information

Article

SCRO: A Domain Ontology for Describing Steel Cold Rolling Processes towards Industry 4.0

Sadeer Beden *, Qiushi Cao and Arnold Beckmann

Department of Computer Science, Swansea University, Swansea SA2 8PP, UK; qiushi.cao@swansea.ac.uk (Q.C.); a.beckmann@swansea.ac.uk (A.B.)
* Correspondence: sadeer.beden@swansea.ac.uk

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.

Keywords: Industry 4.0; steelmaking; cold rolling; ontology; Ontop

check for updates

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

Academic Editor: Arkaitz Zubiaga

Received: 30 June 2021
Accepted: 27 July 2021
Published: 29 July 2021

1. Introduction

The fourth industrial revolution, also known as Industry 4.0, is full of new concepts, technologies, and innovations with the goal to optimize, digitize, and autonomize industrial processes [1]. It is a vision where machines, products, and processes are connected intelligently and are able to derive meaning from data to make autonomous decisions.

Presently, large industrial machines follow rigid automation protocols which generate vast amounts of data. These data are often not machine-understandable, and are stored in data silos that are often not interconnected yet contain data that are semantically related [2]. A fundamental task to enable Industry 4.0 is to enrich data with semantics to make the data interoperable and machine-understandable. The steel industry is one of many manufacturing domains that are working towards this goal [3–5].

Meanwhile, ontologies have become a prominent methodology for knowledge modeling and capturing domain knowledge, as well as addressing and improving data semantics in various domains. By developing an ontology, we are in essence building a knowledge base within a specific domain [6,7]. In the domain of smart manufacturing, ontologies can play a key role as they are able to provide machine-understandable vocabularies and data exchange between different individuals and processes. Ontologies provide additional functionalities such as stream reasoning which infer new knowledge, and ontology-based data access which allows data to be queried without being physically integrated.

Cold rolling is one of many different steel-making processes within a steel factory. Rolling, in general, processes the greatest tonnage of metals compared to any other metal working technique [8]. The purpose of cold rolling is to compress steel to produce steel coils. During the cold rolling process, the material undergoes deformation, and is compressed by a pair of rolls that rotate in opposite directions under a heavy force. There is a gap between the two rolls that is smaller than the material, thus forcing the material to decrease in size as it passes through the rolls.

Due to strong forces being involved, these rolls are affected by roll wear, where the roll service life and the quality of the product are significantly impacted [9]. To avoid this, the rolls are refurbished regularly, where the diameter of the rolls are marginally reduced to remove the worn surface. One long-term aim of our research is to use the semantically interoperable data to optimize the life of the rolls, improving their total tonnage and yield. In addition, accidents and anomalies that occur, such as overloading, spalling, and incorrect grinding operation [10], can be avoided preemptively once achieving better semantic interoperability.

The goal of this study is to develop an ontology that focuses on modeling the cold rolling processes that occur during steelmaking. Thereby, this paper 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 (https://www.tatasteeleurope.com/ts/, accessed on 26 July 2021). In this case study, we exploit Virtual Knowledge Graphs (VKGs) to access and query the data sets to obtain valuable knowledge.

The remainder of the paper is structured as follows. In Section 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 3, we describe SCRO in detail, including its classes and main concepts. In Section 4, we demonstrate the usefulness of the ontology in an application that uses real-world data. In Section 5, we discuss the validation of SCRO to ensure that the knowledge is accurate. Finally, we reflect on our work and end with a conclusion and future work in Section 6.

2. Literature Review

The W3C has developed a formal ontology language named the Web Ontology Language (OWL) (https://www.w3.org/TR/owl2-overview/, accessed on 26 July 2021) 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.

2.1. Ontologies for Industry 4.0

There have been numerous ontologies developed in recent years to tackle and achieve aspects of Industry 4.0. The Reference Architecture Model for Industry 4.0 (RAMI 4.0) [11], a model that highlights the fundamental requirements for achieving Industry 4.0, has introduced the fundamental concept of an Asset Administration Shell (AAS) as a way for storing and communicating data between assets. A core requirement to enable the AAS concept is to enhance assets with rich data semantics and make them interoperable. As a result, one research direction shifted towards ontology development to capture domain knowledge and concepts to achieve this goal. Previously, we surveyed the scientific contributions of semantic AASs that used ontologies to model the *Information* and *Communication* layers of RAMI 4.0 in [12]. In that paper, we summarized the use cases and technologies used to develop different semantic AASs for the overall goal of improved data semantics and interopability within Industry 4.0. Meanwhile, in this review, we group the Industry 4.0 literature into three categories: *product-related* concepts, *process-related* concepts, and *resource-related* concepts.

Firstly, when looking at product-related concepts, Vegetti et al. [13] developed the Product Ontology (PRONTO) to model *Complex Products* which consider different abstraction levels of product concepts such as *Family* and *Variant*. This approach has benefits and drawbacks. One benefit is that it extends conventional product structure representations, and considers composition and decomposition structures of products from a wide range of different manufacturing environments. One drawback is that there is a lack of capability to refer to existing international standards related to the modeling of product structure, processes, and features. Further research in this direction has been led by Panetto et al. [14] as they developed the ontological model ONTO-PDM which overcomes these shortcomings. This ontology uses the knowledge related to the product technical data to formalize heterogeneous information that is scattered across different organizations [14]. ONTO-PDM also incorporates different standardization initiatives, including the International Electrotechnical Commission (IEC) standards and International Organization for Standardization (ISO) standards. Another example of product-related concept modeling includes the MASON ontology, developed by Lemaignan et al. [15] to create a common semantic net for Industry 4.0. It models three core concepts: *Entities*, *Operations*, and *Resources*, and specifies the product information as *Geometric Entities*, *Raw Material*, and *Cost Entities*. Using the proposed semantic net, they accurately link the product-related concepts with the description of manufacturing process and resources.

Secondly, some ontologies focus on *resource-related* concepts within Industry 4.0. *Resources* in this context are defined as the physical objects within an Industry 4.0 environment that are capable of executing a range of different operations. The MASON ontology mentioned above also studies the notion of *Resources* and deconstructs it into four sub-notions: *Machine-tools, Tools, Human Resources, and Geographical Resources*. The modeling of resources enables estimations of total costs for certain manufacturing activities. Additionally, Borgo and Leitão defines a *Resource* as "an entity that can execute a certain range of jobs, when it is available, as long as its capacity is not exceeded" in [16]. The authors used the Java Agent Development Framework (JADE) to implement their ontology as a part of a multi-agent control system, and concluded that an ontology is a core requirement in handling heterogeneous data generated by manufacturing control applications.

Finally, some ontologies address *process-related* concepts within Industry 4.0. These processes are generally a linear sequence of activities in which raw materials undergo some treatment such as assembly and integration, before converting into the final product. The Process Specification Language (PSL) Ontology [17] was developed by Grüninger et al. to facilitate different methods of exchanging process information between manufacturing systems. Using PSL and first-order logic theories, the authors formalize the concept of a *process*. This formalization has been widely adopted in many different domain applications such as process modeling and process monitoring [17]. Another ontology that focuses on process-related concepts was developed by Cao et al. [18] which formalizes essential concepts and relationships related to condition monitoring. Their ontology contains three sub-modules: *Manufacturing*, *Context*, and *Condition Monitoring*, which are used within a cyber-physical system to enable a case study to model real-time predictive maintenance. The same authors developed a new ontology named the *Manufacturing Predictive Maintenance Ontology (MPMO)* in [19] which uses Semantic Web Rule Language (SWRL) rules to enable ontology reasoning. Using a real-world data set, this ontology is able to detect and predict possible anomalies within an Industry 4.0 manufacturing process.

2.2. Ontologies for the Steel Industry

In the steel industry, ontologies are used as an effective and intelligent knowledge management tool for conceptual modeling and information integration. Leveraging the strong modeling 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 [20] is used to model domain knowledge of steelmaking and enhance the usability and interoperability of the BDAKMS. The developed ontology is further used together with SWRL [21] rules to infer knowledge regarding the demand of raw materials. In [22], 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 [23], 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 the ANSI/ISA-S95 standard to construct the main body of the framework. Another ontology is introduced in [24] 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).

2.3. Ontology Development Methodology

Over the years, several methodologies have been introduced to support the development and engineering of ontologies. The authors of [25] provide a comprehensive survey of different methodologies available for ontology engineering, including: the Toronto 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 decided to use *Ontology Design Pattern* (ODP) [26] methodology.

We have chosen ODPs as it is an approved modeling solution that supports re-usability of good design practices and experiences to solve ontology design problems [26]. 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 use the *Extreme Design (XD) methodology* [27] 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 [28]. This was necessary when developing SCRO as most of the domain knowledge obtained was provided by the involvement and feedback of experts from Tata Steel. Other XD principles include: collaboration and integration, task-oriented design, and test-driven design, which are explained in [27].

We conclude that this design approach offers numerous evident advantages for developing ontologies, including: a faster ontology design process, more flexible design choices, improved interoperability, and high ontology quality [29].

3. SCRO: Steel Cold Rolling Ontology

Most of the domain knowledge mentioned in this section was obtained from a case study with Tata Steel, at the cold rolling mill in the Port Talbot plant. SCRO models the fundamental structure and operations of the rolling processes in the case study. Although SCRO was initially designed for the processes and machines at Tata Steel, it could potentially be reused by other steel manufacturers for knowledge modeling. In this section, we describe SCRO in detail, beginning with the encoding and classes.

3.1. Coding

SCRO was developed using a free, open-source ontology editor and framework called Protégé [30]. We used the latest version, Protégé 5.5.0, that offers a unique interface 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 [31]. Figure 1 displays the structure and the architecture of SCRO, whereas Figure 2 displays the classes, object properties, and data properties.

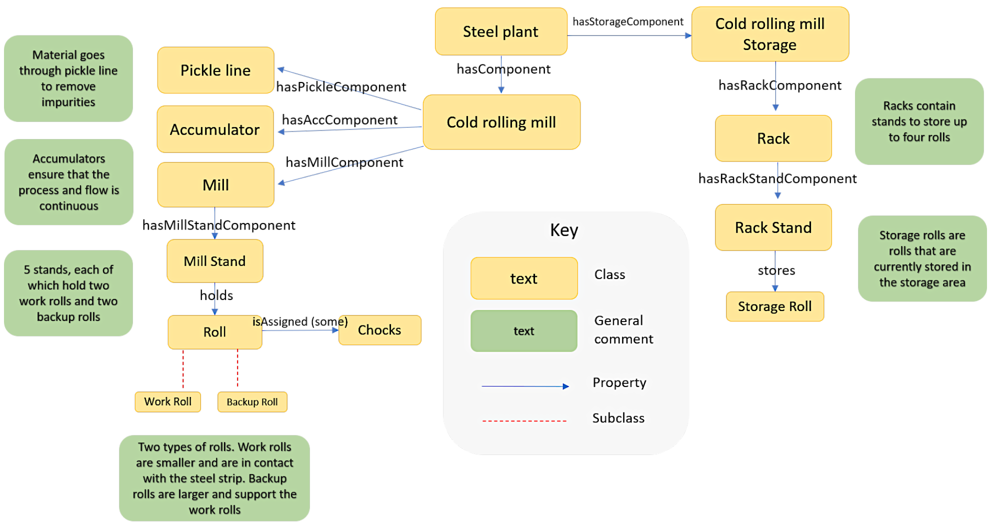
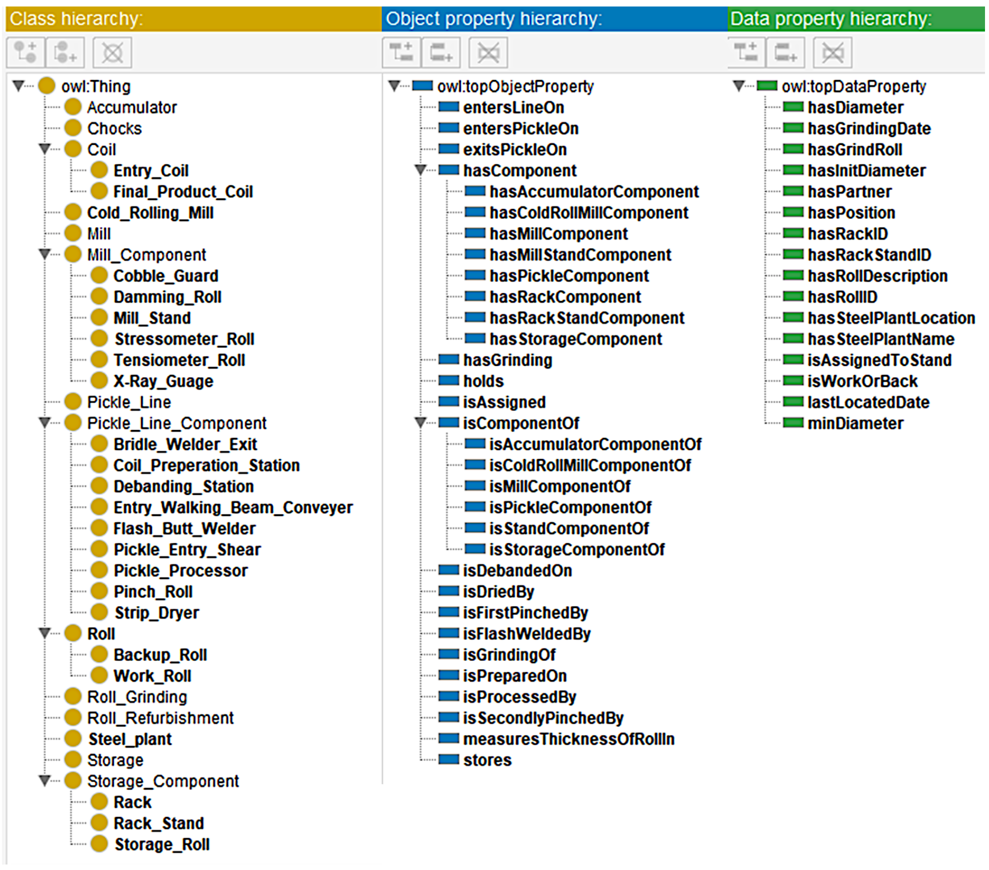

Material goes through pickle line to remove impurities
Accumulators ensure that the process and flow is continuous
5 stands, each of which hold two work rolls and two backup rolls
Pickle line
Accumulator
Mill
hasPickleComponent
hasAccComponent
hasMillComponent
hasMillStandComponent
Mill Stand
holds
Roll
isAssigned (some)
Chocks
Work Roll
Backup Roll
Two types of rolls. Work rolls are smaller and are in contact with the steel strip. Backup rolls are larger and support the work rolls
Steel plant
hasComponent
Cold rolling mill
hasStorageComponent
Cold rolling mill Storage
hasRackComponent
Rack
hasRackStandComponent
Rack Stand
stores
Storage Roll
Racks contain stands to store up to four rolls
Storage rolls are rolls that are currently stored in the storage area
Key
text
Class
text
General comment
Property
Subclass

Figure 1. Structure of SCRO.



Class hierarchy:

- owl:Thing
  - Accumulator
  - Chocks
  - Coil
    - Entry_Coil
    - Final_Product_Coil
  - Cold_Rolling_Mill
  - Mill
  - Mill_Component
    - Cobble_Guard
    - Damming_Roll
    - Mill_Stand
    - Stressometer_Roll
    - Tensiometer_Roll
    - X-Ray_Guage
  - Pickle_Line
  - Pickle_Line_Component
    - Bridle_Welder_Exit
    - Coil_Preperation_Station
    - Debanding_Station
    - Entry_Walking_Beam_Conveyer
    - Flash_Butt_Welder
    - Pickle_Entry_Shear
    - Pickle_Processor
    - Pinch_Roll
    - Strip_Dryer
  - Roll
    - Backup_Roll
    - Work_Roll
  - Roll_Grinding
  - Roll_Refurbishment
  - Steel_plant
  - Storage
  - Storage_Component
    - Rack
    - Rack_Stand
    - Storage_Roll

Object property hierarchy:

- owl:topObjectProperty
  - entersLineOn
  - entersPickleOn
  - exitsPickleOn
  - hasComponent
    - hasAccumulatorComponent
    - hasColdRollMillComponent
    - hasMillComponent
    - hasMillStandComponent
    - hasPickleComponent
    - hasRackComponent
    - hasRackStandComponent
    - hasStorageComponent
  - hasGrinding
  - holds
  - isAssigned
  - isComponentOf
    - isAccumulatorComponentOf
    - isColdRollMillComponentOf
    - isMillComponentOf
    - isPickleComponentOf
    - isStandComponentOf
    - isStorageComponentOf
  - isDebandedOn
  - isDriedBy
  - isFirstPinchedBy
  - isFlashWeldedBy
  - isGrindingOf
  - isPreparedOn
  - isProcessedBy
  - isSecondlyPinchedBy
  - measuresThicknessOfRollIn
  - stores

Data property hierarchy:

- owl:topDataProperty
  - hasDiameter
  - hasGrindingDate
  - hasGrindRoll
  - hasInitDiameter
  - hasPartner
  - hasPosition
  - hasRackID
  - hasRackStandID
  - hasRollDescription
  - hasRollID
  - hasSteelPlantLocation
  - hasSteelPlantName
  - isAssignedToStand
  - isWorkOrBack
  - lastLocatedDate
  - minDiameter

Figure 2. Classes, object properties, and data properties of SCRO.

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 [32]. These are excluded from Figure 2 but play an important role in SCRO.

3.3. Classes

There are many processes and components on the shop floor that are fundamental for cold rolling, as depicted in Figure 3. 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.

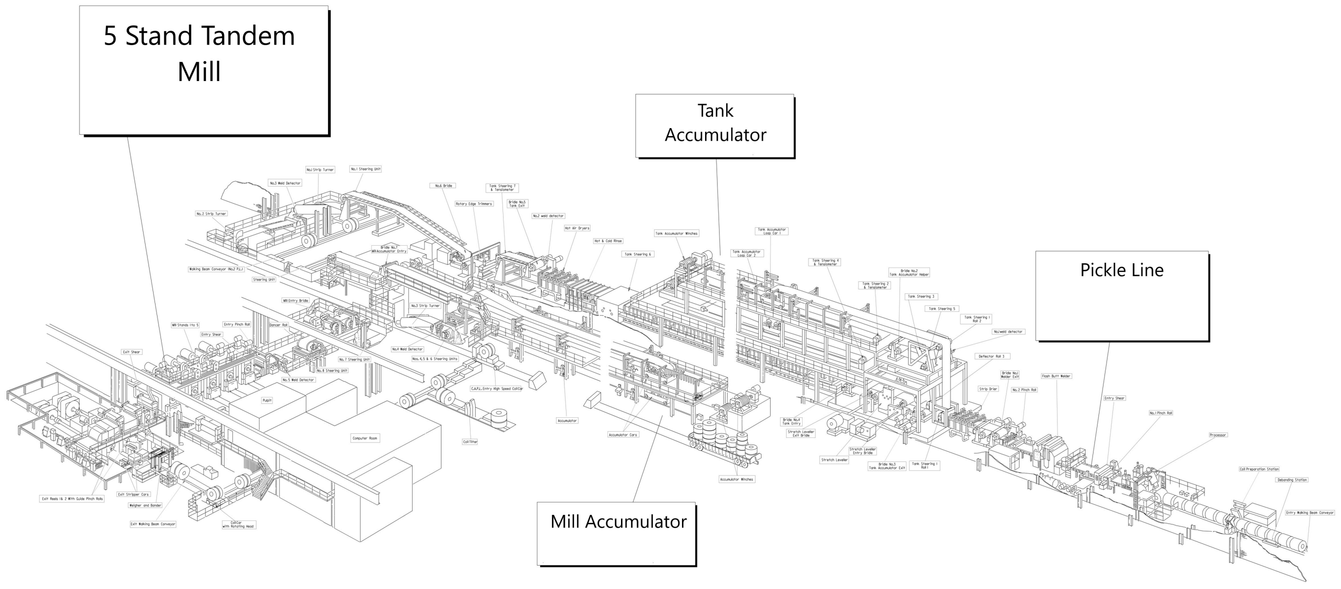


Figure 3. The big picture of the cold rolling processes at the Port Talbot plant provided by Tata Steel. Copyright © 2021 all rights reserved.

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 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 is 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 in Table 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 means 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 1. Figure 4 displays the hierarchy of all the classes, generated by the Protégé tool.

Table 1. Description of SCRO classes.

SCRO Classes	Description
Accumulator	Manage the speed of the rolling processes to ensure flow is continuous
Chocks	Attached to rolls. Chocks contain bearings that allow rolls to rotate
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 where thickness of the steel strip is reduced
Mill_Component	**Superclass of all Mill components**
Cobble_Guard	Component that reduces 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_Gauge	Measures the thickness of the steel strip
Pickle_Line	Process where the entry coil undergoes surface pickling
Pickle_Line_Component	**Superclass of all Pickle components**
Bridle_Welder_Exit	Mill exit equipment that the strip uses to exit the pickling process
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 together and welds the ends of the workpiece
Pickle_Entry_Shear	Machine that cuts rolls to desired size
Pickle_Processor	Processes the coil and minimizes the tendency for coils to break
Pinch_Roll	Machine that holds and moves the strip
Strip_Dryer	Removes excess water from the strip to prevent rusting
Roll	**Superclass of the two types of 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 thickness of steel during milling
Roll_Grinding	Contains previous grinding data of rolls
Roll_Refurbishment	Process where rolls are sent to be refurbished
Steel_Plant	Denotes the whole steel plant
Storage	Section of the cold rolling mill where assets (e.g., unused rolls) are stored
Storage_Component	**Superclass of the Storage components**
Rack	Contains stands for rolls to be stored
Rack_Stand	Stores one storage roll
Storage_Roll	A roll that is not currently being used and is stored away

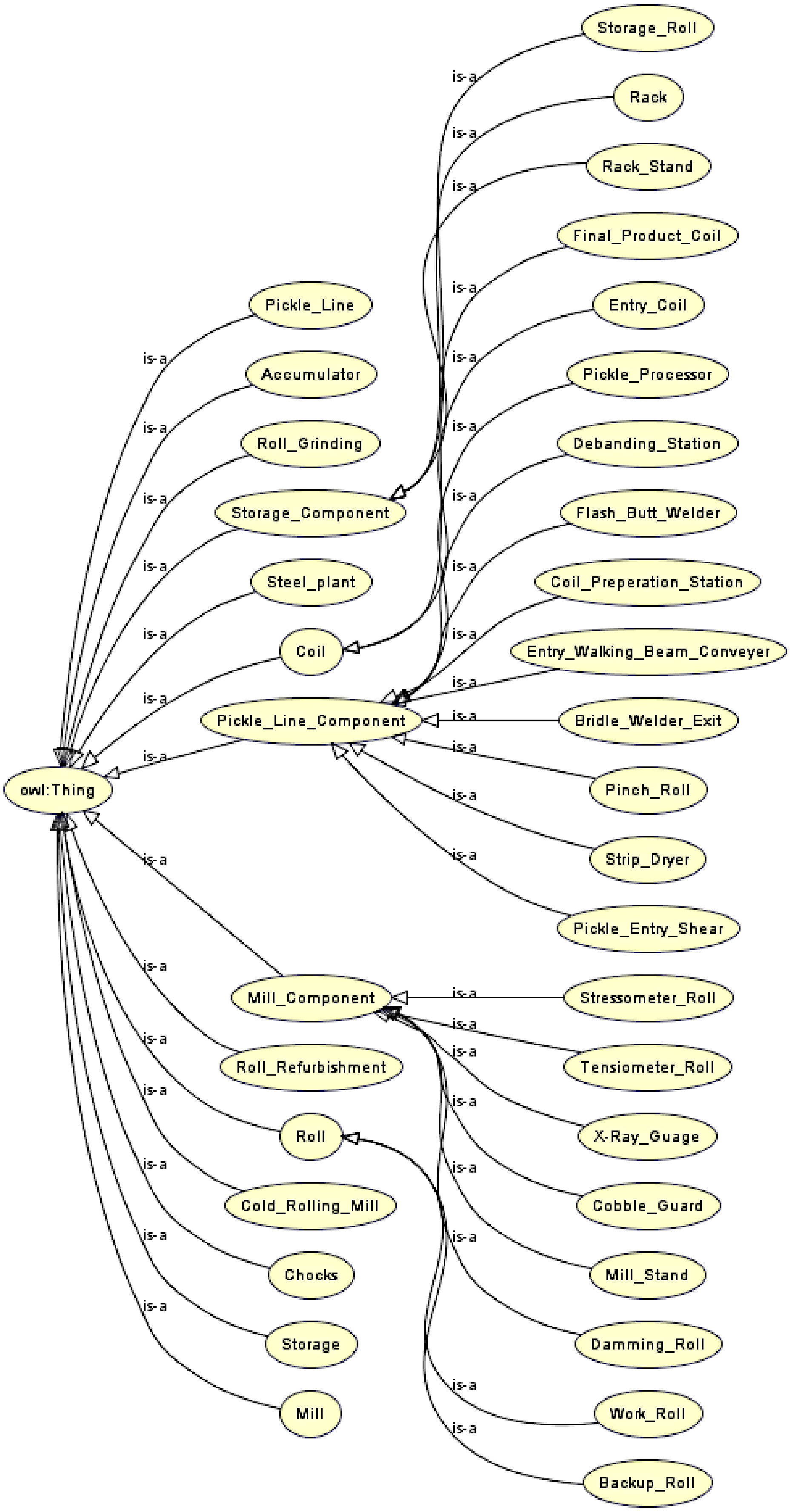


Figure 4. Hierarchy of all the classes in SCRO generated by OWLViz plugin in Protégé.

3.4. Object and Data Properties

To semantically describe the relations between classes, it is important that we specify the domain and ranges of the properties. These properties are clarified below:

- entersLineOn(object1, object2) where object1 is an *Entry_Coil* and object2 is an *Entry_Walking_Beam.*
- entersPickleOn(object1, object2) where object1 is an *Entry_Coil* and object2 is a *Pickle_Entry_Shear.*
- exitsPickleOn(object1, object2) where object1 is an *Entry_Coil* and object2 is a *Bridle_Welder_Exit.*
- hasComponent(object1, object2) where object1 and object2 are left undefined as this is the superclass for all hasComponents mentioned below.
- hasAccumaltorComponent(object1, object2) where object1 is a *Cold_Rolling_Mill* and object2 is an *Accumulator.*
- hasColdRollMillComponent(object1, object2) where object1 is a *Steel_plant* and object2 is a *Cold_Rolling_Mill.*
- hasMillComponent(object1, object2) where object1 is a *Cold_Rolling_Mill* and object2 is a *Mill.*
- hasMillStandComponent(object1, object2) where object1 is a *Mill* and object2 is a *Mill_Stand.*
- hasPickleComponent(object1, object2) where object1 is a *Cold_Rolling_Mill* and object2 is a *Pickle_Line.*
- hasRackComponent(object1, object2) where object1 is a *Storage* and object2 is a *Rack.*
- hasRackStandComponent(object1, object2) where object1 is a *Rack* and object2 is a *Rack_Stand.*
- hasStorageComponent(object1, object2) where object1 is a *Steel_Plant* and object2 is a *Storage.*
- hasGrinding(object1, object2) where object1 is a *Roll* and object2 is a *Roll_Grinding.*
- holds(object1, object2) where object1 is a *Mill_Stand* and object2 is a *Storage_Roll.*
- isAssigned(object1, object2) where object1 is a *Roll* and object2 are *Chocks.*
- The superclass *isComponentOf* which is the inverse of *hasComponent*, as well as all of its subclasses.
- isDebandedOn(object1, object2) where object1 is an *Entry_Coil* and object2 is a *Debanding_station.*
- isDriedBy(object1, object2) where object1 is a *Entry_Coil* and object2 is a *Strip_Dryer.*
- isFrstPinchedBy(object1, object2) where object1 is a *Entry_Coil* and object2 is a *Pinch_Roll.*
- isFlashWeldedBy(object1, object2) where object1 is an *Entry_Coil* and object2 is a *Flash_Butt_Welder.*
- isPreparedOn(object1, object2) where object1 is an *Entry_Coil* and object2 is a *Coil_Preparation_Station.*
- isProcessedBy(object1, object2) where object1 is an *Entry_Coil* and object2 is a *Pickle_Processor.*
- MeasuresThicknessOfRollIn (object1, object2) where object1 is an *X-Ray_Gauge* and object2 is a *Mill_Stand.*
- stores(object1, object2) where object1 is a *Rack_Stand* and object2 is a *Storage_Roll.*

Similarly, this is carried out with the data proprieties in the ontology:

- hasDiameter(object, datatype) where object is Roll and datatype is xsd:double.
- hasGrindingDate(object, datatype) where object is *Time instant* and datatype is xsd:date.
- hasGrindRoll(object, datatype) where object is *Roll_Grinding* and datatype is xsd:integer.
- hasInitDiameter(object, datatype) where object is *Roll* and datatype is xsd:double.
- hasPartner(object, datatype) where object is *Roll* and datatype is xsd:integer.
- hasPosition(object, datatype) where object is *Roll* and datatype is xsd:string.
- hasRackID(object, datatype) where object is *Rack* and datatype is xsd:integer.
- hasStackStandID(object, datatype) where object is *Rack_Stand* and datatype is xsd:integer.

- hasRollDescription(object, datatype) where object is *Storage_Roll* and datatype is xsd:String.
- hasRollID(object, datatype) where object is *Roll* and datatype is xsd:integer.
- hasSteelPlantLocation(object, datatype) where object is *Steel_Plant* and datatype is xsd:String.
- hasSteelPlantName(object, datatype) where object is *Steel_Plant* and datatype is xsd:String.
- isAssignedToStand(object, datatype) where object is *Roll* and datatype is xsd:integer.
- isWorkOrBack(object, datatype) where object is *Roll* and datatype is xsd:string.
- lastLocatedDate(object, datatype) where object is *Time instant* and datatype is xsd:dateTime.
- minDiameter(object, datatype) where object is *Roll* and datatype is xsd:double.

4. Application

4.1. Data Set

We test and evaluate SCRO through a real-world industrial application. Within this industrial application, a collection of real-world data sets is provided by Tata Steel. These data sets specifically come from their 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. These data are considerable in quantity and located in different tables within the database. For our research, we focused on three specific tables: the *Roll*, *Roll Grinding*, and *Roll Storage* tables. These tables contain many fields of data that we have chosen not to include in SCRO. Instead, we only include the fields we acknowledged as the core fields, such as *RollID* and *Diameter*, but not *SupplierID*. The domain experts from Tata Steel agreed with this approach. Table 2 describes the tables in the database, including the fields, data types, and descriptions.

Secondly, the data sets also contain dynamic data from the cold rolling mill that are read through 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 include the chemistry of the rolls, temperature, pressure, and much more.

Table 2. Description of all three tables from the data sets.

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 the position 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 each roll**
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 that stores the data 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., if it is a new roll or damaged roll
Actual_Diameter	Double	Stores the value of the diameter of the roll

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, integration is required. However, it can be costly to join, clean, and homogenize the data. To avoid this, in recent years, VKGs have been developed as a paradigm for data integration and access by exploiting data virtualization [2]. This is achieved by creating graphs on top of relational databases where the data are not physically moved to another database and instead kept and viewed at a virtual level [33]. Virtualization is achieved by creating an ontology, and linking the data sources to the ontology via *Mappings*. 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 [34], Morph [35], and Ontop [33]. For our approach, we have adopted the Ontop framework.

4.2. Ontop Framework

The Ontop framework (https://ontop-vkg.org//, accessed on 26 July 2021) 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 [36]. Additionally, it supports widely used standards, including: (1) ontologies: Ontop supports the *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 [37].

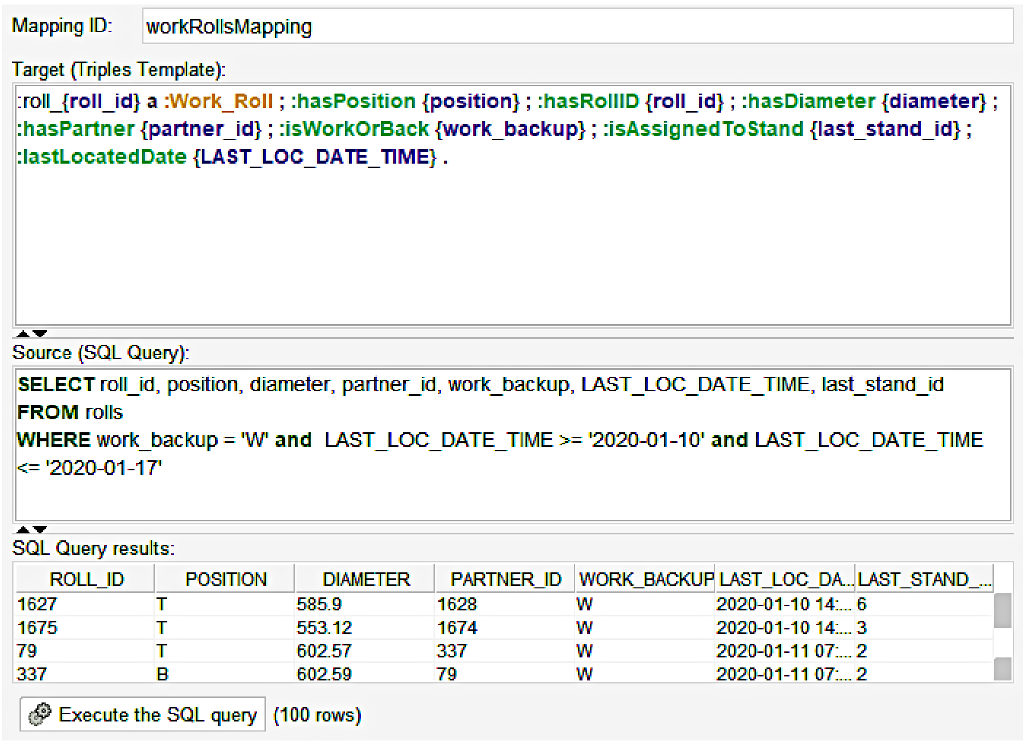
4.3. Mappings

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 [38]. For our mappings, as mentioned above, we used the Ontop mapping language which is fully interoperable with R2RML [36].

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 (MBs) which automatically generate a mapping for a data source [2]. These mappings follow a set of rules based on the W3C Direct Mapping specification to generate RDF graphs [39]. Ontop bootstrapper and BootOX [40] are two examples of existing MBs. A benchmark suite named Relational-to-Ontology Data Integration (RODI) [41] 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 with 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 5 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. As 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 5, 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 the query to restrict the *last_loc_date_time* timestamp value to a seven-day period.

We can then click “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 from the 10–17 of January 2020.



workRollsMapping

```
:roll_{roll_id} a :Work_Roll ; :hasPosition {position} ; :hasRollID {roll_id} ; :hasDiameter {diameter} ;
:hasPartner {partner_id} ; :isWorkOrBack {work_backup} ; :isAssignedToStand {last_stand_id} ;
:lastLocatedDate {LAST_LOC_DATE_TIME} .
```

```
SELECT roll_id, position, diameter, partner_id, work_backup, LAST_LOC_DATE_TIME, last_stand_id
FROM rolls
WHERE work_backup = 'W' and  LAST_LOC_DATE_TIME >= '2020-01-10' and LAST_LOC_DATE_TIME
<= '2020-01-17'
```

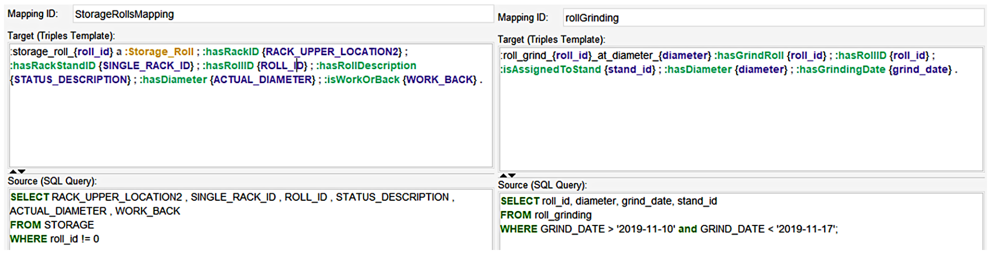
ROLL_ID	POSITION	DIAMETER	PARTNER_ID	WORK_BACKUP	LAST_LOC_DA...	LAST_STAND_...
1627	T	585.9	1628	W	2020-01-10 14:...	6
1675	T	553.12	1674	W	2020-01-10 14:...	3
79	T	602.57	337	W	2020-01-11 07:...	2
337	B	602.59	79	W	2020-01-11 07:...	2

Figure 5. Ontop mapping for work rolls.

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 (https://www.w3.org/TR/turtle/, accessed on 26 July 2021). 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 a *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 full stop, 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 is shown in Figure 5. 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 that the **:roll_{roll_id} a :Work_roll** becomes **:roll_{roll_id} a :Backup_roll** and the *work_backup* field in the SQL *WHERE* clause is set to equal *”B”*.

Figure 6 depicts two other mappings. The mapping on the left manages and links SCRO with the *roll_storage* data set, whereas the mapping on the right manages historical grinding values of rolls from the *roll_grinding* data set.


Mapping ID: StorageRollsMapping

Target (Triples Template):

```
:storage_roll_{roll_id} a :Storage_Roll ; :hasRackID {RACK_UPPER_LOCATION2} ;
:hasRackStandID {SINGLE_RACK_ID} ; :hasRollID {ROLL_ID} ; :hasRollDescription
{STATUS_DESCRIPTION} ; :hasDiameter {ACTUAL_DIAMETER} ; :isWorkOrBack {WORK_BACK} .
```

Source (SQL Query):

```
SELECT RACK_UPPER_LOCATION2 , SINGLE_RACK_ID , ROLL_ID , STATUS_DESCRIPTION ,
ACTUAL_DIAMETER , WORK_BACK
FROM STORAGE
WHERE roll_id != 0
```

Mapping ID: rollGrinding

Target (Triples Template):

```
:roll_grind_{roll_id}_at_diameter_{diameter} :hasGrindRoll {roll_id} ; :hasRollID {roll_id} ;
:isAssignedToStand {stand_id} ; :hasDiameter {diameter} ; :hasGrindingDate {grind_date} .
```

Source (SQL Query):

```
SELECT roll_id, diameter, grind_date, stand_id
FROM roll_grinding
WHERE GRIND_DATE > '2019-11-10' and GRIND_DATE < '2019-11-17';
```

Figure 6. Ontop mapping for grindings and storage rolls.

4.4. SPARQL

We use SPARQL (https://www.w3.org/TR/sparql11-overview/, accessed on 26 July 2021) 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 [38]. As described above, the RDF triples are generated by the Ontop mappings that are depicted in Figures 5 and 6, 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 [42], which provides an interactive interface that generates components to build SPARQL queries. However, we decided to write our SPARQL queries manually using a text editor provided by the Protégé software. Below are some queries that we developed to query the data.

Listing 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 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 diameter values matching other rolls from either data set. Given a scenario where a roll gets damaged, we can use this query to see if there are other rolls in both the storage data set and roll data set that have the same diameter as the damaged roll.

Listing 1. Diameter values which appear for more than two rolls.

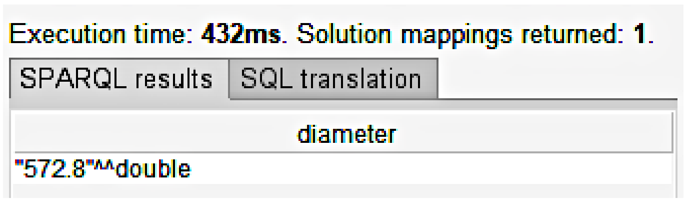
```
PREFIX : http://www.semanticweb.org/sadee/ontologies/2021/1/SCRO#
PREFIX time: http://www.w3.org/2006/time#

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 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.

Then, 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 1, we have chosen to select a variable called *?diameter* to show that 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 subjects 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 roll grinding 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 our *Select* clause. In this example, we are only printing out the diameter variable.

Figure 7 displays the results of this SPARQL query. The results show that *572.8* is the only diameter value that three or more rolls have that were last located from the 10–17th of January 2020. We create another query to print out these rolls in Listing 2.


```
Execution time: 432ms. Solution mappings returned: 1.
```

diameter
"572.8"^^double

Figure 7. SPARQL result from Listing 1.

Listing 2. All rolls that have a diameter of 572.8.

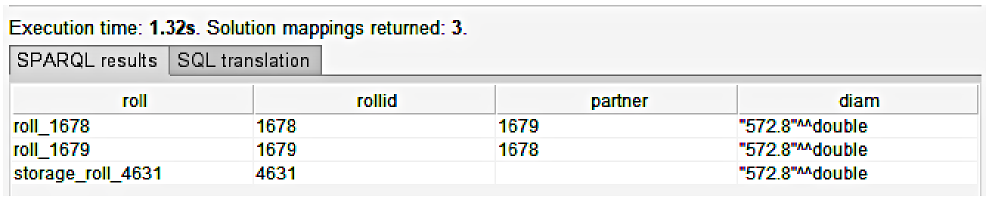
```
PREFIX time: http://www.w3.org/2006/time#
PREFIX : http://www.semanticweb.org/sadee/ontologies/2021/1/SCRO#

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
```

Listing 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 values 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.

Then, we filtered the *?diam* value to only return rolls that have a diameter value of *572.8*, which was the result from the first SPARQL query in Listing 1. Figure 8 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 an 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 are stored separately from active roll data, so this query skips the need for integration.



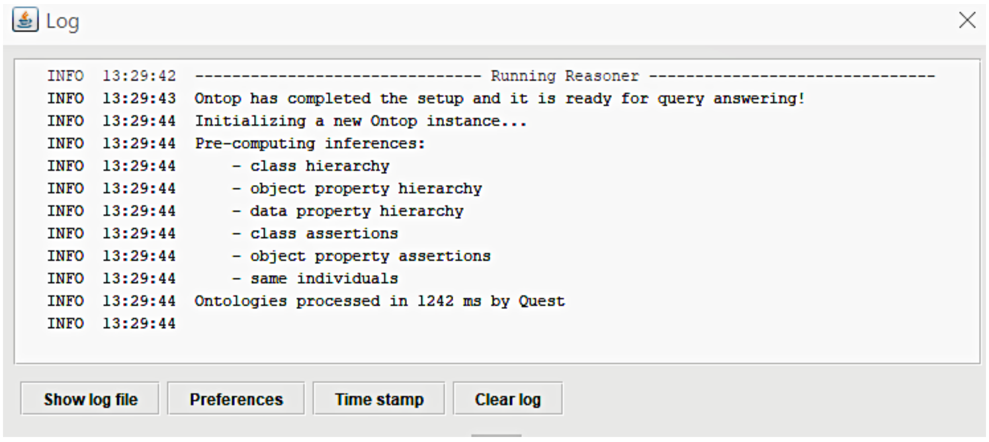
Execution time: **1.32s**. Solution mappings returned: **3**.

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 8. SPARQL result from Listing 2.

5. 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 [43]. In addition, a well-known ontology validation tool known as the "Ontology Pitfall Scanner" (OOPS) [44] has been developed to validate ontologies by detecting common pitfalls aligned to a dimension classification developed in [45]. 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 carried out. Figure 9 shows that there are no inconsistency or correctness errors in the console log when running the stream reasoner, allowing us to query the data.



```
INFO  13:29:42  ------------------------------- Running Reasoner -------------------------------
INFO  13:29:43  Ontop has completed the setup and it is ready for query answering!
INFO  13:29:44  Initializing a new Ontop instance...
INFO  13:29:44  Pre-computing inferences:
INFO  13:29:44      - class hierarchy
INFO  13:29:44      - object property hierarchy
INFO  13:29:44      - data property hierarchy
INFO  13:29:44      - class assertions
INFO  13:29:44      - object property assertions
INFO  13:29:44      - same individuals
INFO  13:29:44  Ontologies processed in 1242 ms by Quest
INFO  13:29:44
```

Figure 9. The console displaying no inconsistency or incorrectness messages when starting the Ontop stream reasoner.

5.1. Ontology Pitfall Scanner

Different pitfalls have different impacts and levels of importance. Because of this, OOPS categorizes 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 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 [46], when using OWL, it is best practice not to specify the domain and ranges of superclasses but instead mention them in their respective 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 [46], but rather viewed as *axioms* for reasoning. As a result of this, we have decided 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.

5.2. Expert Knowledge Validation

As this work is 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. Additionally, we gained better understanding of the future goals that the steel industry is working towards and its current limitations, one of which being data integration. One benefit of using ontologies is using the knowledge graph paradigm to exploit data virtualization which we also presented in the demonstration.

6. Conclusions

To conclude, this paper presents a novel steel cold rolling 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, which is the first step towards achieving Industry 4.0. 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 the core of 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.

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 of other companies. A key future goal will be to look at more cold rolling plants and compare any differences in processes and machinery to generalize the ontology, and add flexibility. Another future goal is to enhance the logic axioms for formalization of the knowledge. Presently, we have only mentioned basic axioms that show the relationships between classes and their properties. This paper does not include any logical constraints or logical connectives, whereas the ontology currently contains a few constraints, such as work roll and backup roll classes being disjointed. 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.

Author Contributions: Conceptualization, S.B. and Q.C.; methodology, S.B.; software, S.B.; validation, S.B., Q.C. and A.B.; formal analysis, S.B.; investigation, S.B.; resources, S.B.; data curation, S.B.; writing—original draft preparation, S.B.; writing—review and editing, S.B., Q.C. and A.B.; visualization, S.B.; supervision, A.B.; project administration, A.B.; funding acquisition, A.B. All authors have read and agreed to the published version of the manuscript.

Funding: S. Beden was supported by the Engineering and Physical Sciences Research Council (grant number EP/T517537/1) and by Tata Steel. Q. Cao and A. Beckmann (in part) were supported by the Engineering and Physical Sciences Research Council (grant number EPSRC EP/S018107/1).

Data Availability Statement: Not applicable.

Acknowledgments: We would like to acknowledge Steve Thornton as the domain expert from Tata Steel during the development of the ontology.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Horvath, D.; Szabo, R. Driving forces and barriers of Industry 4.0: Do multinational and small and medium-sized companies have equal opportunities? *Technol. Forecast. Soc. Chang.* **2019**, *146*, 119–132. [CrossRef]
2. Xiao, G.; Ding, L.; Cogrel, B.; Calvanese, D. Virtual Knowledge Graphs: An Overview of Systems and Use Cases. *Data Intell.* **2019**, *1*, 201–223. [CrossRef]
3. Peters, H. *How Could Industry 4.0 Transform the Steel Industry*; Future Steel Forum: Prague, Czech Republic, 2017.
4. Miśkiewicz, R.; Wolniak, R. Practical Application of the Industry 4.0 Concept in a Steel Company. *Sustainability* **2020**, *12*, 5776.
5. Naujok, N.; Stamm, H. *Industry 4.0 in Steel: Status, Strategy, Roadmap and Capabilities*; Future Steel Forum: Prague, Czech Republic, 2017.
6. Noy, N.F.; McGuinness, D.L. *Ontology Development 101: A Guide to Creating Your First Ontology*; Technical Report; Knowledge Systems Laboratory Stanford University: Stanford, CA, USA, 2001.
7. Cao, Q.; Zanni-Merk, C.; Reich, C. Ontologies for manufacturing process modeling: A survey. In *International Conference on Sustainable Design and Manufacturing*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 61–70.
8. Roberts, W.L. *Cold Rolling of Steel*; Routledge: London, UK, 1978.
9. Schroeder, D.K.H. A Basic Understanding of the Mechanics of Rolling Mill Rolls. 2003. Available online: http://docshare04.docshare.tips/files/15568/155680328.pdf (accessed on 26 July 2021)
10. Ray, A.; Mishra, K.; Das, G.; Chaudhary, P. Life of rolls in a cold rolling mill in a steel plant-operation versus manufacture. *Eng. Fail. Anal.* **2000**, *7*, 55–67. [CrossRef]
11. Zezulka, F.; Marcon, P.; Vesely, I.; Sajdl, O. Industry 4.0—An Introduction in the phenomenon. *IFAC-PapersOnLine* **2016**, *49*, 8–12. [CrossRef]
12. Beden, S.; Cao, Q.; Beckmann, A. Semantic Asset Administration Shells in Industry 4.0: A Survey. In Proceedings of the 2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS), Victoria, BC, Canada, 10–13 May 2021; pp. 31–38. [CrossRef]
13. Vegetti, M.; Henning, G.P.; Leone, H.P. Product ontology: Definition of an ontology for the complex product modelling domain. In Proceedings of the Mercosur Congress on Process Systems Engineering, Rio de Janeiro, Brazil, 14–18 August 2005.
14. Panetto, H.; Dassisti, M.; Tursi, A. ONTO-PDM: Product-driven ONTOlogy for Product Data Management interoperability within manufacturing process environment. *Adv. Eng. Inform.* **2012**, *26*, 334–348.
15. Lemaignan, S.; Siadat, A.; Dantan, J.Y.; Semenenko, A. MASON: A proposal for an ontology of manufacturing domain. In Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06), Prague, Czech Republic, 15–16 June 2006; IEEE: Piscataway, NY, USA, 2006; pp. 195–200.
16. Borgo, S.; Leitão, P. Foundations for a core ontology of manufacturing. In *Ontologies*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 751–775.
17. Grüninger, M. Using the PSL ontology. In *Handbook on Ontologies*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 423–443.
18. Cao, Q.; Giustozzi, F.; Zanni-Merk, C.; de Bertrand de Beuvron, F.; Reich, C. Smart condition monitoring for industry 4.0 manufacturing processes: An ontology-based approach. *Cybern. Syst.* **2019**, *50*, 82–96. [CrossRef]
19. Cao, Q.; Samet, A.; Zanni-Merk, C.; de Bertrand de Beuvron, F.; Reich, C. Combining chronicle mining and semantics for predictive maintenance in manufacturing processes. *Semant. Web* **2020**, *11* , 927–948.
20. Bao, Q.; Wang, J.; Cheng, J. Research on ontology modeling of steel manufacturing process based on big data analysis. In Proceedings of the MATEC Web of Conferences, Amsterdam, The Netherlands, 23–25 March 2016; EDP Sciences: Ulis, France, 2016; Volume 45, p. 04005.
21. Horrocks, I.; Patel-Schneider, P.F.; Boley, H.; Tabet, S.; Grosof, B.; Dean, M. SWRL: A semantic web rule language combining OWL and RuleML. *W3C Memb. Submiss.* **2004**, *21*, 1–31.
22. Wang, X.; Wong, T.; Fan, Z.P. Ontology-based supply chain decision support for steel manufacturers in China. *Expert Syst. Appl.* **2013**, *40*, 7519–7533.

23. Dobrev, M.; Gocheva, D.; Batchkova, I. An ontological approach for planning and scheduling in primary steel production. In Proceedings of the 2008 4th International IEEE Conference Intelligent Systems, Varna, Bulgaria, 6–8 September 2008 ; IEEE: Piscataway, NY, USA, 2008; Volume 1, pp. 6–14.
24. Ugwu, O.; Anumba, C.J.; Thorpe, A.; Arciszewski, T. Building knowledge level ontology for the collaborative design of steel frame structures. In *Advances in Intelligent Computing in Engineering—Proceedings of the 9th International Workshop of the European Group of Intelligent Computing in Engineering (EG-ICE), Darmstadt, Germany, 1–3 August 2002*; Technische Universitat Darmstadt: Darmstadt, Germany, 2002; pp. 1–3.
25. Jones, D.M.; Bench-Capon, T.J.M.; Visser, P.R.S. Methodologies for Ontology Development. 2007. Available online: https://cgi.csc.liv.ac.uk/~tbc/publications/itknows.pdf (accessed on 26 July 2021).
26. Gangemi, A.; Presutti, V. *Ontology Design Patterns*; Springer: Berlin/Heidelberg, Germany, 2009 ; pp. 221–243. [CrossRef]
27. Presutti, V.; Daga, E.; Gangemi, A.; Blomqvist, E. eXtreme design with content ontology design patterns. In Proceedings of the Workshop on Ontology Patterns, Washington, DC, USA, 25–29 October 2009; pp. 83–97.
28. Beck, K.; Gamma, E. *Extreme Programming Explained: Embrace Change*; Apt Book Series; An Alan, R., Ed.; Addison-Wesley: Boston, MA, USA, 2000.
29. Blomqvist, E.; Presutti, V.; Daga, E.; Gangemi, A. Experimenting with eXtreme Design. In *Knowledge Engineering and Management by the Masses*; Cimiano, P., Pinto, H.S., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 120–134.
30. Musen, M.A. The protégé project: A look back and a look forward. *AI Matters* **2015**, *1*, 4–12. [CrossRef]
31. Antoniou, G.; van Harmelen, F. *Web Ontology Language: OWL*; Springer: Berlin/Heidelberg, Germany , 2004; [CrossRef]
32. Hobbs, J.R.; Pan, F. Time ontology in OWL. *W3C Work. Draft* **2006**, *27*, 133.
33. Xiao, G.; Lanti, D.; Kontchakov, R.; Komla-Ebri, S.; Güzel-Kalaycı, E.; Ding, L.; Corman, J.; Cogrel, B.; Calvanese, D.; Botoeva, E. The Virtual Knowledge Graph System Ontop. In *The Semantic Web—ISWC 2020*; Pan, J.Z., Tamma, V., d'Amato, C., Janowicz, K., Fu, B., Polleres, A., Seneviratne, O., Kagal, L., Eds.; Springer International Publishing: Cham, Switzerland , 2020; pp. 259–277.
34. Calvanese, D.; Giacomo, G.D.; Lembo, D.; Lenzerini, M.; Poggi, A.; Rodriguez-Muro, M.; Rosati, R.; Ruzzi, M.; Savo, D.F. The MASTRO system for ontology-based data access. *Semant. Web* **2011**, *2*, 43–53.
35. Priyatna, F.; Corcho, O.; Sequeda, J. Formalisation and Experiences of R2RML-Based SPARQL to SQL Query Translation Using Morph. In *WWW '14, Proceedings of the 23rd International Conference on World Wide Web, Seoul, Korea, 7–11 April 2014*; Association for Computing Machinery: New York, NY, USA, 2014; pp. 479–490. [CrossRef]
36. Calvanese, D.; Cogrel, B.; Komla-Ebri, S.; Kontchakov, R.; Lanti, D.; Rezk, M.; Rodriguez-Muro, M.; Xiao, G. Ontop: Answering SPARQL queries over relational databases. *Semant. Web* **2016**, *8*, 471–487. [CrossRef]
37. Bagosi, T.; Calvanese, D.; Hardi, J.; Komla-Ebri, S.; Lanti, D.; Rezk, M.; Rodríguez-Muro, M.; Slusnys, M.; Xiao, G. The Ontop Framework for Ontology Based Data Access. In *The Semantic Web and Web Science*; Zhao, D., Du, J., Wang, H., Wang, P., Ji, D., Pan, J.Z., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; pp. 67–77.
38. Rodríguez-Muro, M.; Rezk, M. Efficient SPARQL-to-SQL with R2RML mappings. *J. Web Semant.* **2015**, *33*, 141–169. [CrossRef]
39. Sequeda, J.F.; Tirmizi, S.H.; Corcho, O.; Miranker, D.P. Review: Survey of Directly Mapping Sql Databases to the Semantic Web. *Knowl. Eng. Rev.* **2011**, *26*, 445–486. [CrossRef]
40. Jiménez-Ruiz, E.; Kharlamov, E.; Zheleznyakov, D.; Horrocks, I.; Pinkel, C.; Skjæveland, M.G.; Thorstensen, E.; Mora, J. BootOX: Practical Mapping of RDBs to OWL 2. In *The Semantic Web-ISWC 2015*; Arenas, M., Corcho, O., Simperl, E., Strohmaier, M., d'Aquin, M., Srinivas, K., Groth, P., Dumontier, M., Heflin, J., Thirunarayan, K., Staab, S., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 113–132.
41. Pinkel, C.; Binnig, C.; Jiménez-Ruiz, E.; May, W.; Ritze, D.; Skjæveland, M.; Solimando, A.; Kharlamov, E. RODI: A Benchmark for Automatic Mapping Generation in Relational-to-Ontology Data Integration. In *European Semantic Web Conference*; Springer: Cham, Switzerland, 2015; pp. 21–37. [CrossRef]
42. Soylu, A.; Kharlamov, E.; Zheleznyakov, D.; Jimenez-Ruiz, E.; Giese, M.; Skjaeveland, M.G.; Hovland, D.; Schlatte, R.; Brandt, S.; Lie, H.; et al. OptiqueVQS: A visual query system over ontologies for industry. *Semant. Web* **2018**, *9*, 627–660. [CrossRef]
43. Brank, J.; Mladenic, D.; Grobelnik, M. Gold standard based ontology evaluation using instance assignment. In Proceedings of the EON 2006 Workshop, Edinburgh, UK, 22 May 2006.
44. Poveda-Villalón, M.; Gómez-Pérez, A.; Suárez-Figueroa, M.C. OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation. *Int. J. Semant. Web Inf. Syst. IJSWIS* **2014**, *10*, 7–34.
45. Gómez-Pérez, A. Ontology evaluation. In *Handbook on Ontologies*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 251–273.
46. Horridge, M.; Knublauch, H.; Rector, A.; Stevens, R.; Wroe, C. *A Practical Guide to Building OWL Ontologies Using the Prot'eg'e-OWL Plugin and CO-ODE Tools*; University of Manchester: Manchester, UK, 2004.