenerateRulesNew.Type type = GenerateRulesNew.Type.BRIEMAN;
42
43
44         //dataFactory for later
45         OWLDataFactory df = OWLManager.getOWLDataFactory();
46
47         OWLOntologyManager ontologyManager = OWLManager.createOWLOntologyManager();
48
49         ontologyManager.getIRIMappers().add(new SimpleIRIMapper(iri_rf(),IRI.create(new
                File("Ontologies/RF.owl"))));
50         OWLOntology ontology = ontologyManager.loadOntologyFromOntologyDocument(new
               File("Ontologies/scro.owl"));
51
52
53         OWLDocumentFormat format = ontologyManager.getOntologyFormat(ontology);
54         assert format != null;
55         if (format.isPrefixOWLOntologyFormat()) {
```

```
56            //mappings
57            Map<String, String> map = format.asPrefixOWLOntologyFormat().
                  getPrefixName2PrefixMap();
58            System.out.println("map: " + map);
59        }
60
61        // Create a SWRL rule engine using the SWRLAPI
62        SWRLRuleEngine swrlRuleEngine = SWRLAPIFactory.createSWRLRuleEngine(ontology);
63        swrlRuleEngine.importAssertedOWLAxioms();
64
65
66        System.out.println("Generating a Random Forest individual...");
67        OWLClass tClass = df.getOWLClass(IRI.create(iri_rf()+"#Random_Forest"));
68        OWLIndividual rf = df.getOWLNamedIndividual(IRI.create(iri_rf()+"#RF_001"));
69        OWLClassAssertionAxiom classAssertion = df.getOWLClassAssertionAxiom(tClass, rf
              );
70        ontologyManager.addAxiom(ontology, classAssertion);
71
72
73        OWLObjectProperty hasRF = df.getOWLObjectProperty(IRI
74                .create(iri_rf()+"#hasRandomForest"));
75        OWLObjectProperty hasDataFrom = df.getOWLObjectProperty(IRI
76                .create(iri_rf()+"#hasDataFrom"));
77        OWLObjectProperty parsesDataTo = df.getOWLObjectProperty(IRI
78                .create(iri_rf()+"#parsesDataTo"));
79        ontologyManager.addAxiom(ontology,
80                df.getOWLInverseObjectPropertiesAxiom(hasDataFrom, parsesDataTo));
81        ontologyManager.saveOntology(ontology, IRI.create(((new File("Ontologies/
              main1311output.owl")).toURI())));
82        Set<OWLNamedIndividual> individuals;
83        individuals = ontology.getIndividualsInSignature();
84        for (OWLNamedIndividual individual : individuals) {
85            if (individual.toString().contains("roll_") && (individual.toString().
                  contains("_trip"))) {
86                OWLObjectPropertyAssertionAxiom axiom1 = df
87                        .getOWLObjectPropertyAssertionAxiom(hasRF, individual, rf);
88                OWLObjectPropertyAssertionAxiom axiom2 = df
89                        .getOWLObjectPropertyAssertionAxiom(hasDataFrom, rf, individual)
                              ;
90                AddAxiom addAxiom1 = new AddAxiom(ontology, axiom1);
91                AddAxiom addAxiom2 = new AddAxiom(ontology, axiom2);
92
93                //apply changes using the manager
94                ontologyManager.applyChange(addAxiom1);
95                ontologyManager.applyChange(addAxiom2);
96                ontologyManager.saveOntology(ontology, IRI.create(((new File("
                      Ontologies/main1311output.owl")).toURI())));
97            }
98        }
99        System.out.println("Done...");
100
101       //read all rules from rules file
102       ArrayList<SWRLAPIRule> ruleList = new ArrayList<>();
103       System.out.println("---READING TREES FROM FILE---");
104       try (BufferedReader br = new BufferedReader(new FileReader(SPARQL_FILE))) {
105           for (String line; (line = br.readLine()) != null; ) {
106               ruleList.add(swrlRuleEngine.createSWRLRule("temprule"+ ruleList.size(),
                      line));
```

```
107                 if(ruleList.size()%10000==0){
108                     System.out.println("10000 new rules found...");
109                 }
110             }
111             System.out.println("Rules size: " + ruleList.size());
112             System.out.println("Type of RF: " + type);
113 
114             System.out.println("---INFERRING ENGINE---");
115             swrlRuleEngine.infer();
116             System.out.println("---INFERRING ENGINE COMPLETED---");
117 
118             individuals = ontology.getIndividualsInSignature();
119             OWLDataProperty hasPrediction = df.getOWLDataProperty(IRI.create(iri_rf()+"
                    #hasPrediction"));
120 
121             OWLDataProperty hasDiamDifference = df.getOWLDataProperty(IRI.create(iri()
                    + "hasRollRefurbConditionInferred"));
122             System.out.println("---COMPUTING PROBABILITY FOR EACH INDIVIDUAL---");
123 
124             //loop through roll_unit_trip individuals
125             for (OWLIndividual individual: individuals) {
126                 System.out.println("indivdual: " + individual);
127 
128                 if (individual.toString().contains("roll_") && (individual.toString().
                        contains("_trip"))) {
129                     Multimap<OWLObjectPropertyExpression, OWLIndividual> temp =
                            EntitySearcher.getObjectPropertyValues(individual, ontology);
130                     if(type.equals(GenerateRulesNew.Type.SCIKITLEARN)){
131                         double sum = 0.0;
132                         int probCount = 0;
133                         //temp.values(): [<http://www.semanticweb.org/sadee/ontologies
                                /2021/1/Tatatology##ddd2f432_d492_49f6_8408_97c4c5a2bc5f>,
134                         // <http://www.semanticweb.org/sadee/ontologies/2021/1/
                                Tatatology##193dfacb_451b_4fc4_9b25_241c92099767>,
135                         // <http://www.semanticweb.org/sadee/ontologies/2021/1/
                                Tatatology##63f96f18_6e2d_4b06_bc1e_1cb6194b1ad8>,
136                         // <http://www.semanticweb.org/sadee/ontologies/2021/1/
                                Tatatology##a1a7f900_405e_44f8_a1e6_83d10a6d07a7>,
137                         // <http://www.semanticweb.org/sadee/ontologies/2021/1/
                                Tatatology##70d13e18_c865_43a1_bd81_98dd65ae94e1> etc
138                         for (OWLIndividual i : temp.values()) {
139                             probCount++;
140                             //hasDamageFlagProbability2: ["0.13836017569546122"^^xsd:
                                    decimal]
141                             String hasDamageFlagProbability = EntitySearcher.
                                    getDataPropertyValues(i, hasPrediction, ontology).
                                    toString();
142                             String probStart = hasDamageFlagProbability.substring(
                                    hasDamageFlagProbability.indexOf("\"") + 1);
143                             double probEnd = Double.parseDouble(probStart.substring(0,
                                    probStart.indexOf("\"")));
144                             sum += probEnd;
145 
146                         }
147 
148                         double finalProbability = sum / probCount;
149                         String outcome = "";
150                         if (finalProbability > 0.5){
```

```
                    finalProbability = 1.0;
                    outcome = "Bad";
                } else if (finalProbability <= 0.25){
                    finalProbability = 0.0;
                    outcome = "Best";
                } else if (finalProbability > 0.25 && finalProbability <= 0.5) {
                    outcome = "Good";
                }
                OWLDataPropertyAssertionAxiom dataPropertyAssertion = df.
                    getOWLDataPropertyAssertionAxiom(hasDiamDifference,
                    individual, outcome);

                AddAxiom addAxiom1 = new AddAxiom(ontology,
                    dataPropertyAssertion);
                ontologyManager.applyChange(addAxiom1);
            } else if (type.equals(GenerateRulesNew.Type.BRIEMAN)){
                //temp.values(): [<http://www.semanticweb.org/sadee/ontologies
                    /2021/1/Tatatology##ddd2f432_d492_49f6_8408_97c4c5a2bc5f>,
                // <http://www.semanticweb.org/sadee/ontologies/2021/1/
                    Tatatology##193dfacb_451b_4fc4_9b25_241c92099767>,
                // <http://www.semanticweb.org/sadee/ontologies/2021/1/
                    Tatatology##63f96f18_6e2d_4b06_bc1e_1cb6194b1ad8>,
                // <http://www.semanticweb.org/sadee/ontologies/2021/1/
                    Tatatology##a1a7f900_405e_44f8_a1e6_83d10a6d07a7>,
                // <http://www.semanticweb.org/sadee/ontologies/2021/1/
                    Tatatology##70d13e18_c865_43a1_bd81_98dd65ae94e1> etc
                HashMap<String,Integer> map = new HashMap<>();
                int classAs0 = 0;
                int classAs1 = 0;
                int classAs2 = 0;
                System.out.println("temp.val" + temp.values());
                for (OWLIndividual i : temp.values()) {
                    String classOutput = EntitySearcher.getDataPropertyValues(i,
                        hasPrediction, ontology).toString().trim();
                    for (OWLDataProperty owlDataProperty : i.
                        getDataPropertiesInSignature()) {
                        System.out.println("owldataprop: " + owlDataProperty);
                    }
                    System.out.println("classoutput: " + classOutput);
                    if(classOutput.contains("\"1.0\"^^xsd:double")){
                        classAs1++;
                    } else if(classOutput.contains("\"2.0\"^^xsd:double")){
                        classAs2++;
                    } else if(classOutput.contains("\"0.0\"^^xsd:double")){
                        classAs0++;
                    }
                }
                map.put("Bad",classAs0);
                map.put("Best",classAs1);
                map.put("Good",classAs2);

                System.out.println("-------");
                System.out.println("for individual " + individual);
                String finalResult= Collections.max(map.entrySet(), Comparator.
                    comparingInt(Map.Entry::getValue)).getKey();

                System.out.println("has class0(bad): " + classAs0 + ". class1(
                    best): " + classAs1 + ". class2(good): " + classAs2 + ".
```

```
                    output: " + finalResult);
197
198                 OWLDataPropertyAssertionAxiom dataPropertyAssertion = df.
                        getOWLDataPropertyAssertionAxiom(hasDiamDifference,
                        individual, finalResult);
199                 AddAxiom addAxiom1 = new AddAxiom(ontology,
                        dataPropertyAssertion);
200                 ontologyManager.applyChange(addAxiom1);
201             }
202
203
204
205         }
206     }
207
208     System.out.println("---DELETING ALL INTERMEDIATE AND TEMPORARY DATA---");
209     //delete all temp data properties and temp rules
210     deleteAllTempRules(swrlRuleEngine, ruleList);
211
212     System.out.println("---SAVING ONTOLOGY---");
213     //save and update the ontology
214
215     //remove intermediate values and decision tree individuals from ontology
216     deleteTempIndividuals(df,ontology,ontologyManager);
217
218     //add expert knowledge to the roll individuals (recent roll refurbish
            knowledge)
219     temp(individuals,ontology,df,ontologyManager);
220
221
222     //expert rules
223     ruleList.add(swrlRuleEngine.createSWRLRule("expertrule"+ ruleList.size(),"
            Roll(?r) ^ hasRollUnitTrip(?r, ?trip) ^ hasRollRefurbConditionInferred
            (?trip, ?cond) ^ swrlb:equal(?cond, \"Best\") ^
            hasRecentHighStockRemoval(?trip, ?stock) ^ swrlb:equal(?stock, false) ^
             hasTripTonnage(?trip, ?tonnage) ^ swrlb:lessThan(?tonnage, 4500) ->
            hasStatus(?trip, \"Continue rolling: predicted condition is best on a
            healthy roll. Recommended tonnage of 4500.\")"));
224     ruleList.add(swrlRuleEngine.createSWRLRule("expertrule"+ ruleList.size(),"
            Roll(?r) ^ hasRollUnitTrip(?r, ?trip) ^ hasRollRefurbConditionInferred
            (?trip, ?cond) ^ swrlb:equal(?cond, \"Best\") ^
            hasRecentHighStockRemoval(?trip, ?stock) ^ swrlb:equal(?stock, false) ^
             hasTripTonnage(?trip, ?tonnage) ^ swrlb:greaterThanOrEqual(?tonnage,
            4500) -> hasStatus(?trip, \"Stop rolling: predicted condition is best
            on a healthy roll. Recommended tonnage of 4500 has been exceeded.\")"))
            ;
225     ruleList.add(swrlRuleEngine.createSWRLRule("expertrule"+ ruleList.size(),"
            Roll(?r) ^ hasRollUnitTrip(?r, ?trip) ^ hasRollRefurbConditionInferred
            (?trip, ?cond) ^ swrlb:equal(?cond, \"Best\") ^
            hasRecentHighStockRemoval(?trip, ?stock) ^ swrlb:equal(?stock, true) ^
            hasTripTonnage(?trip, ?tonnage) ^ swrlb:lessThan(?tonnage, 4000) ->
            hasStatus(?trip, \"Continue rolling: predicted condition is best but
            recent high stock removal may affect roll. Recommended tonnage of
            4000.\")"));
226     ruleList.add(swrlRuleEngine.createSWRLRule("expertrule"+ ruleList.size(),"
            Roll(?r) ^ hasRollUnitTrip(?r, ?trip) ^ hasRollRefurbConditionInferred
            (?trip, ?cond) ^ swrlb:equal(?cond, \"Best\") ^
            hasRecentHighStockRemoval(?trip, ?stock) ^ swrlb:equal(?stock, true) ^
```

```
            hasTripTonnage(?trip, ?tonnage) ^ swrlb:greaterThanOrEqual(?tonnage,
                4000) -> hasStatus(?trip, \"Stop rolling: predicted condition is best
                but recent high stock removal may affect roll. Recommended tonnage of
                4000 has been exceeded.\")"));
227
228         ruleList.add(swrlRuleEngine.createSWRLRule("expertrule"+ ruleList.size(),"
                Roll(?r) ^ hasRollUnitTrip(?r, ?trip) ^ hasRollRefurbConditionInferred
                (?trip, ?cond) ^ swrlb:equal(?cond, \"Good\") ^
                hasRecentHighStockRemoval(?trip, ?stock) ^ swrlb:equal(?stock, false) ^
                 hasTripTonnage(?trip, ?tonnage) ^ swrlb:lessThan(?tonnage, 4000) ->
                hasStatus(?trip, \"Continue rolling: predicted condition is good on a
                healthy roll. Recommended tonnage of 4000.\")"));
229         ruleList.add(swrlRuleEngine.createSWRLRule("expertrule"+ ruleList.size(),"
                Roll(?r) ^ hasRollUnitTrip(?r, ?trip) ^ hasRollRefurbConditionInferred
                (?trip, ?cond) ^ swrlb:equal(?cond, \"Good\") ^
                hasRecentHighStockRemoval(?trip, ?stock) ^ swrlb:equal(?stock, false) ^
                 hasTripTonnage(?trip, ?tonnage) ^ swrlb:greaterThanOrEqual(?tonnage,
                4000) -> hasStatus(?trip, \"Stop rolling: predicted condition is good
                on a healthy roll. Recommended tonnage of 4000 has been exceeded.\")"))
                ;
230         ruleList.add(swrlRuleEngine.createSWRLRule("expertrule"+ ruleList.size(),"
                Roll(?r) ^ hasRollUnitTrip(?r, ?trip) ^ hasRollRefurbConditionInferred
                (?trip, ?cond) ^ swrlb:equal(?cond, \"Good\") ^
                hasRecentHighStockRemoval(?trip, ?stock) ^ swrlb:equal(?stock, true) ^
                hasTripTonnage(?trip, ?tonnage) ^ swrlb:lessThan(?tonnage, 3500) ->
                hasStatus(?trip, \"Continue rolling: predicted condition is good but
                recent high stock removal may affect roll. Recommended tonnage of
                3500.\")"));
231         ruleList.add(swrlRuleEngine.createSWRLRule("expertrule"+ ruleList.size(),"
                Roll(?r) ^ hasRollUnitTrip(?r, ?trip) ^ hasRollRefurbConditionInferred
                (?trip, ?cond) ^ swrlb:equal(?cond, \"Good\") ^
                hasRecentHighStockRemoval(?trip, ?stock) ^ swrlb:equal(?stock, true) ^
                hasTripTonnage(?trip, ?tonnage) ^ swrlb:greaterThanOrEqual(?tonnage,
                3500) -> hasStatus(?trip, \"Stop rolling: predicted condition is good
                but recent high stock removal may affect roll. Recommended tonnage of
                3500 has been exceeded.\")"));
232
233         ruleList.add(swrlRuleEngine.createSWRLRule("expertrule"+ ruleList.size(),"
                Roll(?r) ^ hasRollUnitTrip(?r, ?trip) ^ hasRollRefurbConditionInferred
                (?trip, ?cond) ^ swrlb:equal(?cond, \"Bad\") ^
                hasRecentHighStockRemoval(?trip, ?stock) ^ swrlb:equal(?stock, false) ^
                 hasTripTonnage(?trip, ?tonnage) ^ swrlb:lessThan(?tonnage, 3000) ->
                hasStatus(?trip, \"Continue rolling: predicted condition is bad on a
                previously healthy roll. Recommended tonnage of 3000.\")"));
234         ruleList.add(swrlRuleEngine.createSWRLRule("expertrule"+ ruleList.size(),"
                Roll(?r) ^ hasRollUnitTrip(?r, ?trip) ^ hasRollRefurbConditionInferred
                (?trip, ?cond) ^ swrlb:equal(?cond, \"Bad\") ^
                hasRecentHighStockRemoval(?trip, ?stock) ^ swrlb:equal(?stock, false) ^
                 hasTripTonnage(?trip, ?tonnage) ^ swrlb:greaterThanOrEqual(?tonnage,
                3000) -> hasStatus(?trip, \"Stop rolling: predicted condition is bad on
                 a previously healthy roll. Recommended tonnage of 3000 has been
                exceeded.\")"));
235         ruleList.add(swrlRuleEngine.createSWRLRule("expertrule"+ ruleList.size(),"
                Roll(?r) ^ hasRollUnitTrip(?r, ?trip) ^ hasRollRefurbConditionInferred
                (?trip, ?cond) ^ swrlb:equal(?cond, \"Bad\") ^
                hasRecentHighStockRemoval(?trip, ?stock) ^ swrlb:equal(?stock, true) ^
                hasTripTonnage(?trip, ?tonnage) ^ swrlb:lessThan(?tonnage, 2500) ->
                hasStatus(?trip, \"Stop rolling: predicted condition is bad on a roll
```

```
                that was previously damaged.\")"));
236             ruleList.add(swrlRuleEngine.createSWRLRule("expertrule"+ ruleList.size(),"
                    Roll(?r) ^ hasRollUnitTrip(?r, ?trip) ^ hasRollRefurbConditionInferred
                    (?trip, ?cond) ^ swrlb:equal(?cond, \"Bad\") ^
                    hasRecentHighStockRemoval(?trip, ?stock) ^ swrlb:equal(?stock, true) ^
                    hasTripTonnage(?trip, ?tonnage) ^ swrlb:greaterThanOrEqual(?tonnage,
                    2500) -> hasStatus(?trip, \"Stop rolling: predicted condition is bad on
                     a roll that was previously damaged. Recommended tonnage of 2500 has
                    been exceeded.\")"));
237 // ruleList.add(swrlRuleEngine.createSWRLRule("expertrule"+ ruleList.size(),"Roll(?r)
        ^ hasRollUnitTrip(?r, ?trip) ^ hasRollRefurbConditionInferred(?trip, ?cond) ^
        swrlb:equal(?cond, \"Bad\") ^ hasRecentHighStockRemoval(?trip, ?stock) ^ swrlb:
        equal(?stock, false) ^ hasTripTonnage(?trip, ?tonnage) ^ swrlb:lessThan(?tonnage,
        1000) -> hasStatus(?trip, \"Stop rolling: predicted condition is bad on a roll
        that was previously damaged. Recommended tonnage of 3000.\")"));
238
239
240
241             Path path2 = Paths.get("Ontologies/scro.owl");
242             String file2 = new String(Files.readAllBytes(path2));
243             file2 = file2.replaceAll("<Literal>Bad</Literal>", "<Literal datatypeIRI=\"
                    http://www.w3.org/2001/XMLSchema#string\">Bad</Literal>");
244             file2 = file2.replaceAll("<Literal>Good</Literal>", "<Literal datatypeIRI
                    =\"http://www.w3.org/2001/XMLSchema#string\">Good</Literal>");
245             file2 = file2.replaceAll("<Literal>Best</Literal>", "<Literal datatypeIRI
                    =\"http://www.w3.org/2001/XMLSchema#string\">Best</Literal>");
246             Files.writeString(path2, file2, StandardOpenOption.CREATE);
247
248             swrlRuleEngine.infer();
249
250             ontologyManager.saveOntology(ontology, IRI.create(((new File("Ontologies/
                    main1311output.owl")).toURI())));
251             System.out.println("---SUCCESS---");
252         } catch (SWRLParseException | IOException | OWLOntologyStorageException e) {
253             e.printStackTrace();
254         } catch (SWRLBuiltInException e) {
255             throw new RuntimeException(e);
256         }
257     }
258
259     private static void temp(Set<OWLNamedIndividual> individuals, OWLOntology ontology
            , OWLDataFactory df,OWLOntologyManager manager) throws
            OWLOntologyStorageException {
260
261         /** example of RDF triple
262          //subject -> predicate -> object
263          //roll_1234 -> hasRollGrinding -> 1234_grind_1
264          */
265
266         for (OWLIndividual individual: individuals) {
267
268             //loop through individuals that contain "roll" but not "_trip" (subject)
269             if (individual.toString().contains("roll_") && (!individual.toString().
                    contains("_trip"))) {
270                 System.out.println("indivdual: " + individual);
271
272                 //loop through the triples of that individual
273                 Multimap<OWLObjectPropertyExpression, OWLIndividual> objects =
```

```
                EntitySearcher.getObjectPropertyValues(individual, ontology);
274             boolean output = false;
275             //loop through the objects (third part of the RDF triple)
276             outerloop:
277             for (OWLIndividual value : objects.values()) {
278                 if (value.toString().contains("trip")) {
279                     Set<OWLDataPropertyAssertionAxiom> properties=ontology.
                            getDataPropertyAssertionAxioms(value);
280                     //loop through the data properties of that grinding
281                     for (OWLDataPropertyAssertionAxiom ax: properties) {
282                         //if the data property is the diam difference then retrieve
                                the value as a double
283                         if (ax.getProperty().toString().contains("hasGrindNr")) {
284                             Matcher m = Pattern.compile("\\d+\\.\\d+").matcher(ax.
                                    getAxiomWithoutAnnotations().toString());
285                             double val = 0;
286                             while (m.find()) {
287                                 val = Double.parseDouble(m.group(0));
288                                 System.out.println("val@@@ is " + val);
289                             }
290                         }
291                     }
292                 }
293                 //String[] parts = value.toString().split(".", 4); // Split into at
                        most 3 parts
294                 if(value.toString().contains("grind")){
295                     System.out.println("grind: " + value);
296                     String[] parts = value.toString().split("_", 3); // Split into
                            at most 3 parts
297                     if (parts.length >= 3) {
298                         String remainingString = parts[2];
299                         System.out.println(remainingString);
300                     } else {
301                         // Handle case when there are fewer than 3 parts
302                         System.out.println("No second occurrence of '_' found.");
303                     }
304                     Set<OWLDataPropertyAssertionAxiom> properties=ontology.
                            getDataPropertyAssertionAxioms(value);
305
306                     //loop through the data properties of that grinding
307                     for (OWLDataPropertyAssertionAxiom ax: properties) {
308                         //if the data property is the diam difference then retrieve
                                the value as a double
309                         if(ax.getProperty().toString().contains("DiamDifference")) {
310                             Matcher m = Pattern.compile("\\d+\\.\\d+").matcher(ax.
                                    getAxiomWithoutAnnotations().toString());
311                             double val = 0;
312                             while (m.find()) {
313                                  val = Double.parseDouble(m.group(0));
314                                 System.out.println("val is " + val);
315                             }
316                             //if any of the diam difference data properties for this
                                    individual (subject) value
317                             // is greater than 0.5 then it's considered high stock
                                    removal and flagged as so
318                             if (val >= 2) {
319                                 output = true;
320                                 System.out.println(" damage is " + output);
```

```
321                                 break outerloop; //exit nested loop all together
322                             } else {
323                                 output = false;
324                                 System.out.println(" damage is " + output);
325                             }
326                         }
327 
328                             }
329                         }
330                 }
331                 //store the high stock removal flag in the "hasRecentHighStockRemoval"
                        data property
332                 OWLDataProperty hasRecentHighStockRemoval = df.getOWLDataProperty(IRI.
                        create("hasRecentHighStockRemoval"));
333                 System.out.println("output: " + output);
334                 OWLDataPropertyAssertionAxiom dataPropertyAssertion =
335                         df.getOWLDataPropertyAssertionAxiom(hasRecentHighStockRemoval,
                                individual, output);
336                 manager.addAxiom(ontology, dataPropertyAssertion);
337                 manager.saveOntology(ontology, IRI.create(((new File("Ontologies/
                        main1311output.owl")).toURI())));
338 
339             }
340 
341              }
342         }
343 // }
344 
345     private static void deleteTempIndividuals(OWLDataFactory df, OWLOntology ontology,
             OWLOntologyManager ontologyManager) throws OWLOntologyStorageException {
346         System.out.println("Removing all intermediate individuals and decision trees...
                ");
347         OWLClass decisionTreeClass = df.getOWLClass(IRI.create(iri_rf()+"#Decision_Tree
                "));
348         OWLEntityRemover remover = new OWLEntityRemover(Collections.singleton(ontology)
                );
349         int i = 0;
350         for (OWLClassAssertionAxiom classAssertionAxiom : ontology.
                getClassAssertionAxioms(decisionTreeClass)) {
351             classAssertionAxiom.getIndividual().asOWLNamedIndividual().accept(remover);
352             i++;
353         }
354         System.out.println(i+" individuals removed.");
355         ontologyManager.applyChanges(remover.getChanges());
356 // ontologyManager.saveOntology(ontology);
357         remover.reset();
358     }
359 
360     private static void deleteAllTempRules(SWRLRuleEngine swrlRuleEngine,List<
            SWRLAPIRule> ruleList) {
361         for(int i = 0; i< ruleList.size(); i++){
362             swrlRuleEngine.deleteSWRLRule("temprule"+i);
363         }
364     }
365 
366     private static void deleteObjectPropertyAssertions(OWLOntologyManager
            ontologyManager,
367                                                        OWLDataFactory df,OWLOntology
```

```
                                                    ontology,
368                                                OWLObjectProperty property,
                                                    OWLIndividual name1,
                                                    OWLIndividual name2) {
369
370         OWLObjectPropertyAssertionAxiom assertion = df.
                getOWLObjectPropertyAssertionAxiom(property, name1, name2);
371         ontologyManager.applyChange(new RemoveAxiom(ontology, assertion));
372
373     }
374
375     private static void deleteExistingHasDamageFlagValues(OWLDataFactory df,
            OWLDataProperty hasDamageFlag,
376                                                     OWLIndividual name,OWLOntology
                                                        ontology,OWLOntologyManager
                                                        ontologyManager) {
377         OWLDataPropertyAssertionAxiom dataPropertyRemove0 = df.
                getOWLDataPropertyAssertionAxiom(hasDamageFlag, name,0.0d);
378         OWLDataPropertyAssertionAxiom dataPropertyRemove1 = df.
                getOWLDataPropertyAssertionAxiom(hasDamageFlag, name,1.0d);
379         OWLDataPropertyAssertionAxiom dataPropertyRemove2 = df.
                getOWLDataPropertyAssertionAxiom(hasDamageFlag, name,0);
380         OWLDataPropertyAssertionAxiom dataPropertyRemove3 = df.
                getOWLDataPropertyAssertionAxiom(hasDamageFlag, name,1);
381
382         RemoveAxiom removeAxiom0 = new RemoveAxiom(ontology,dataPropertyRemove0);
383         RemoveAxiom removeAxiom1 = new RemoveAxiom(ontology,dataPropertyRemove1);
384         RemoveAxiom removeAxiom2 = new RemoveAxiom(ontology,dataPropertyRemove2);
385         RemoveAxiom removeAxiom3 = new RemoveAxiom(ontology,dataPropertyRemove3);
386         ontologyManager.applyChange(removeAxiom0);
387         ontologyManager.applyChange(removeAxiom1);
388         ontologyManager.applyChange(removeAxiom2);
389         ontologyManager.applyChange(removeAxiom3);
390     }
391
392
393 }
```

Code for the Continuous Streamer

The following code is the source code for initialising the continuous streamer in the context of cold rolling.

Code Listing B.3: Java code for setting up the continuous streamer.

```
1  import eu.larkc.csparql.cep.api.RdfQuadruple;
2  import eu.larkc.csparql.cep.api.RdfStream;
3  import org.apache.commons.lang3.StringUtils;
4
5  import java.io.*;
6  import java.math.BigDecimal;
7  import java.math.RoundingMode;
8  import java.util.ArrayList;
9  import java.util.Random;
10
11 public class Streamer extends RdfStream implements Runnable {
12
```

```
13      private boolean keepRunning = false;
14
15      //ontology string
16      private String scro = "http://www.semanticweb.org/sadee/ontologies/2021/1/
            Tatatology#";
17
18      public Streamer(final String iri) {
19          super(iri);
20          System.out.println("Streamer running...");
21      }
22      public void pleaseStop() {
23          keepRunning = false;
24      }
25      @Override
26      public void run() {
27          keepRunning = true;
28          long timeStamp;
29          int randomVal;
30
31          ArrayList<String> featureList = new ArrayList<>();
32          try (BufferedReader br = new BufferedReader(new FileReader("features.txt"))) {
33              for (String line; (line = br.readLine()) != null; ) {
34                  featureList.add(line);
35              }
36          } catch (IOException e) {
37              e.printStackTrace();
38          }
39
40          while (keepRunning) {
41
42              try {
43                  Thread.sleep(3000); //3000 = 3 seconds
44              } catch (InterruptedException e) {
45                  e.printStackTrace();
46              }
47                  timeStamp = System.currentTimeMillis();
48                  RdfQuadruple rdfQuad;
49                  randomVal = (int) randomWithRange(900000, 999999);
50
51                  for (int i = 0; i < featureList.size(); i++) {
52                      String featureName = StringUtils.substringBefore(featureList.get(i),
                            ",");
53                      double min = Double.parseDouble(StringUtils.substringBetween(
                            featureList.get(i), ",", ","));
54                      double max = Double.parseDouble(StringUtils.substringAfterLast(
                            featureList.get(i), ","));
55
56                      if (((featureName.endsWith("id") || featureName.endsWith("ID")) && !
                            featureName.endsWith("InputCoilID"))) {
57                          rdfQuad = (new RdfQuadruple(super.getIRI() + "roll_unit_trip_" +
                                randomVal,
58                                  this.scro + featureName, "\"" + wholeNumberWithRange(min,
                                        max) + "\"" + "^^http://www.w3.org/2001/XMLSchema#
                                    double", timeStamp));
59                      } else {
60                          rdfQuad = (new RdfQuadruple(super.getIRI() + "roll_unit_trip_" +
                                randomVal,
61                                  this.scro + featureName, "\"" + randomWithRange(min, max)
```

```
                                    + "\"" + "^^http://www.w3.org/2001/XMLSchema#double"
                                    , timeStamp));
62                  }
63                      //System.out.println(rdfQuad); //each RDF can be printed here
64                  this.put(rdfQuad);
65              }
66              System.err.println(" New RDF triples found...");
67
68
69          }
70      }
71
72
73      double wholeNumberWithRange(double min, double max) {
74          return (int) (Math.random() * (max - min)) + min;
75      }
76
77      double randomWithRange(double min, double max) {
78          Random r = new Random();
79          double num = min + (max - min) * r.nextDouble();
80
81          return new BigDecimal(num).setScale(3, RoundingMode.HALF_UP).doubleValue();
82      }
83  }
```

The following code is the source code for the main application for the streamer.

Code Listing B.4: Java code for running the continuous streamer application.

```
1   import eu.larkc.csparql.core.engine.CsparqlEngine;
2   import eu.larkc.csparql.core.engine.CsparqlEngineImpl;
3   import eu.larkc.csparql.core.engine.CsparqlQueryResultProxy;
4   import org.semanticweb.owlapi.apibinding.OWLManager;
5   import org.semanticweb.owlapi.model.*;
6   import org.semanticweb.owlapi.reasoner.InferenceType;
7   import org.semanticweb.owlapi.reasoner.OWLReasoner;
8   import org.semanticweb.owlapi.reasoner.OWLReasonerFactory;
9   import org.semanticweb.owlapi.reasoner.structural.StructuralReasonerFactory;
10  import org.semanticweb.owlapi.util.DefaultPrefixManager;
11  import org.semanticweb.owlapi.util.OWLEntityRemover;
12  import org.swrlapi.core.SWRLAPIRule;
13  import org.swrlapi.core.SWRLRuleEngine;
14  import org.swrlapi.exceptions.SWRLBuiltInException;
15  import org.swrlapi.factory.SWRLAPIFactory;
16  import org.swrlapi.parser.SWRLParseException;
17  import org.swrlapi.sqwrl.SQWRLQueryEngine;
18
19  import java.io.*;
20  import java.text.ParseException;
21  import java.util.*;
22
23  public class StreamerMain {
24      protected static IRI iri() {
25          return IRI.create("http://www.semanticweb.org/sadee/ontologies/2021/1/
                Tatatology#");
26      }
27
28      public enum Type {
29          SCIKITLEARN,
```

```
30          BRIEMAN
31      }
32      public static void main(String[] args) throws OWLOntologyCreationException,
            SWRLParseException, SWRLBuiltInException, OWLOntologyStorageException,
            FileNotFoundException {
33
34          Type type = Type.SCIKITLEARN;
35
36          // Create OWLOntology instance using the OWLAPI
37          OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
38          OWLOntology ontology = manager.loadOntologyFromOntologyDocument(new File("
                Ontologies/scro.owl"));
39          String ontologyIRI = "http://www.semanticweb.org/sadee/ontologies/2021/1/
                Tatatology";
40          DefaultPrefixManager prefixManager = new DefaultPrefixManager(null, null,
                ontologyIRI);
41
42
43          OWLReasonerFactory reasonerFactory = new StructuralReasonerFactory();
44          OWLReasoner reasoner = reasonerFactory.createReasoner(ontology);
45          reasoner.precomputeInferences(InferenceType.CLASS_HIERARCHY);
46
47
48          // Create a SWRL rule engine using the SWRLAPI
49          SWRLRuleEngine swrlRuleEngine = SWRLAPIFactory.createSWRLRuleEngine(ontology);
50          swrlRuleEngine.importAssertedOWLAxioms();
51
52          SWRLRuleEngine expertRuleEngine = SWRLAPIFactory.createSWRLRuleEngine(ontology)
                ;
53
54          ArrayList<SWRLAPIRule> expertRuleList = new ArrayList<>();
55          expertRuleList.add(expertRuleEngine.createSWRLRule("expertRule" +
                expertRuleList.size(),
56                  "Roll_Unit_Trip(?trip) ^ hasDamageFlag(?trip, ?DamageFlag) ^ swrlb:
                        equal(?DamageFlag, \"1.0\"^^xsd:double) " +
57                          "^ hasTonsRolledComputed(?trip, ?TonsRolledComputed) ^ swrlb:
                                greaterThanOrEqual(?TonsRolledComputed, \"50000\"^^xsd:
                                double) -> hasStatus(?trip, \"Stop Operation\"^^xsd:string)"
                                ));
58          expertRuleList.add(expertRuleEngine.createSWRLRule("expertRule" +
                expertRuleList.size(),
59                  "Roll_Unit_Trip(?trip) ^ hasDamageFlag(?trip, ?DamageFlag) ^ swrlb:
                        lessThan(?DamageFlag, \"1.0\"^^xsd:double) " +
60                          "^ hasTonsRolledComputed(?trip, ?TonsRolledComputed) ^ swrlb:
                                lessThan(?TonsRolledComputed, \"50000\"^^xsd:double) ->
                                hasStatus(?trip, \"Continue Operation\"^^xsd:string)"));
61          expertRuleList.add(expertRuleEngine.createSWRLRule("expertRule" +
                expertRuleList.size(),
62                  "Roll_Unit_Trip(?trip) ^ hasDamageFlag(?trip, ?DamageFlag) ^ swrlb:
                        lessThanOrEqual(?DamageFlag, \"0.0\"^^xsd:double) -> hasStatus(?trip
                        , \"Continue Operation\"^^xsd:string)"));
63          SQWRLQueryEngine queryEngine = SWRLAPIFactory.createSQWRLQueryEngine(ontology);
64          OWLDataFactory owlDataFactory = manager.getOWLDataFactory();
65
66          //read all rules from rules file
67          ArrayList<SWRLAPIRule> ruleList = new ArrayList<>();
68
69          System.out.println("---READING TREES FROM FILE---");
```

```
70          try (BufferedReader br = new BufferedReader(new FileReader("rules_proba_output.
                txt"))) {
71              for (String line; (line = br.readLine()) != null; ) {
72                  ruleList.add(swrlRuleEngine.createSWRLRule("temprule" + ruleList.size()
                        , line));
73              }
74              System.out.println("Rules size: " + ruleList.size());
75
76
77          } catch (IOException | SWRLParseException | SWRLBuiltInException e) {
78              e.printStackTrace();
79          }
80          OWLClass RUTclass = owlDataFactory.getOWLClass(IRI.create(iri() + "
                Roll_Unit_Trip"));//
81          Set<OWLNamedIndividual> individuals;
82          individuals = ontology.getIndividualsInSignature();
83          HashMap<OWLIndividual,Set<OWLDataPropertyAssertionAxiom>> test = new HashMap
                <>();
84          for (OWLIndividual individual: individuals) {
85              if (individual.toString().contains("roll_unit_trip")) {
86                  Set<OWLDataPropertyAssertionAxiom> properties=ontology.
                        getDataPropertyAssertionAxioms(individual);
87                  test.put(individual,properties);
88              }
89          }
90
91          String query;
92          Streamer testGenerator;
93
94          // init C-SPARQL Engine
95          CsparqlEngine engine = new CsparqlEngineImpl();
96          engine.initialize(true);
97
98          //query is large so stored externally. Timer set to 40s
99          query = new Scanner(new File("query.txt")).useDelimiter("\\Z").next();
100
101         testGenerator = new Streamer("http://www.semanticweb.org/sadee/ontologies
                /2021/1/Tatatology#");
102
103         // Register an RDF Stream
104         engine.registerStream(testGenerator);
105
106         // Start the stream
107         final Thread t = new Thread(testGenerator);
108         t.start();
109
110         // Register a C-SPARQL query
111         CsparqlQueryResultProxy c1 = null;
112
113         try {
114             c1 = engine.registerQuery(query, false);
115         } catch (ParseException e) {
116             e.printStackTrace();
117         }
118
119         // Attach a Result Formatter to the query result proxy
120         if (c1 != null) {
121             c1.addObserver(new SPARQLResults(swrlRuleEngine,expertRuleEngine, ontology,
```

```
                reasoner, prefixManager, manager, owlDataFactory, queryEngine));
122         }
123 
124         try {
125             Thread.sleep(300000);// run streamer for this long
126         } catch (InterruptedException e) {
127             e.printStackTrace();
128         }
129 
130         // unregister query and stream afterwards
131         if (c1 != null) {
132             engine.unregisterQuery(c1.getId());
133         }
134         engine.unregisterStream(testGenerator.getIRI());
135         testGenerator.pleaseStop();
136 
137         //engine.unregisterStream(testGenerator.getIRI());
138 
139         //remove all RF rules
140         deleteAllTempRules(swrlRuleEngine,ruleList);
141 
142         //delete any new roll_unit_trips
143         OWLEntityRemover remover = new OWLEntityRemover(Collections.singleton(ontology)
                );
144         for (OWLClassAssertionAxiom classAssertionAxiom : ontology.
                getClassAssertionAxioms(RUTclass)) {
145             classAssertionAxiom.getIndividual().asOWLNamedIndividual().accept(remover);
146             manager.applyChanges(remover.getChanges());
147             remover.reset();
148         }
149         //re-add all roll_unit_Trip individuals and their data properties
150         for(Map.Entry<OWLIndividual, Set<OWLDataPropertyAssertionAxiom>> entry : test.
                entrySet()) {
151             OWLIndividual key = entry.getKey();
152             Set<OWLDataPropertyAssertionAxiom> value = entry.getValue();
153 
154             OWLClassAssertionAxiom classAssertion = owlDataFactory.
                    getOWLClassAssertionAxiom(RUTclass, key);
155             manager.addAxiom(ontology, classAssertion);
156             for (OWLDataPropertyAssertionAxiom owlDataPropertyAssertionAxiom : value) {
157                 manager.addAxiom(ontology, owlDataPropertyAssertionAxiom);
158             }
159         }
160 
161 // if(input.equals("yes")){
162 // for(Map.Entry<OWLIndividual, Set<OWLDataPropertyAssertionAxiom>> entry :
        RDFResultsFormatter.returnData().entrySet()) {
163 // OWLIndividual key = entry.getKey();
164 // Set<OWLDataPropertyAssertionAxiom> value = entry.getValue();
165 //
166 // OWLClassAssertionAxiom classAssertion = owlDataFactory.getOWLClassAssertionAxiom(
        RUTclass, key);
167 // manager.addAxiom(ontology, classAssertion);
168 // for (OWLDataPropertyAssertionAxiom owlDataPropertyAssertionAxiom : value) {
169 // manager.addAxiom(ontology, owlDataPropertyAssertionAxiom);
170 // }
171 // }
172 //
```

```
// }
        System.out.println("Saving ontology...");
        manager.saveOntology(ontology, IRI.create(((new File("Ontologies/cswrloutput.
            owl")).toURI())));
    }
    private static void deleteAllTempRules(SWRLRuleEngine swrlRuleEngine, List<
        SWRLAPIRule> ruleList) {
        for(int i = 0; i< ruleList.size(); i++){
            swrlRuleEngine.deleteSWRLRule("temprule"+i);
        }
    }

}
```

The following code is the source code for the SPARQL results formatting.

Code Listing B.5: Java code for the SPARQL results formater during streaming.

```
import com.google.common.collect.Multimap;

import eu.larkc.csparql.common.RDFTable;
import eu.larkc.csparql.common.RDFTuple;
import eu.larkc.csparql.core.ResultFormatter;
import org.apache.commons.lang3.time.StopWatch;
import org.semanticweb.owlapi.model.*;
import org.semanticweb.owlapi.reasoner.OWLReasoner;
import org.semanticweb.owlapi.search.EntitySearcher;
import org.semanticweb.owlapi.util.DefaultPrefixManager;
import org.semanticweb.owlapi.util.OWLEntityRemover;
import org.swrlapi.core.SWRLRuleEngine;
import org.swrlapi.sqwrl.SQWRLQueryEngine;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.*;

public class SPARQLResults extends ResultFormatter {

    private final SWRLRuleEngine ruleEngine;
    private final SWRLRuleEngine expertRuleEngine;
    private final OWLOntology ontology;
    private final OWLReasoner reasoner;
    private final DefaultPrefixManager prefixManager;
    private final OWLOntologyManager manager;
    private final OWLDataFactory owlDataFactory;
    private final SQWRLQueryEngine queryEngine;

    SPARQLResults(SWRLRuleEngine ruleEngine, SWRLRuleEngine expertRuleEngine,
        OWLOntology newOnto, OWLReasoner reasoner,
                DefaultPrefixManager prefixManager, OWLOntologyManager manager,
                    OWLDataFactory owlDataFactory, SQWRLQueryEngine queryEngine) {
        this.ruleEngine = ruleEngine;
        this.expertRuleEngine = expertRuleEngine;
        this.ontology = newOnto;
        this.reasoner = reasoner;
        this.prefixManager = prefixManager;
        this.manager = manager;
        this.owlDataFactory = owlDataFactory;
        this.queryEngine = queryEngine;
```

```
40      }
41
42      private static HashMap<OWLIndividual,Set<OWLDataPropertyAssertionAxiom>> test =
            new HashMap<>();
43      public static HashMap<OWLIndividual,Set<OWLDataPropertyAssertionAxiom>> returnData
            (){
44          return test;
45      }
46
47      protected static IRI iri() {
48          return IRI.create("http://www.semanticweb.org/sadee/ontologies/2021/1/
                Tatatology#");
49      }
50      @Override
51      public void update(Observable o, Object arg) {
52          RDFTable res = (RDFTable) arg;
53          System.out.println("\nres: "+res.getTuples());
54          //System.out.println("C-SPARQL output");
55          System.out.println("Data stream found " + res.size() + " new result(s) at
                timestamp: " + System.currentTimeMillis() + ".");
56          System.out.println("-------");
57          int numrator = 1;
58          long totProcTime = 0;
59
60          ArrayList<String> featureNames = new ArrayList(res.getNames());
61          //start timer
62          StopWatch stopWatch = new StopWatch();
63          OWLClass RUTclass = owlDataFactory.getOWLClass(IRI.create(iri() + "
                Roll_Unit_Trip"));//
64
65          //temporarily delete all individuals of RUT class
66          OWLEntityRemover remover = new OWLEntityRemover(Collections.singleton(ontology)
                );
67          for (OWLClassAssertionAxiom classAssertionAxiom : ontology.
                getClassAssertionAxioms(RUTclass)) {
68              classAssertionAxiom.getIndividual().asOWLNamedIndividual().accept(
                    remover);
69              manager.applyChanges(remover.getChanges());
70              remover.reset();
71          }
72
73          for (final RDFTuple t : res) {
74  // System.out.println("--------");
75              Set<OWLDataPropertyAssertionAxiom> axiomList = new HashSet<>();
76              String rut = t.get(0);
77              DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss");
78              Date date = new Date();
79              String obsTime = dateFormat.format(date);
80
81              //print C-SPARQL results
82              System.out.println("#" + numrator + " time: [" + obsTime + "] found: " +
                    rut + ". ");
83
84              //create roll_unit_trip individual as part of rut class
85              OWLIndividual newRUT = owlDataFactory.getOWLNamedIndividual(IRI.create(rut)
                    );
86              OWLClassAssertionAxiom tmpClassAxiom = owlDataFactory.
                    getOWLClassAssertionAxiom(RUTclass, newRUT);
```

```
87              manager.applyChange(new AddAxiom(ontology, tmpClassAxiom));
88
89              //add tripSeeqnumber on roll_unit_trip individual
90              String tripSeqNum = t.get(0);
91              String tripSeqNumString = tripSeqNum.substring(tripSeqNum.indexOf("
                    roll_unit_trip_"));
92              tripSeqNumString = tripSeqNumString.replaceAll("[^\\d]*","");
93              int tripSeqNumDouble = Integer.parseInt(tripSeqNumString);
94              OWLDataProperty hasTripSeq = owlDataFactory.getOWLDataProperty(IRI.create(
                    iri() + "hasTripSequenceNo"));
95              OWLAxiom dataTypeAssertionTrip = owlDataFactory.
                    getOWLDataPropertyAssertionAxiom(hasTripSeq, newRUT, tripSeqNumDouble);
96              manager.applyChange(new AddAxiom(ontology, dataTypeAssertionTrip));
97
98              axiomList.add(owlDataFactory.getOWLDataPropertyAssertionAxiom(hasTripSeq,
                    newRUT,tripSeqNumDouble));
99
100             for (int i = 1; i < res.getNames().size(); i++) {
101                 String feature = featureNames.get(i);
102                 String featureValue = t.get(i).substring(t.get(i).indexOf("\"") + 1);
103                 double featureValueDouble = Double.parseDouble(featureValue.substring
                        (0, featureValue.indexOf("\"")));
104
105                 OWLDataProperty dataProperty = owlDataFactory.getOWLDataProperty(IRI.
                        create(iri() + "has" + feature));
106                 OWLAxiom dataTypeAssertion = owlDataFactory.
                        getOWLDataPropertyAssertionAxiom(dataProperty, newRUT,
                        featureValueDouble);
107
108                 manager.applyChange(new AddAxiom(ontology, dataTypeAssertion));
109                 axiomList.add(owlDataFactory.getOWLDataPropertyAssertionAxiom(
                        dataProperty,newRUT,featureValueDouble));
110
111             }
112             test.put(newRUT,axiomList);
113             numrator++;
114
115         }
116         stopWatch.start();
117         System.out.println("-------");
118         //RUN SWRL engine
119         System.out.println("Inferring rule engine...");
120         //remove any existing decision_tree individuals
121         deleteTempIndividuals(owlDataFactory, ontology, manager);
122         ruleEngine.infer();
123         System.out.println("Inferring rule engine complete");
124
125         try {
126             fromOtherCode(ontology, owlDataFactory, manager, StreamerMain.Type.
                    SCIKITLEARN);
127         } catch (OWLOntologyStorageException e) {
128             e.printStackTrace();
129         }
130         System.out.println("Applying Expert Rules to new data...");
131         expertRuleEngine.infer();
132         try {
133             returnStatus(ontology,owlDataFactory,manager, StreamerMain.Type.SCIKITLEARN
                    );
```

```
134         } catch (OWLOntologyStorageException e) {
135             e.printStackTrace();
136         }
137         stopWatch.stop();
138         totProcTime = totProcTime + stopWatch.getTime();
139         System.out.println("(Total time)"
140                 + ": " + totProcTime + " ms"
141         );
142         stopWatch.reset();
143 
144     }
145 
146     private static void deleteTempIndividuals(OWLDataFactory df, OWLOntology ontology,
            OWLOntologyManager ontologyManager) {
147         OWLClass decisionTreeClass = df.getOWLClass(IRI.create(iri() + "Decision_Tree")
                );
148         OWLEntityRemover remover = new OWLEntityRemover(Collections.singleton(ontology)
                );
149         for (OWLClassAssertionAxiom classAssertionAxiom : ontology.
                getClassAssertionAxioms(decisionTreeClass)) {
150             classAssertionAxiom.getIndividual().asOWLNamedIndividual().accept(remover);
151         }
152         ontologyManager.applyChanges(remover.getChanges());
153         remover.reset();
154     }
155 
156     private double randomWithRange(double min, double max) {
157         double range = (max - min) + 1;
158         return (double) (Math.random() * range) + min;
159     }
160 
161     private static void returnStatus(OWLOntology ontology, OWLDataFactory df,
            OWLOntologyManager ontologyManager, StreamerMain.Type type) throws
            OWLOntologyStorageException {
162         Set<OWLNamedIndividual> individuals;
163         individuals = ontology.getIndividualsInSignature();
164 
165         OWLDataProperty hasStatus = df.getOWLDataProperty(IRI.create(iri() + "
                hasDamageFlag"));
166         for (OWLIndividual individual : individuals) {
167             if (individual.toString().contains("roll_unit_trip")) {
168 // System.out.println("individual: " + individual);
169                 Multimap<OWLDataPropertyExpression, OWLLiteral> temp = EntitySearcher.
                        getDataPropertyValues(individual, ontology);
170                 for (OWLLiteral i : temp.values()) {
171                     if(i.toString().contains("^^xsd:string")){
172                         System.out.println("Result says: " + i + " for individual: " +
                                individual);
173                     }
174                     //hasDamageFlagProbability2: ["0.13836017569546122"^^xsd:decimal]
175 // System.out.println("DT "+probCount+" " +i.toString()+" for " +individual+": ");
176 // String hasStatusText = EntitySearcher.get
177 // System.out.println("status val: " + hasStatusText);
178                 }
179             }
180         }
181 
182     }
```

```
183
184     private static void fromOtherCode(OWLOntology ontology, OWLDataFactory df,
            OWLOntologyManager ontologyManager, StreamerMain.Type type) throws
            OWLOntologyStorageException {
185         Set<OWLNamedIndividual> individuals;
186         individuals = ontology.getIndividualsInSignature();
187
188         OWLDataProperty hasPrediction = df.getOWLDataProperty(IRI.create(iri() + "
                hasPrediction"));
189         OWLDataProperty hasDamageFlag = df.getOWLDataProperty(IRI.create(iri() + "
                hasDamageFlag"));
190
191         System.out.println("---COMPUTING PROBABILITY FOR EACH INDIVIDUAL---");
192
193         //loop through roll_unit_trip individuals
194         for (OWLIndividual individual : individuals) {
195             if (individual.toString().contains("roll_unit_trip")) {
196                 Multimap<OWLObjectPropertyExpression, OWLIndividual> temp =
                        EntitySearcher.getObjectPropertyValues(individual, ontology);
197
198                 if (type.equals(StreamerMain.Type.SCIKITLEARN)) {
199                     double sum = 0.0;
200                     int probCount = 0;
201                     //temp.values(): [<http://www.semanticweb.org/sadee/ontologies
                            /2021/1/Tatatology##ddd2f432_d492_49f6_8408_97c4c5a2bc5f>,
202                     // <http://www.semanticweb.org/sadee/ontologies/2021/1/Tatatology
                            ##193dfacb_451b_4fc4_9b25_241c92099767>,
203                     // <http://www.semanticweb.org/sadee/ontologies/2021/1/Tatatology
                            ##63f96f18_6e2d_4b06_bc1e_1cb6194b1ad8>,
204                     // <http://www.semanticweb.org/sadee/ontologies/2021/1/Tatatology##
                            a1a7f900_405e_44f8_a1e6_83d10a6d07a7>,
205                     // <http://www.semanticweb.org/sadee/ontologies/2021/1/Tatatology
                            ##70d13e18_c865_43a1_bd81_98dd65ae94e1> etc
206                     for (OWLIndividual i : temp.values()) {
207                         probCount++;
208                         //hasDamageFlagProbability2: ["0.13836017569546122"^^xsd:decimal
                                ]
209 // System.out.println("DT "+probCount+" " +i.toString()+" for " +individual+": ");
210                         String hasDamageFlagProbability = EntitySearcher.
                                getDataPropertyValues(i, hasPrediction, ontology).toString()
                                ;
211                         String probStart = hasDamageFlagProbability.substring(
                                hasDamageFlagProbability.indexOf("\"") + 1);
212                         double probEnd = Double.parseDouble(probStart.substring(0,
                                probStart.indexOf("\"")));
213                         sum += probEnd;
214
215                     }
216
217                     double finalProbability = sum / probCount;
218
219                     if (finalProbability > 0.5) {
220                         finalProbability = 1.0;
221                     } else {
222                         finalProbability = 0.0;
223                     }
224                     deleteExistingHasDamageFlagValues(df, hasDamageFlag, individual,
                            ontology, ontologyManager);
```

```
225                     OWLDataPropertyAssertionAxiom dataPropertyAssertion = df.
                            getOWLDataPropertyAssertionAxiom(hasDamageFlag, individual,
                            finalProbability);
226                     System.out.println("damageFlag prediction: " + finalProbability + "
                            for individual " + individual);
227
228                     AddAxiom addAxiom1 = new AddAxiom(ontology, dataPropertyAssertion);
229                     ontologyManager.applyChange(addAxiom1);
230                 } else if (type.equals(StreamerMain.Type.BRIEMAN)) {
231                     //temp.values(): [<http://www.semanticweb.org/sadee/ontologies
                            /2021/1/Tatatology##ddd2f432_d492_49f6_8408_97c4c5a2bc5f>,
232                     // <http://www.semanticweb.org/sadee/ontologies/2021/1/Tatatology
                            ##193dfacb_451b_4fc4_9b25_241c92099767>,
233                     // <http://www.semanticweb.org/sadee/ontologies/2021/1/Tatatology
                            ##63f96f18_6e2d_4b06_bc1e_1cb6194b1ad8>,
234                     // <http://www.semanticweb.org/sadee/ontologies/2021/1/Tatatology##
                            a1a7f900_405e_44f8_a1e6_83d10a6d07a7>,
235                     // <http://www.semanticweb.org/sadee/ontologies/2021/1/Tatatology
                            ##70d13e18_c865_43a1_bd81_98dd65ae94e1> etc
236                     int classAs0 = 0;
237                     int classAs1 = 0;
238                     for (OWLIndividual i : temp.values()) {
239                         //hasDamageFlagProbability2: ["0.13836017569546122"^^xsd:decimal
                                ]
240                         String classOutput = EntitySearcher.getDataPropertyValues(i,
                                hasPrediction, ontology).toString();
241 // System.out.println("hasDamageFlagProbability2:" + classOutput );
242                         if (classOutput.contains("1")) {
243                             classAs1++;
244                         } else {
245                             classAs0++;
246                         }
247 // String probStart = hasDamageFlagProbability2.substring(hasDamageFlagProbability2.
    indexOf("\"") + 1);
248 // double probEnd = Double.parseDouble(probStart.substring(0, probStart.indexOf("\""))
    );
249 // sum += probEnd;
250
251                     }
252                     System.out.println("-------");
253                     System.out.println("for individual " + individual);
254                     System.out.println("has class0: " + classAs0 + ". class1: " +
                            classAs1);
255
256 // double finalProbability = sum / probCount;
257                     deleteExistingHasDamageFlagValues(df, hasDamageFlag, individual,
                            ontology, ontologyManager);
258                     double finalResult;
259                     if (classAs0 >= classAs1) {
260                         finalResult = 0.0;
261                     } else {
262                         finalResult = 1.0;
263                     }
264                     OWLDataPropertyAssertionAxiom dataPropertyAssertion = df.
                            getOWLDataPropertyAssertionAxiom(hasDamageFlag, individual,
                            finalResult);
265                     System.out.println("damageFlag prediction: " + finalResult + " for
                            individual " + individual);
```

```
                    AddAxiom addAxiom1 = new AddAxiom(ontology, dataPropertyAssertion);
                    ontologyManager.applyChange(addAxiom1);

                }
            }

        }

    }

    private static void deleteExistingHasDamageFlagValues(OWLDataFactory df,
        OWLDataProperty hasDamageFlag,
                                                          OWLIndividual name, OWLOntology
                                                              ontology, OWLOntologyManager
                                                              ontologyManager) {
        OWLDataPropertyAssertionAxiom dataPropertyRemove0 = df.
            getOWLDataPropertyAssertionAxiom(hasDamageFlag, name, 0.0d);
        OWLDataPropertyAssertionAxiom dataPropertyRemove1 = df.
            getOWLDataPropertyAssertionAxiom(hasDamageFlag, name, 1.0d);
        OWLDataPropertyAssertionAxiom dataPropertyRemove2 = df.
            getOWLDataPropertyAssertionAxiom(hasDamageFlag, name, 0);
        OWLDataPropertyAssertionAxiom dataPropertyRemove3 = df.
            getOWLDataPropertyAssertionAxiom(hasDamageFlag, name, 1);

        RemoveAxiom removeAxiom0 = new RemoveAxiom(ontology, dataPropertyRemove0);
        RemoveAxiom removeAxiom1 = new RemoveAxiom(ontology, dataPropertyRemove1);
        RemoveAxiom removeAxiom2 = new RemoveAxiom(ontology, dataPropertyRemove2);
        RemoveAxiom removeAxiom3 = new RemoveAxiom(ontology, dataPropertyRemove3);
        ontologyManager.applyChange(removeAxiom0);
        ontologyManager.applyChange(removeAxiom1);
        ontologyManager.applyChange(removeAxiom2);
        ontologyManager.applyChange(removeAxiom3);
    }
}
```

Code for the Creation of a Random Forest Using the Sci-kit Learn Library

The following code uses the sci-kit learn library to create a random forest based on the input data we provide. This code is written in Python using Jupyter Notebook IDE and with the Numpy library.

Code Listing B.6: Python code to create a random forest and update `hasRecentHighStockRemoval` flag.

```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "176dcdfb",
   "metadata": {},
   "outputs": [],
   "source": [
    "import numpy as np\n",
    "import pandas as pd\n",
    "import seaborn as sns\n",
```

```
    "import datetime as dt\n",
    "import matplotlib.pyplot as plt\n",
    "import re\n",
    "from pandasql import sqldf\n",
    "from sklearn.model_selection import train_test_split\n",
    "from sklearn.metrics import r2_score\n",
    "from sklearn.metrics import mean_squared_error, mean_absolute_error,
        mean_absolute_percentage_error\n",
    "from sklearn.tree import DecisionTreeRegressor\n",
    "\n",
    "import sys\n",
    "\n",
    "sys.path.insert(0,'..')\n",
    "from src.utils import *\n",
    "from src.models import *\n",
    "from src.preprocess import *\n",
    "\n",
    "from sklearn.tree import export_text\n",
    "from sklearn.preprocessing import StandardScaler\n",
    "from sklearn.tree import export_text\n",
    "import seaborn as sns\n",
    "from sklearn.tree import DecisionTreeRegressor\n",
    "from sklearn.utils import check_array\n",
    "from sklearn.utils.validation import check_is_fitted\n",
    "import joblib\n",
    "from sklearn.ensemble import RandomForestClassifier\n",
    "from sklearn.tree import export_text"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "02d96498",
   "metadata": {},
   "outputs": [],
   "source": [
    "df = pd.read_csv('../raw_data/main_data.csv')\n",
    "filtered_df = df[~df['damage_flag'].isin([1])]"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "6a141f0f-2eba-401e-bdb6-e9774101a683",
   "metadata": {},
   "outputs": [],
   "source": [
    "\n",
    "df = pd.read_csv('../raw_data/main_data.csv')\n",
    "values_to_remove = [1]\n",
    "df = df[~df['damage_flag'].isin(values_to_remove)]\n",
    "\n",
    "df = df[df.diam_difference <= 5]\n",
    "\n",
    "\n",
    "# mill_id 2 has only 71 entries so drop all \n",
    "df = df[df.mill_id != 2]\n",
    "\n",
```

```
    "df.dropna(subset=['diam_difference'])\n",
    "df.reset_index(drop=True, inplace=True)\n",
    " \n",
    "labels = []\n",
    "\n",
    "#output labels\n",
    "for row in range(len(df.diam_difference)): \n",
    " if df.diam_difference.iloc[row] > 0.5:\n",
    " labels.append('Bad')\n",
    " elif df.diam_difference.iloc[row] <= 0.25:\n",
    " labels.append('Best')\n",
    " elif df.diam_difference.iloc[row] > 0.25 and df.diam_difference.iloc[row] <=
      0.5:\n",
    " labels.append('Good')\n",
    " else:\n",
    " continue\n",
    "\n",
    "df['labels'] = pd.DataFrame(labels)\n",
    "\n",
    "#features for RF \n",
    "dff = df[[\n",
    " 'trip_sequence_no',\n",
    " 'roll_id',\n",
    " 'trip_tonnage',\n",
    " 'trip_meterage',\n",
    " 'grind_nr',\n",
    " 'stand_id',\n",
    " 'tons_rolled_computed',\n",
    " 'length_rolled_computed',\n",
    " 'coils_rolled_computed',\n",
    " 'meas_camber',\n",
    " 'surface_ra',\n",
    " 'diam_before',\n",
    " 'labels']]\n",
    "dff.set_index(['trip_sequence_no','roll_id'],inplace=True) #set id for index\n",
    "df_corr = dff.corr()\n",
    "\n",
    "target = ['labels']\n",
    "predictors = [feat for feat in dff.columns if feat not in target and feat !=
      'labels']\n",
    "\n",
    "X = dff[predictors]\n",
    "y = dff[target]\n",
    "\n",
    "\n",
    "X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
      random_state=42) #split and train data\n",
    "\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "fbfb7bfb",
   "metadata": {},
   "outputs": [],
   "source": [
    "#retrieve test data\n",
```

```
    "X_test_with_id =
        X_test.rename_axis(['trip_sequence_no','roll_id']).reset_index()\n",
    "test_data = np.concatenate((X_test_with_id,y_test), axis = 1)\n",
    "test_data = pd.DataFrame(test_data)\n",
    "test_data.rename(columns={0:'trip_sequence_no',
        1:'roll_id',2:'trip_tonnage',3:'trip_meterage',4:'grind_nr',5:'stand_id',6:'tons_rolled_compu
        inplace=True)\n",
    "test_data.trip_sequence_no = test_data.trip_sequence_no.astype(np.int64)\n",
    "test_data.roll_id = test_data.roll_id.astype(np.int64)\n",
    "test_data.to_csv('test_data_0409.csv', index=True, encoding='utf-8')"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "3b485cbe-888b-4380-8ed5-8cef3241d429",
   "metadata": {},
   "outputs": [],
   "source": [
    "clf = RandomForestClassifier(n_estimators=20, max_depth=22)\n",
    "clf = clf.fit(X_train, y_train)\n",
    "\n",
    "#Predict the response for test dataset\n",
    "y_pred = clf.predict(X_test)\n",
    "\n",
    "error_analysis(y_test, y_pred)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "a69ab45d",
   "metadata": {},
   "outputs": [],
   "source": [
    "# confusion matrix\n",
    "from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay\n",
    "import seaborn as sn\n",
    "labels = ['Bad', 'Best', 'Good']\n",
    "cm = confusion_matrix(y_test, y_pred)\n",
    "cmd = ConfusionMatrixDisplay(cm, display_labels= ['Bad', 'Best', 'Good'])\n",
    "#sn.heatmap(cmd, annot=True)\n",
    "cmd.plot()\n",
    "cmd.ax_.set(xlabel='Predicted', ylabel='True')\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "dbe96958",
   "metadata": {},
   "outputs": [],
   "source": [
    "from sklearn.tree import export_text\n",
    "#expert rf into plain text file\n",
    "rules = ''\n",
    "line = '********************************************** \\n'\n",
    "for t in clf.estimators_:\n",
```

```
    " rules += export_text(t, feature_names=predictors, max_depth=22,
        show_weights=True)\n",
    " rules += line"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "89baf8cb",
   "metadata": {},
   "outputs": [],
   "source": [
    "with open('rules_77_0409.txt','w') as f:\n",
    " f.write(rules)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "c9b0632a",
   "metadata": {},
   "outputs": [],
   "source": [
    "#the following code is for checking the last five grindings for a list of
        rolls\n",
    "\n",
    "#list of rolls we want to check\n",
    "roll_list = pd.read_csv('mycsv0409.csv') \n",
    "#full data set including the historical grindings of each roll\n",
    "df = pd.read_csv('../raw_data/main_data.csv')\n",
    "values_to_remove = [1]\n",
    "df = df[~df['damage_flag'].isin(values_to_remove)]\n",
    "\n",
    "\n",
    "# mill_id 2 has only 71 entries so drop all \n",
    "df = df[df.mill_id != 2]\n",
    "\n",
    "df.dropna(subset=['diam_difference'])\n",
    "df.reset_index(drop=True, inplace=True)\n",
    "\n",
    "L = roll_list['roll_id'].tolist()\n",
    "df = df.sort_values(['roll_id', 'grind_nr'], ascending=[True, True])"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "3c23c558",
   "metadata": {},
   "outputs": [],
   "source": [
    "roll_list = pd.read_csv('mycsv0609.csv') \n",
    "roll_list.set_index('trip', inplace=True)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
```

```
   "id": "a1a686c4",
   "metadata": {},
   "outputs": [],
   "source": [
    "df2 = pd.DataFrame()\n",
    "\n",
    "#loop through the roll list, and match it with the bigger data set to find the last five grindings of each roll in roll list\n",
    "for (id,nr) in roll_list.itertuples(index=False):\n",
    " data = df[ (df['roll_id'] == id) & (df['grind_nr'] < nr) ]\n",
    " data = data.sort_values(['roll_id', 'grind_nr','diam_difference'], ascending=[True, True,False])\n",
    " data2 = data.groupby('roll_id').tail(5) #get last 5 entries for each roll\n",
    " \n",
    " if data2['diam_difference'].max() > 1: #if any of the entries are greater than specific value\n",
    " roll_list.loc[roll_list['roll_id'] == id, 'hasRecentHighStockRemoval'] = True #set flag to true\n",
    " else:\n",
    " roll_list.loc[roll_list['roll_id'] == id, 'hasRecentHighStockRemoval'] = False#flag to false\n",
    "print(roll_list)\n",
    "\n",
    "print(data2[['roll_id','grind_nr','diam_difference']])\n",
    "\n",
    "roll_list.to_csv(\"highStockRemoval1509.csv\", encoding='utf-8', index=True)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "a66ad127",
   "metadata": {},
   "outputs": [],
   "source": [
    "#test with one data vlaue example \n",
    "df2 = pd.DataFrame()\n",
    "data = df[ (df['roll_id'] == 1524) & (df['grind_nr'] < 137) ]\n",
    "data = data.sort_values(['roll_id', 'grind_nr','diam_difference'], ascending=[True, True,False])\n",
    "data2 = data.groupby('roll_id').tail(5)\n",
    " \n",
    "if data2['diam_difference'].max() > 1:\n",
    " roll_list.loc[roll_list['roll_id'] == 1524, 'hasRecentHighStockRemoval'] = True\n",
    "else:\n",
    " roll_list.loc[roll_list['roll_id'] == 1524, 'hasRecentHighStockRemoval'] = False\n",
    "\n",
    "print(data2[['roll_id','grind_nr','diam_difference']])\n",
    "\n",
    "#roll_list.to_csv(\"highStockRemoval1509.csv\", encoding='utf-8', index=True)"
   ]
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3.10.6 64-bit",
```

```
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.10.6"
  },
  "vscode": {
   "interpreter": {
    "hash": "eac9142a4fc6ce54a1c61b3fe7360112b2e67278aa489ff13858b47584317d1f"
   }
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```